Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 11, 2018

M. Boucadair
C. Jacquenet
Orange

October 8, 2017

RADIUS Extensions for Network-Assisted Multipath TCP (MPTCP)
draft-boucadair-mptcp-radius-05

Abstract

Because of the lack of Multipath TCP (MPTCP) support at the server side, some service providers now consider a network-assisted model that relies upon the activation of a dedicated function called MPTCP Conversion Point (MCP). Network-assisted MPTCP deployment models are designed to facilitate the adoption of MPTCP for the establishment of multi-path communications without making any assumption about the support of MPTCP by the communicating peers. MCPs located in the network are responsible for establishing multi-path communications on behalf of endpoints, thereby taking advantage of MPTCP capabilities to achieve different goals that include (but are not limited to) optimization of resource usage (e.g., bandwidth aggregation), of resiliency (e.g., primary/backup communication paths), and traffic offload management.

This document specifies a new Remote Authentication Dial-In User Service (RADIUS) attributes that carry the IP addresses that will be returned to authorized users to reach one or multiple MCPs.

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 https://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 April 11, 2018.

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].

Network-assisted MPTCP deployment models are designed to facilitate the adoption of MPTCP for the establishment of multi-path communications without making any assumption about the support of MPTCP by the communicating peers. This deployment scenario relies on MPTCP proxies located on both the CPE and network sides (Figure 1).

MPTCP proxies are responsible for establishing multi-path communications on behalf of endpoints, thereby taking advantage of MPTCP capabilities to optimize resource usage to achieve different goals that include (but are not limited to) bandwidth aggregation, primary/backup communication paths, and traffic offload management.

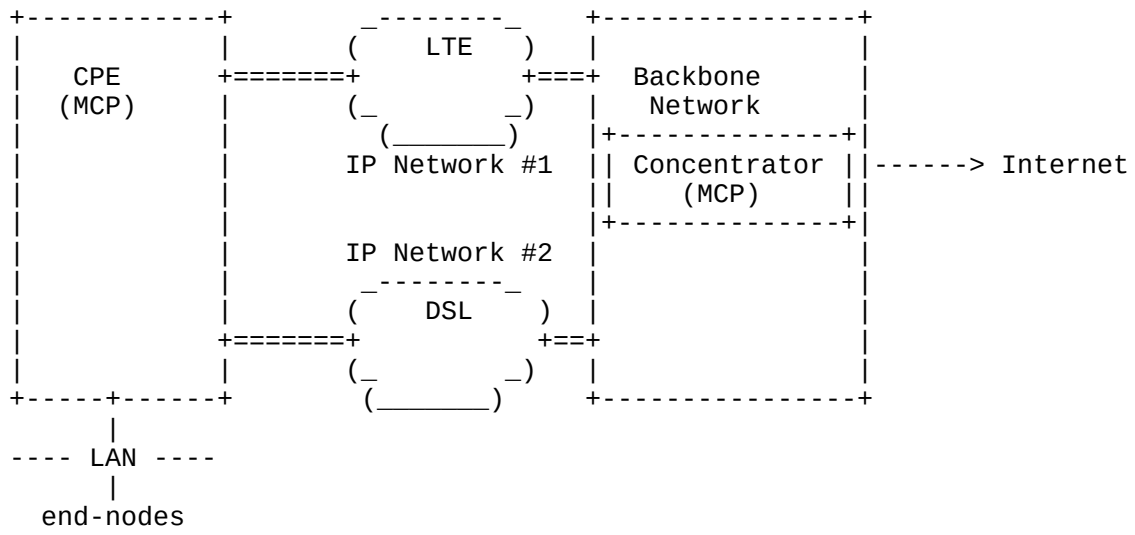


Figure 1: Network-Assisted MPTCP: Reference Architecture

Within this document, an MPTCP Conversion Point (MCP) 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 MCPs 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 MCP terminates the MPTCP connections received from its customer-facing interfaces and transforms these connections into legacy TCP connections [RFC0793] towards upstream servers. On the downlink path, the MCP turns the legacy server's TCP connection into MPTCP connections towards its customer-facing interfaces.

This document specifies two new Remote Authentication Dial-In User Service (RADIUS, [RFC2865]) attributes that carry the MCP 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 MCP(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]). Further Network-Assisted MPTCP deployment and operational considerations are discussed in [I-D.nam-mptcp-deployment-considerations].

This specification assumes an MCP 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 [RFC8044] for defining the new attributes.

2. MPTCP RADIUS Attributes

2.1. MPTCP-MCP-IPv4

Description

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

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

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

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

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

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

Type

TBA (see Section 6).

Length

6

Data Type

The attribute MPTCP-MCP-IPv4 is of type ip4addr (Section 3.3 of [RFC8044]).

Value

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

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

2.2. MPTCP-MCP-IPv6

Description

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

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

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

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

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

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

Type

TBA (see Section 6).

Length

18

Data Type

The attribute MPTCP-MCP-IPv6 is of type ip6addr (Section 3.9 of [RFC8044]).

Value

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

The MPTCP-MCP-IPv6 attribute MUST NOT include multicast and host loopback addresses [RFC6890]. Anycast addresses are allowed to be included in an MPTCP-MCP-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-MCP-IPv6 and MPTCP-MCP-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 MCP. This example assumes that the Network Access Server (NAS) embeds both RADIUS client and DHCPv6 server capabilities.

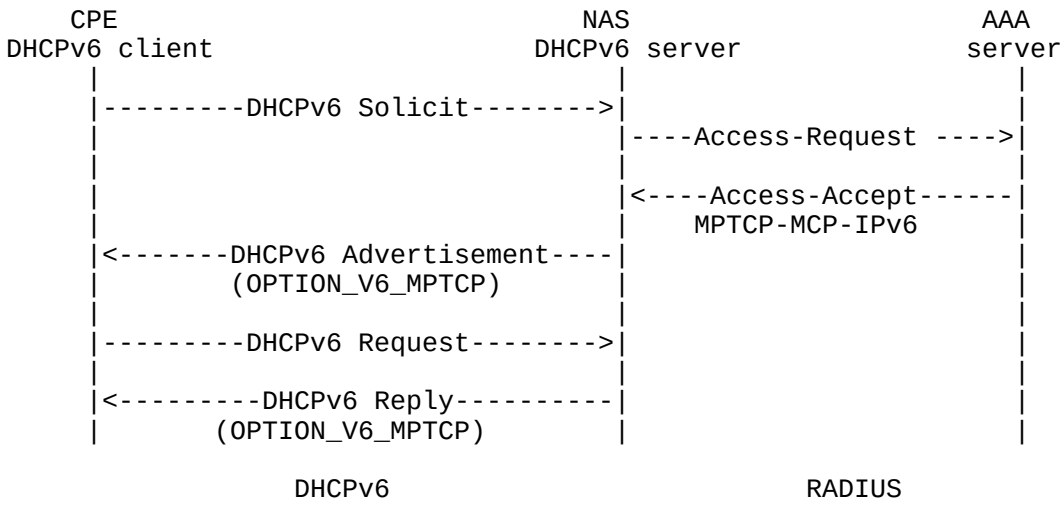


Figure 2: Sample Flow Example (1)

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 MCP reachability information (namely a list of IP addresses).

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

Upon change of the MCP assigned to a CPE, the RADIUS server sends a RADIUS CoA message [RFC5176] that carries the RADIUS MPTCP-MCP-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 MCP with the new one.

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

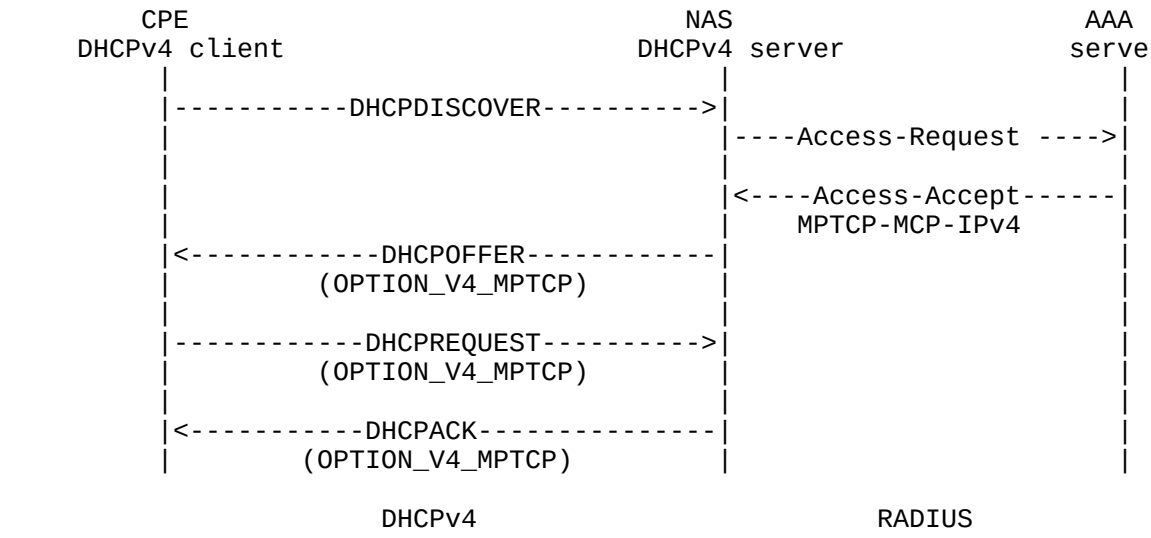


Figure 3: Sample Flow Example (2)

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 MCP is inserted in the path. Indeed, inserting an illegitimate MCP 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 an MCP 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-MCP-IPv4
0+	0+	0	0	0+	TBA MPTCP-MCP-IPv6

CoA-Request	CoA-ACK	CoA-NACK	#	Attribute
0+	0	0	TBA	MPTCP-MCP-IPv4
0+	0	0	TBA	MPTCP-MCP-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-MCP-IPv4 (TBA)

MPTCP-MCP-IPv6 (TBA)

7. Acknowledgements

Thanks to Alan DeKok for the comments.

8. References

8.1. 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, <https://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, <https://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, <https://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, <https://www.rfc-editor.org/info/rfc6890>.

[RFC8044] DeKok, A., "Data Types in RADIUS", RFC 8044, DOI 10.17487/RFC8044, January 2017, <https://www.rfc-editor.org/info/rfc8044>.

8.2. Informative References

[I-D.nam-mptcp-deployment-considerations] Boucadair, M., Jacquenet, C., Bonaventure, O., Henderickx, W., and R. Skog, "Network-Assisted MPTCP: Use Cases, Deployment Scenarios and Operational Considerations", draft-nam-mptcp-deployment-considerations-01 (work in progress), December 2016.

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

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <https://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, <https://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, <https://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, <https://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, <https://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, <https://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, <https://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, <https://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, <https://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
Category: Standards Track
<draft-dekok-saag-dhcp-keys-00.txt>
24 October 2016

DeKok, Alan
Network RADIUS SARL

DHCP Keys via 802.1X

Abstract

While RFC 3118 made provisions for securing DHCP, it made no provisions for creating or distributing authentication keys. This specification describes how in some cases, DHCP keys can be automatically derived from 802.1X authentication. The pros and cons of this approach are also discussed

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 May 24, 2017.

Copyright Notice

Table of Contents

1. Introduction

There has been increased interest in, and awareness of, securing basic networking protocols such as DHCP [RFC2131]. While provisions were made in [RFC3118] for securing the protocol via secret keys, there is little discussion on how the secret keys are created or managed. This specification addresses that issue, for the limited case of DHCP which occurs after 802.1X authentication.

1.1. Problem Statement

This document addresses the situation where a client machine connects to the network via 802.1X / EAP, and where keying material is derived as part of the EAP conversation. Once the client machine authenticated to the network, it requests an IP address via DHCP.

However, there is essentially no communication or interaction between the AAA server which authenticates the client machine, and the DHCP server which allocates IP addresses. This lack of communication means that it is possible to attack the systems independent. That is, the two systems do not work together to increase security.

1.2. Proposed Solution

Have the AAA server derive a shared secret key for signing DHCP packets. And then use that key to sign DHCP packets.

1.3. 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. Generating Keying Material

The algorithm used to generate the keying material is similar to that used for EAP methods, such as EAP-TLS ([RFC5216] Section 2.3), and TTLS ([RFC7542] Secton 8).

Upon successful conclusion of an EAP-TTLS negotiation, 128 octets of keying material are generated and exported for use in securing the data connection between client and access point. The first 4 octets of the keying material constitute the secret ID, the last 124 octets constitute the DHCP shared secret key.

The keying material is generated using the TLS PRF function [RFC5246], with inputs consisting of the TLS master secret, the ASCII-encoded constant string "dhcp keying material", the TLS client random, and the TLS server random. The constant string is not null-terminated.

```
Keying Material = PRF-128(SecurityParameters.master_secret, "dhcp
          keying material", SecurityParameters.client_random +
          SecurityParameters.server_random)

Secret ID = Keying Material [0..3]

DHCP Shared secret key = Keying Material [4..127]
```

We perhaps don’t want to use the keying material directly in the Secret ID. Instead, maybe use the last 4 octets of the keying material? Which should give less information than the first 4 octets. Or, derive the secret ID from a different PRF? Or set it to a fixed ID, which indicates that the client is using this method for signing packets?

The lifetime of the key is the lifetime of the underlying authentication session. If the client machine re-authenticates, a new DHCP shared secret key is derived.

The lifetime of the key MUST be no longer than the lifetime of the underlying authentication session. That is, once the authentication session has expired, the client MUST discard the key along with the corresponding secret ID.

We chould also do something useful with the RDM / Reply Detection fields.

2.1. Implementation Considerations

There has historically been little communication between DHCP servers and AAA servers. As such, there is no clear way for the AAA server to share the derived key with the DHCP server. We leave that problem for implementors to solve.

The DHCP server could receive a packet, and request the corresponding key from the AAA server. Or, it may hand the packet to the AAA server for verification and/or signing. Or, it may retrive the key from a secure co-located database. Or, the AAA server may pro-actively inform the DHCP server that a key exists, along with the keys value.

All of these scenarios are possible, and it is difficult to recommend any one in particular. We can, however make general security recommendations.

The keys are highly secret information. As such, any exchange of keys MUST be done in a secure manner. Keys MUST NOT be visible to any entity other than the DHCP server and AAA server. If keys are stored in a database, they MUST be encrypted with an encryption key known only to the DHCP server and AAA server.

2.2. What does the Signature mean?

While it is nice to sign a DHCP packet for security, it is not at all clear what is meant by the signature. The minimum we can say is that the signature means that the DHCP server has communicated with the AAA server, and obtained a copy of the key.

There is no way to know if the DHCP server is the correct one, or malicious, or under the control of an attacker. Though if that is true, there is no need for the DHCP server to obtain the key. It can just hand out any addresses it wants. And if you can compromise a DHCP server on the network, or run a rogue DHCP server, having signed packets is probably the least of your worries.

We suggest that the signature means (in the expected case), that the DHCP server is known to the AAA server, and is likely known to the network administrators, and is likely the correct DHCP server for the client to communicate with.

2.3. Open Questions

Q: Does this add security?

A: Perhaps. A larger discussion and analysis is necessary. It does

not appear to reduce security. i.e. forging the key is difficult to impossible. The most that can be done is to disable this mechanism, which reduces DHCP to it's current level of security.

Q: How does the AAA server indicate to the DHCP client that it has this capability?

A: There is no way for it to do so. The DHCP client can just sign packets speculatively.

Q: How does the DHCP server know to use the key?

A: RFC3118 has provisions for the client signalling that it has a key.

Q: What does a DHCP server do when it receives a message from the client indicating that the client has a key, but the DHCP server does not have the key?

A: RFC 3118 is silent on this topic. The likeliest response is for the DHCP server to ignore the signature.

Q: How many networks are technically capable of using 802.1X?

A: It's 2016, pretty much all of them.

Q: How many networks are administratively capable of using 802.1X?

A: Some. Not so much for home networks, WiFi hot spots, etc. Most enterprises, telcos (3G), and WiFI systems with Hotspot 2.0 should be capable.

Q: What is the cost of implementing this?

A: Unknown. Some modifications to AAA / DHCP servers. And communication between them via some method to be determined.

3. Security Considerations

This specification is concerned entirely with security. As such, additional security discussion is not necessary.

4. IANA Considerations

There are no IANA considerations for this document.

5. References

5.1. Normative References

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

[RFC2131]
Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

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

[RFC3118]
Droms R. (Ed) and Arbaugh W. (Ed), "Authentication for DHCP Messages", RFC 3118, June 2001.

5.2. Informative References

[RFC5216]
Simon, D., Aboba, B., and Hurst, R, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5246]
Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

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

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

Acknowledgments

None at this time.

Authors' Addresses

Alan DeKok
Network RADIUS SARL
100 Centrepointe Drive
Suite 200

Ottawa, ON
K2G 6B1
Canada

Email: aland@networkradius.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: May 4, 2017

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

October 31, 2016

LoRaWAN Authentication in RADIUS
draft-garcia-radext-radius-lorawan-02

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.

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.

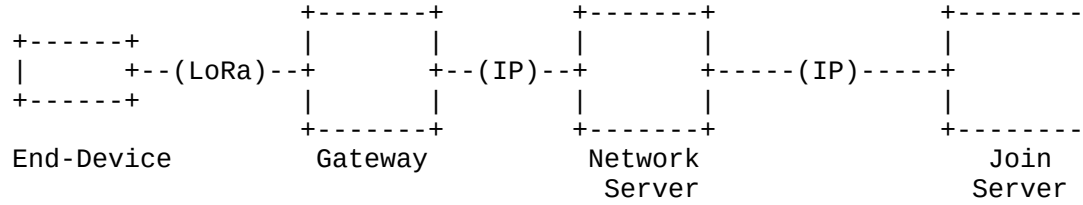


Figure 1: LoRAWAN Architecture

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

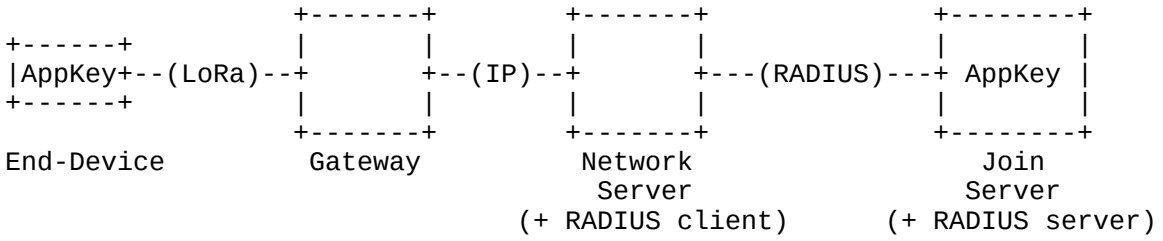


Figure 2: LoRAWAN Architecture with AAA and RADIUS authentication. End-Device and RADIUS server have a shared secret - the AppKey, which is used to derive the session keys (NwkSKey and AppSKey).

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.

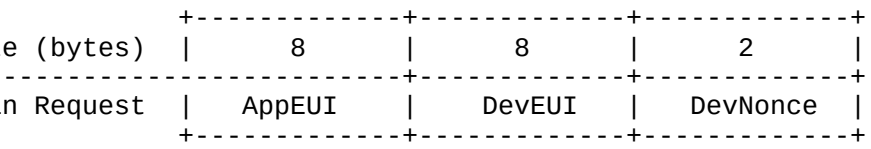


Figure 3: Join Request Message

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).

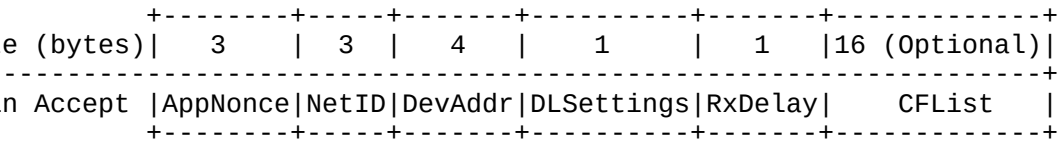


Figure 4: Join Accept Message

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.
- 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.

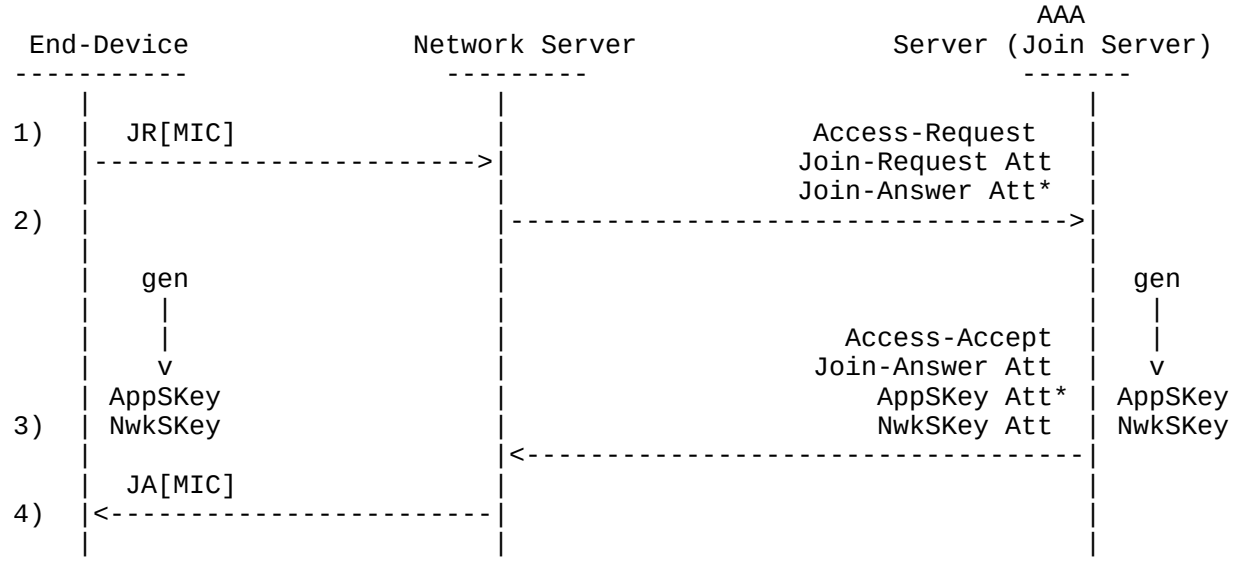
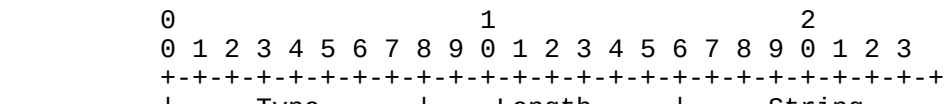


Figure 5: Protocol

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

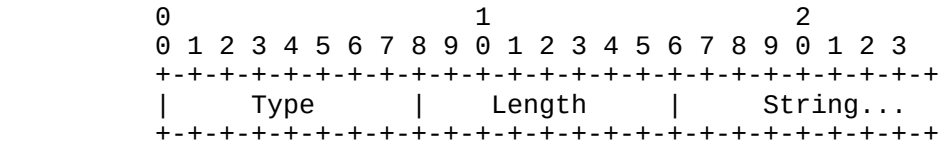
String

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

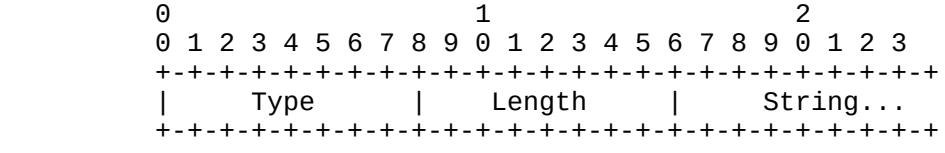
String

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+

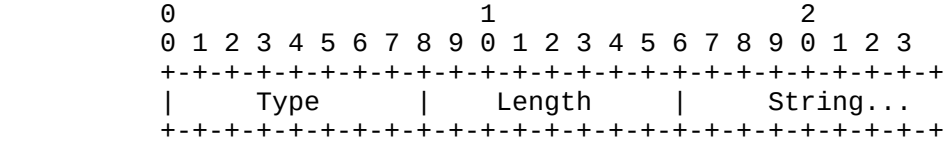
String

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

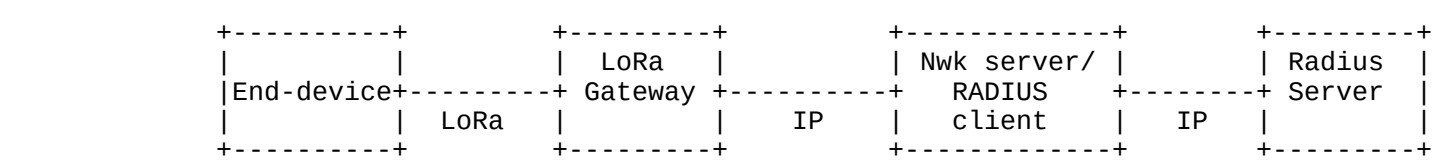
The proof of concept is implemented using the Go programming language, that is well suited for the development of web servers or a network servers as in this case.

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
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.

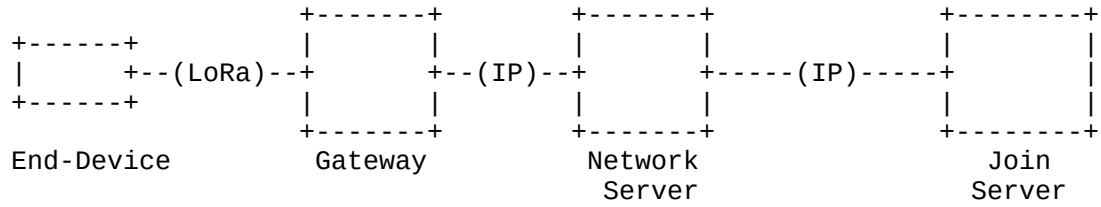


Figure 1: LoRAWAN Architecture

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

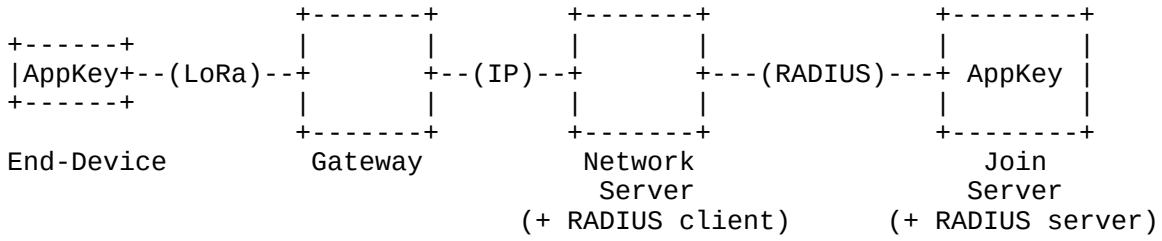


Figure 2: LoRAWAN Architecture with AAA and RADIUS authentication. End-Device and RADIUS server have a shared secret - the AppKey, which is used to derive the session keys (NwkSKey and AppSKey).

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.

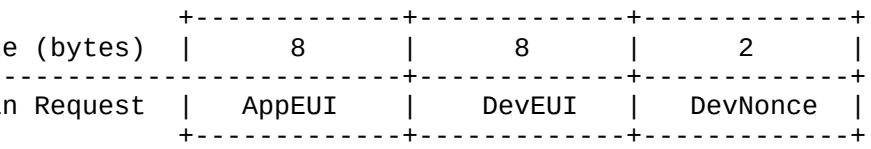


Figure 3: Join Request Message

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).

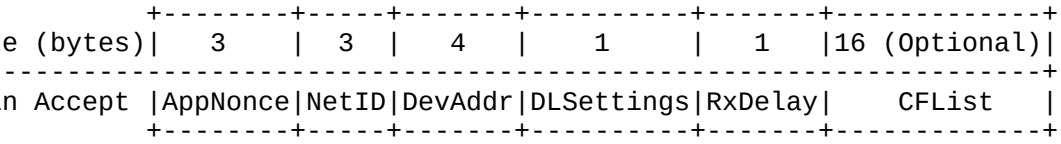


Figure 4: Join Accept Message

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.

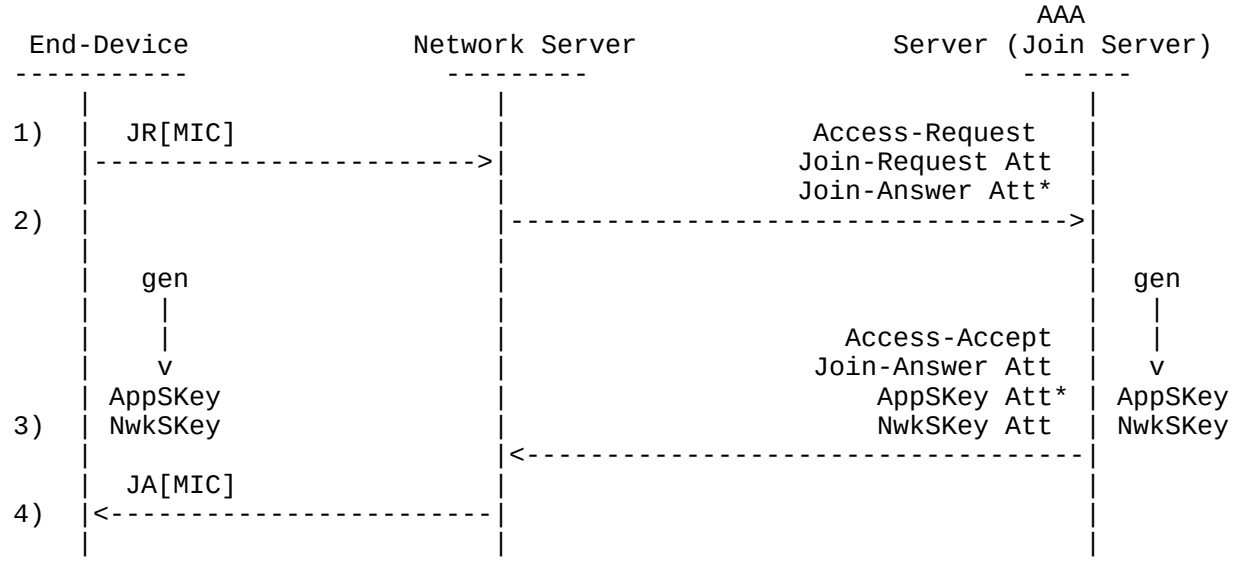
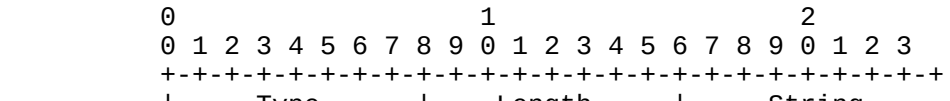


Figure 5: Protocol

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

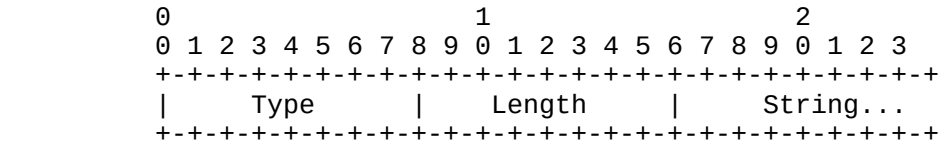
String

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

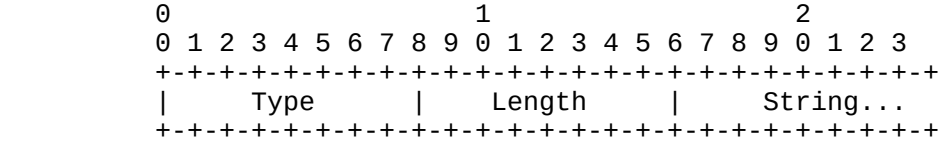
String

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+

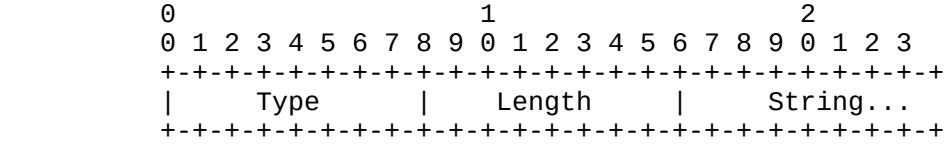
String

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

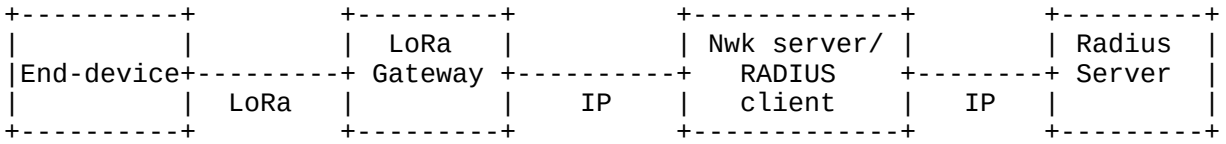
The proof of concept is implemented using the Go programming language, that is well suited for the development of web servers or a network servers as in this case.

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

.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/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 July 22, 2019.

Table of Contents

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 ommission 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 obsfuscate 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 packet prior to sending that packet to the final CoA server (i.e. NAS). This step is necessary due to the 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 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 authentiction.

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-

Identifer, 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 Opertor-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-Identifer 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

Network Working Group
INTERNET-DRAFT
Updates: 2865,3162,6158,6572
Category: Standards Track
<draft-ietf-radext-datatypes-08.txt>
18 October 2016

DeKok, Alan
FreeRADIUS

Data Types in the Remote Authentication Dial-In User Service Protocol (RADIUS)

Abstract

RADIUS specifications have used data types for two decades without defining them as managed entities. During this time, RADIUS implementations have named the data types, and have used them in attribute definitions. This document updates the specifications to better follow established practice. We do this by naming the data types defined in RFC 6158, which have been used since at least the publication of RFC 2865. We provide an IANA registry for the data types, and update the RADIUS Attribute Type registry to include a "Data Type" field for each attribute. Finally, we recommend that authors of RADIUS specifications use these types in preference to existing practice. This document updates RFC 2865, 3162, 6158, and 6572.

Status of this Memo

Table of Contents

1. Introduction

RADIUS specifications have historically defined attributes in terms of name, value, and data type. Of these three pieces of information, the name is recorded by IANA in the RADIUS Attribute Type registry, but not otherwise managed or restricted, as discussed in [RFC6929] Section 2.7.1. The value is managed by IANA, and recorded in that registry. The data type is not managed or recorded in the RADIUS Attribute Type registry. Experience has shown that there is a need to create well known data types, and have them managed by IANA.

This document defines an IANA RADIUS Data Type registry, and updates the RADIUS Attribute Type registry to use those newly defined data types. It recommends how both specifications and implementations should use the data types. It extends the RADIUS Attribute Type registry to have a data type for each assigned attribute.

In this section, we review the use of data types in specifications and implementations. Whe highlight ambiguities and inconsistencies. The rest of this document is devoted to resolving those problems.

1.1. Specification Problems with Data Types

When attributes are defined in the specifications, the terms "Value" and "String" are used to refer to the contents of an attribute. However, these names are used recursively and inconsistently. We suggest that defining a field to recursively contain itself is problematic.

A number of data type names and definitions are given in [RFC2865] Section 5, at the bottom of page 25. These data types are named and clearly defined. However, this practice was not continued in later specifications.

Specifically, [RFC2865] defines attributes of data type "address" to carry IPv4 addresses. Despite this definition, [RFC3162] defines attributes of data type "Address" to carry IPv6 addresses. We suggest that the use of the word "address" to refer to disparate data types is problematic.

Other failures are that [RFC3162] does not give a data type name and definition for the data types IPv6 address, Interface-Id, or IPv6 prefix. [RFC2869] defines Event-Timestamp to carry a time, but does not re-use the "time" data type defined in [RFC2865]. Instead, it just repeats the "time" definition. [RFC6572] defines multiple attributes which carry IPv4 prefixes. However, an "IPv4 prefix" data type is not named, defined as a data type, or called out as an addition to RADIUS. Further, [RFC6572] does not follow the

recommendations of [RFC6158], and does not explain why it fails to follow those recommendations.

These ambiguities and inconsistencies need to be resolved.

1.2. Implementation Problems with Data Types

RADIUS implementations often use "dictionaries" to map attribute names to type values, and to define data types for each attribute. The data types in the dictionaries are defined by each implementation, but correspond to the "ad hoc" data types used in the specifications.

In effect, implementations have seen the need for well-defined data types, and have created them. It is time for RADIUS specifications to follow this practice.

1.3. No Mandated Changes

This document mandates no changes to any RADIUS implementation, past, present, or future. It instead documents existing practice, in order to simplify the process of writing RADIUS specifications, to clarify the interpretation of RADIUS standards, and to improve the communication between specification authors and IANA.

This document suggests that implementations SHOULD use the data types defined here, in preference to any "ad hoc" data types currently in use. This suggestion should have minimal effect on implementations, as most "ad hoc" data types are compatible with the ones defined here. Any difference will typically be limited to the name of the data type.

This document updates [RFC6158] to permit the data types defined in the "Data Type registry" as "basic data types", as per Section 2.1 of that document. The recommendations of [RFC6158] are otherwise unchanged.

1.4. 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. Use of Data Types

The Data Types can be used in two places: specifications, and implementations. This section discusses both uses, and gives guidance on using the data types.

2.1. Specification Use of Data Types

In this section, we give recommendations for how specifications should be written using data types. We first describe how attribute field names can be consistently named. We then describe how attribute definitions should use the data types, and deprecate the use of "ASCII art" for attribute definitions. We suggest a format for new attribute definitions. This format includes recommended fields, and suggestions for how those fields should be described.

Finally, we make recommendations for how new data types should be defined.

2.1.1. Field Names for Attribute Values

Previous specifications used inconsistent and conflicting names for the contents of RADIUS attributes. For example, the term "Value" is used in [RFC2865] Section 5 to define a field which carries the contents of attribute. It is then used in later sections as the sub-field of attribute contents. The result is that the field is defined as recursively containing itself. Similarly, "String" is used both as a data type, and as a sub-field of other data types.

We correct this ambiguity by using context-specific names for various fields of attributes and data types. It then becomes clear that, for example, that a field called "VSA-Data" must contain different data than a field called "EVS-Data". Each new name is defined where it is used.

We also define the following term:

Attr-Data

: The "Value" field of an Attribute as defined in [RFC2865] Section 5. The contents of this field MUST be of a valid data type as defined in the RADIUS Data Type registry.

We consistently use "Attr-Data" to refer to the contents of an attribute, instead of the more ambiguous name "Value". It is RECOMMENDED that new specifications follow this practice.

We consistently use "Value" to refer to the contents of a data type,

where that data type is simple. For example, an "integer" can have a "Value". In contrast, a Vendor-Specific attribute carries complex information, and thus cannot have a "Value".

For data types which carry complex information, we name the fields based on the data type. For example, a Vendor-Specific attribute is defined to carry a "vsa" data type, and the contents of that data type are described herein as "VSA-Data".

These terms are used in preference to the term "String", which was previously used in ambiguous ways. It is RECOMMENDED that future specifications use type-specific names, and the same naming scheme for new types. This use will maintain consistent definitions, and help to avoid ambiguities.

2.1.2. Attribute Definitions using Data Types

New RADIUS specifications MUST define attributes using data types from the RADIUS Data Type registry. The specification may, of course, define a new data type update the "Data Types" registry, and use the new data type, all in the same document. The guidelines given in [RFC6929] MUST be followed when defining a new data type.

Attributes can usually be completely described via the Attribute Type value, name, and data type. The use of "ASCII art" is then limited only to the definition of new data types, and for complex data types.

Use of the new extended attributes [RFC6929] makes ASCII art even more problematic. An attribute can be allocated from any of the extended spaces, with more than one option for attribute header format. This allocation decision is made after the specification has been accepted for publication. As the allocation affects the format of the attribute header, it is essentially impossible to create the correct ASCII art prior to final publication. Allocation from the different spaces also changes the value of the Length field, also making it difficult to define it correctly prior to final publication of the document.

It is therefore RECOMMENDED that "ASCII art" diagrams not be used for new RADIUS attribute specifications.

2.1.3. Format of Attribute Definitions

When defining a new attribute, the following fields SHOULD be given:

Description

A description of the meaning and interpretation of the attribute.

Type

The Attribute Type value, given in the "dotted number" notation from [RFC6929]. Specifications can often leave this as "TBD", and request that IANA fill in the allocated values.

Length

A description of the length of the attribute. For attributes of variable length, a maximum length SHOULD be given. Since the Length may depend on the Type, the definition of Length may be affected by IANA allocations.

Data Type

One of the named data types from the RADIUS Data Type registry.

Value

A description of any attribute-specific limitations on the values carried by the specified data type. If there are no attribute-specific limitations, then the description of this field can be omitted, so long as the Description field is sufficiently explanatory.

Where the values are limited to a subset of the possible range, valid range(s) MUST be defined.

For attributes of data type "enum", a list of enumerated values and names MUST be given, as with [RFC2865] Section 5.6.

Using a consistent format for attribute definitions helps to make the definitions clearer.

2.1.4. Defining a New Data Type

When a specification needs to define a new data type, it SHOULD follow the format used by the definitions in Section 3 of this document. The text at the start of the data type definition MUST describe the data type, including the expected use, and why a new data type is required. That text SHOULD include limits on expected values, and why those limits exist. The field’s "Name", "Value", "Length", and "Format", MUST be given, along with values.

The "Name" field SHOULD be a single name, all lower-case.

Contractions such as "ipv4addr" are RECOMMENDED where they add clarity.

We note that the use of "Value" in the RADIUS Data Type registry can be confusing. That name is also used in attribute definitions, but with a different meaning. We trust that the meaning here is clear from the context.

The "Value" field SHOULD be given as to be determined or "TBD" in specifications. That number is assigned by IANA.

The "Format" field SHOULD be defined with "ASCII art" in order to have a precise definition. Machine-readable formats are also RECOMMENDED.

The definition of a new data type should be done only when absolutely necessary. We do not expect a need for a large number of new data types. When defining a new data type, the guideliness of [RFC6929] with respect to data types MUST be followed.

It is RECOMMENDED that vendors not define "vendor specific" data types. As discussed in [RFC6929], those data types are rarely necessary, and can cause interoperability problems.

Any new data type MUST have unique name in the RADIUS Data Type registry. The number of the data type will be assigned by IANA.

2.2. Implementation Use of Data Types

Implementations not supporting a particular data type MUST treat attributes of that data type as being of data type "string", as defined in Section 3.6. It is RECOMMENDED that such attributes be treated as "invalid attributes", as defined in [RFC6929] Section 2.8.

Where the contents of a data type do not match the definition, implementations MUST treat the the enclosing attribute as being an "invalid attribute". This requirement includes, but is not limited to, the following situations:

* Attributes with values outside of the allowed range(s) for the data type, e.g. as given in the data types "integer", "ipv4addr", "ipv6addr", "ipv4prefix", "ipv6prefix", or "enum".

* "text" attributes where the contents do not match the required format,

* Attributes where the length is shorter or longer than the allowed length(s) for the given data type,

The requirements for "reserved" fields are more difficult to quantify. Implementations SHOULD be able to receive and process attributes where "reserved" fields are non-zero. We do not, however, define any "correct" processing of such attributes. Instead, specifications which define new meaning for "reserved" fields SHOULD describe how the new meaning is compatible with older implementations. We expect that such descriptions are derived from practice. Implementations MUST set "reserved" fields to zero when creating attributes.

3. Data Type Definitions

This section defines the new data types. For each data type, it gives a definition, a name, a number, a length, and an encoding format. Where relevant, it describes subfields contained within the data type. These definitions have no impact on existing RADIUS implementations. There is no requirement that implementations use these names.

Where possible, the name of each data type has been taken from previous specifications. In some cases, a different name has been chosen. The change of name is sometimes required to avoid ambiguity (i.e. "address" versus "Address"). Otherwise, the new name has been chosen to be compatible with [RFC2865], or with use in common implementations. In some cases, new names are chosen to clarify the interpretation of the data type.

The numbers assigned herein for the data types have no meaning other than to permit them to be tracked by IANA. As RADIUS does not encode information about data types in a packet, the numbers assigned to a data type will never occur in a packet. It is RECOMMENDED that new implementations use the names defined in this document, in order to avoid confusion. Existing implementations may choose to use the names defined here, but that is not required.

The encoding of each data type is taken from previous specifications. The fields are transmitted from left to right.

Where the data types have inter-dependencies, the simplest data type is given first, and dependent ones are given later.

We do not create specific data types for the "tagged" attributes defined in [RFC2868]. That specification defines the "tagged" attributes as being backwards compatible with pre-existing data types. In addition, [RFC6158] Section 2.1 says that "tagged" attributes should not be used. There is therefore no benefit to defining additional data types for these attributes. We trust that implementors will be aware that tagged attributes must be treated differently from non-tagged attributes of the same data type.

Similarly, we do not create data types for some attributes having complex structure, such as CHAP-Password, ARAP-Features, or Location-Information. We need to strike a balance between correcting earlier mistakes, and making this document more complex. In some cases, it is better to treat complex attributes as being of type "string", even though they need to be interpreted by RADIUS implementations. The guidelines given in Section 6.3 of [RFC6929] were used to make this determination.

3.1. integer

The "integer" data type encodes a 32-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range. Attributes with Values outside of the allowed ranges SHOULD be treated as "invalid attributes".

Name

integer

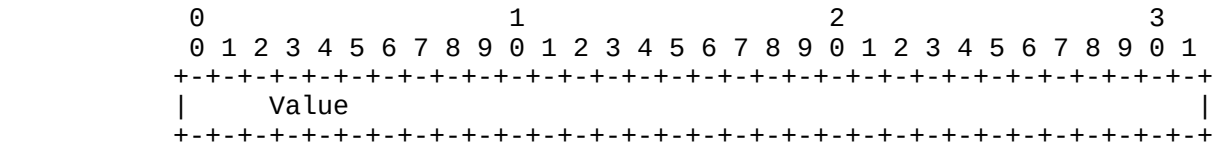
Value

1

Length

Four octets

Format



3.2. enum

The "enum" data type encodes a 32-bit unsigned integer in network byte order. It differs from the "integer" data type only in that it is used to define enumerated types, such as Service-Type (Section 5.6 of [RFC2865]). Specifications MUST define a valid set of enumerated values, along with a unique name for each value. Attributes with Values outside of the allowed enumerations SHOULD be treated as "invalid attributes".

Name

enum

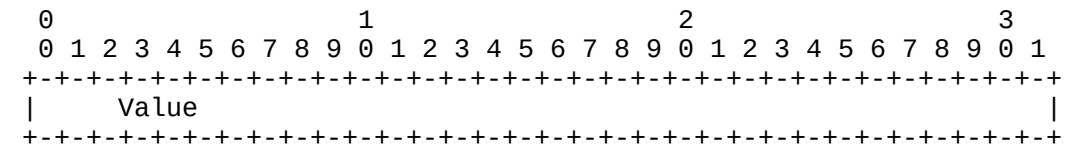
Value

2

Length

Four octets

Format



3.3. time

The "time" data type encodes time as a 32-bit unsigned value in network byte order and in seconds since 00:00:00 UTC, January 1, 1970. We note that dates before the year 2016 are likely to indicate configuration errors, or lack of access to the correct time.

Note that the "time" attribute is defined to be unsigned, which means it is not subject to a signed integer overflow in the year 2038.

Name

time

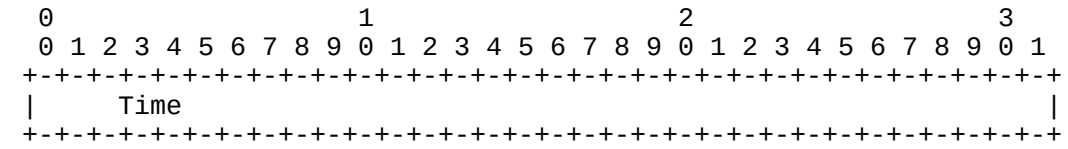
Value

3

Length

Four octets

Format



3.4. text

The "text" data type encodes UTF-8 text [RFC3629]. The maximum length of the text is given by the encapsulating attribute. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid

range(s). Attributes with length outside of the allowed values SHOULD be treated as "invalid attributes".

Attributes of type "text" which are allocated in the standard space (Section 1.2 of [RFC6929]) are limited to no more than 253 octets of data. Attributes of type "text" which are allocated in the extended space can be longer. In both cases, these limits are reduced when the data is encapsulated inside of an another attribute.

Where the text is intended to carry data in a particular format, (e.g. Framed-Route), the format MUST be given. The specification SHOULD describe the format in a machine-readable way, such as via Augmented Backus-Naur Form (ABNF) [RFC5234]. Attributes with values not matching the defined format SHOULD be treated as "invalid attributes".

Note that the "text" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Texts of length zero (0) MUST NOT be sent; omit the entire attribute instead.

Name

text

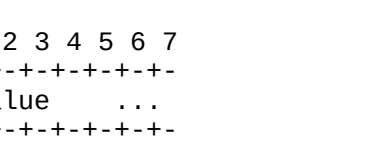
Value

4

Length

One or more octets.

Format



3.5. string

The "string" data type encodes binary data, as a sequence of undistinguished octets. Where the range of lengths for a particular attribute is limited to a sub-set of possible lengths, specifications MUST define the valid range(s). Attributes with length outside of

the allowed values SHOULD be treated as "invalid attributes".

Attributes of type "string" which are allocated in the standard space (Section 1.2 of [RFC6929]) are limited to no more than 253 octets of data. Attributes of type "string" which are allocated in the extended space can be longer. In both cases, these limits are reduced when the data is encapsulated inside of an another attribute.

Note that the "string" data type does not terminate with a NUL octet (hex 00). The Attribute has a Length field and does not use a terminator. Strings of length zero (0) MUST NOT be sent; omit the entire attribute instead. a Where there is a need to encapsulate complex data structures, and TLVs cannot be used, the "string" data type MUST be used. This requirement includes encapsulation of data structures defined outside of RADIUS, which are opaque to the RADIUS infrastucture. It also includes encapsulation of some data structures which are not opaque to RADIUS, such as the contents of CHAP-Password.

There is little reason to define a new RADIUS data type for only one attribute. However, where the complex data type cannot be represented as TLVs, and is expected to be used in many attributes, a new data type SHOULD be defined.

These requirements are stronger than [RFC6158], which makes the above encapsulation a "SHOULD". This document defines data types for use in RADIUS, so there are few reasons to avoid using them.

Name

string

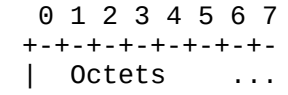
Value

5

Length

One or more octets.

Format



3.6. concat

The "concat" data type permits the transport of more than 253 octets of data in a "standard space" [RFC6929] attribute. It is otherwise identical to the "string" data type.

If multiple attributes of this data type are contained in a packet, all attributes of the same type code MUST be in order and they MUST be consecutive attributes in the packet.

The amount of data transported in a "concat" data type can be no more than the RADIUS packet size. In practice, the requirement to transport multiple attributes means that the limit may be substantially smaller than one RADIUS packet. As a rough guide, is RECOMMENDED that this data type transport no more than 2048 octets of data.

The "concat" data type MAY be used for "standard space" attributes. It MUST NOT be used for attributes in the "short extended space" or the "long extended space". It MUST NOT be used in any field or subfields of the following data types: "tlv", "vsa", "extended", "long-extended", or "evs".

Name

concat

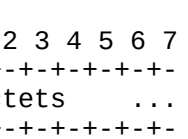
Value

6

Length

One or more octets.

Format



3.7. ifid

The "ifid" data type encodes an Interface-Id as an 8 octet IPv6 Interface Identifier network byte order.

Name

ifid

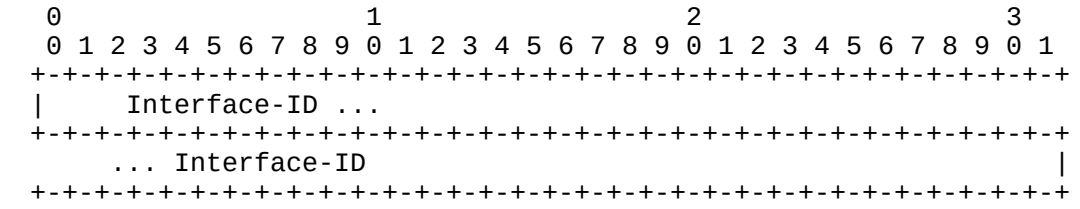
Value

7

Length

Eight octets

Format



3.8. ipv4addr

The "ipv4addr" data type encodes an IPv4 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

ipv4addr

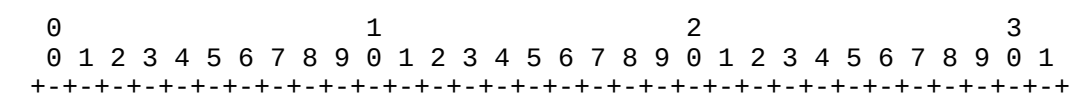
Value

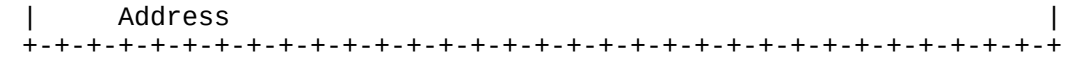
8

Length

Four octets

Format





3.9. ipv6addr

The "ipv6addr" data type encodes an IPv6 address in network byte order. Where the range of address for a particular attribute is limited to a sub-set of possible addresses, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

ipv6addr

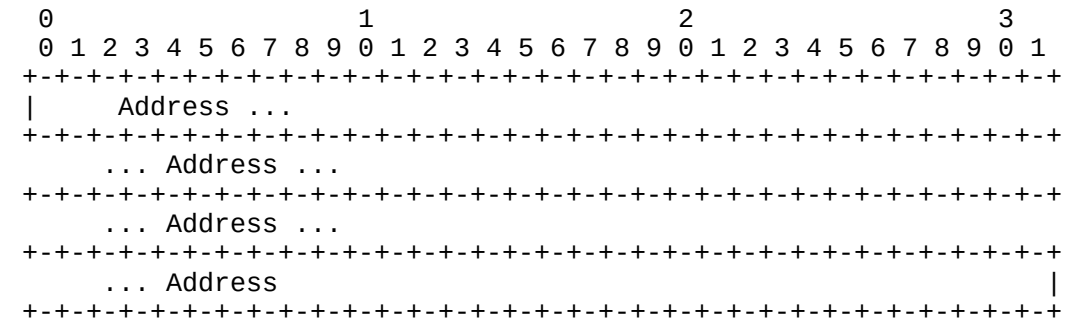
Value

9

Length

Sixteen octets

Format



3.10. ipv6prefix

The "ipv6prefix" data type encodes an IPv6 prefix, using both a prefix length and an IPv6 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 128 MUST be treated as "invalid attributes".

Name

ipv6prefix

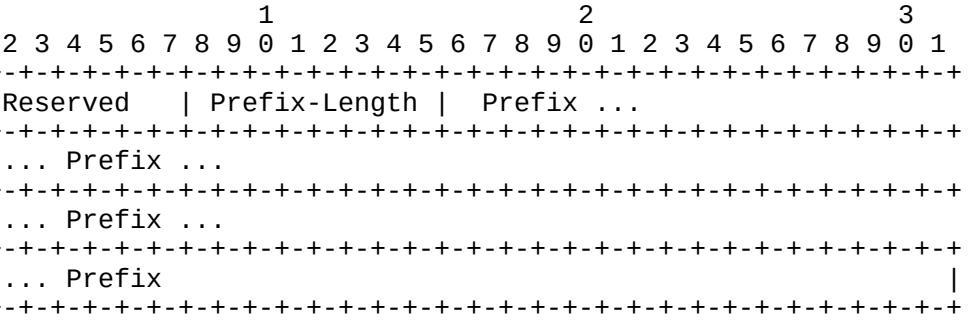
Value

10

Length

At least two, and no more than eighteen octets.

Format



Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero. This field is one octet in length.

Prefix-Length

The length of the prefix, in bits. At least 0 and no larger than 128. This field is one octet in length.

Prefix

The Prefix field is up to 16 octets in length. Bits outside of the Prefix-Length, if included, MUST be zero.

The Prefix field SHOULD NOT contain more octets than necessary to encode the Prefix.

3.11. ipv4prefix

The "ipv4prefix" data type encodes an IPv4 prefix, using both a prefix length and an IPv4 address in network byte order. Where the range of prefixes for a particular attribute is limited to a sub-set of possible prefixes, specifications MUST define the valid range(s). Attributes with Addresses outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Attributes with a Prefix-Length field having value greater than 32 MUST be treated as "invalid attributes".

Name

ipv4prefix

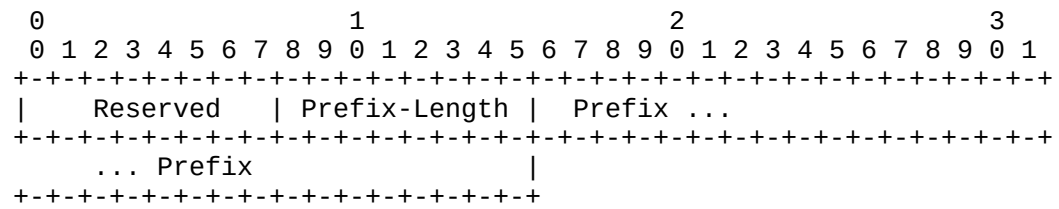
Value

11

Length

six octets

Format



Subfields

Reserved

This field, which is reserved and MUST be present, is always set to zero. This field is one octet in length.

Note that this definition differs from that given in [RFC6572]. See Prefix-Length, below, for an explanation.

Prefix-Length

The length of the prefix, in bits. The values MUST be no larger than 32. This field is one octet in length.

Note that this definition differs from that given in [RFC6572].

As compared to [RFC6572], the Prefix-Length field has increased in size by two bits, both of which must be zero. The Reserved field has decreased in size by two bits. The result is that both fields are aligned on octet boundaries, which removes the need for bit masking of the fields.

Since [RFC6572] required the Reserved field to be zero, the definition here is compatible with the definition in the original specification.

Prefix

The Prefix field is 4 octets in length. Bits outside of the Prefix-Length MUST be zero. Unlike the "ipv6prefix" data type, this field is fixed length. If the address is all zeros (i.e. "0.0.0.0", then the Prefix-Length MUST be set to 32.

3.12. integer64

The "integer64" data type encodes a 64-bit unsigned integer in network byte order. Where the range of values for a particular attribute is limited to a sub-set of the values, specifications MUST define the valid range(s). Attributes with Values outside of the allowed range(s) SHOULD be treated as "invalid attributes".

Name

integer64

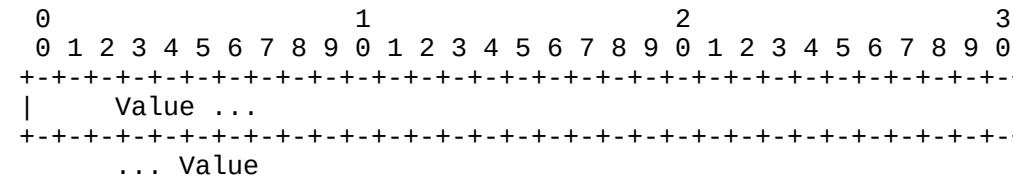
Value

12

Length

Eight octets

Format



```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.13. tlv

The "tlv" data type encodes a type-length-value, as defined in [RFC6929] Section 2.3.

Name

tlv

Value

13

Length

Three or more octets

Format

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   TLV-Type    |  TLV-Length   |     TLV-Data ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Subfields

TLV-Type

This field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Values of 254-255 are "Reserved" for use by future extensions to RADIUS. The value 26 has no special meaning, and MUST NOT be treated as a Vendor Specific attribute.

The TLV-Type is meaningful only within the context defined by "Type" fields of the encapsulating Attributes, using the dotted-number notation introduced in [RFC6929].

A RADIUS server MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS client MAY ignore Attributes with an unknown "TLV-Type".

A RADIUS proxy SHOULD forward Attributes with an unknown "TLV-Type" verbatim.

TLV-Length

The TLV-Length field is one octet, and indicates the length of this TLV including the TLV-Type, TLV-Length and TLV-Value fields. It MUST have a value between 3 and 255. If a client or server receives a TLV with an invalid TLV-Length, then the attribute which encapsulates that TLV MUST be considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

TLVs having TLV-Length of two (2) MUST NOT be sent; omit the entire TLV instead.

TLV-Data

The TLV-Data field is one or more octets and contains information specific to the Attribute. The format and length of the TLV-Data field is determined by the TLV-Type and TLV-Length fields.

The TLV-Data field MUST contain only known RADIUS data types. The TLV-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

3.14. vsa

The "vsa" data type encodes Vendor-Specific data, as given in [RFC2865] Section 5.26. It is used only in the Attr-Data field of a Vendor-Specific Attribute. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "vsa" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

vsa

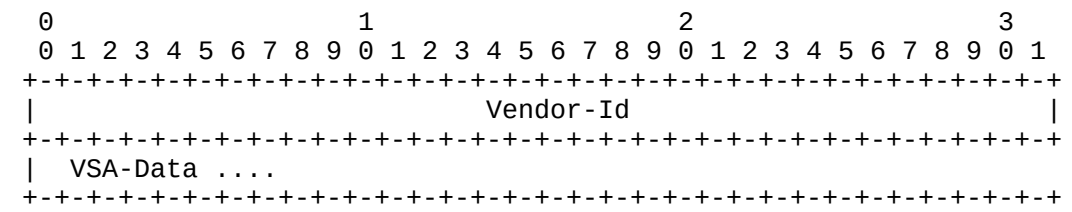
Value

14

Length

Five or more octets

Format



Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

VSA-Data

The VSA-Data field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets.

The codification of the range of allowed usage of this field is outside the scope of this specification.

The "vsa" data type SHOULD contain as a sequence of "tlv" data types. The interpretation of the TLV-Type and TLV-Data fields are dependent on the vendor's definition of that attribute.

The "vsa" data type MUST be used as contents of the Attr-Data field of the Vendor-Specific attribute. The "vsa" data type MUST NOT appear in the contents of any other data type.

3.15. extended

The "extended" data type encodes the "Extended Type" format, as given in [RFC6929] Section 2.1. It is used only in the Attr-Data field of an Attribute allocated from the "standard space". It MUST NOT appear in the contents of any other data type.

Name

extended

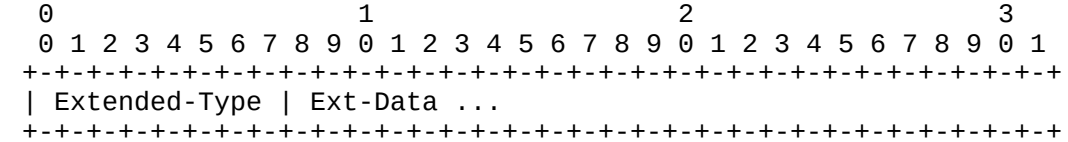
Value

15

Length

Two or more octets

Format



Subfields

Extended-Type

The Extended-Type field is one octet. Up-to-date values of this field are specified according to the policies and rules described in [RFC6929] Section 10. Unlike the Type field defined in [RFC2865] Section 5, no values are allocated for experimental or implementation-specific use. Values 241-255 are reserved and MUST NOT be used.

The Extended-Type is meaningful only within a context defined by the Type field. That is, this field may be thought of as defining a new type space of the form "Type.Extended-Type". See [RFC6929] Section 2.5 for additional discussion.

A RADIUS server MAY ignore Attributes with an unknown "Type.Extended-Type".

A RADIUS client MAY ignore Attributes with an unknown "Type.Extended-Type".

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the

Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

3.16. long-extended

The "long-extended" data type encodes the "Long Extended Type" format, as given in [RFC6929] Section 2.2. It is used only in the Attr-Data field of an Attribute. It MUST NOT appear in the contents of any other data type.

Name

long-extended

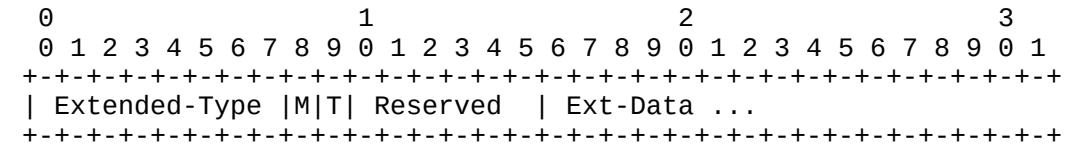
Value

16

Length

Three or more octets

Format



Subfields

Extended-Type

This field is identical to the Extended-Type field defined above in Section 3.15.

M (More)

The More field is one (1) bit in length, and indicates whether or not the current attribute contains "more" than 251 octets of data. The More field MUST be clear (0) if the Length field has value less than 255. The More field MAY be set (1) if the Length field has value of 255.

If the More field is set (1), it indicates that the Ext-Data field has been fragmented across multiple RADIUS attributes.

When the More field is set (1), the attribute MUST have a Length field of value 255; there MUST be an attribute following this one; and the next attribute MUST have both the same Type and Extended Type. That is, multiple fragments of the same value MUST be in order and MUST be consecutive attributes in the packet, and the last attribute in a packet MUST NOT have the More field set (1).

That is, a packet containing a fragmented attribute needs to contain all fragments of the attribute, and those fragments need to be contiguous in the packet. RADIUS does not support inter-packet fragmentation, which means that fragmenting an attribute across multiple packets is impossible.

If a client or server receives an attribute fragment with the "More" field set (1), but for which no subsequent fragment can be found, then the fragmented attribute is considered to be an "invalid attribute", and handled as per [RFC6929] Section 2.8.

T (Truncation)

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

Please see [RFC7499] for further discussion of the uses of this flag.

Reserved

This field is 6 bits long, and is reserved for future use. Implementations MUST set it to zero

(0) when encoding an attribute for sending in a packet. The contents SHOULD be ignored on reception.

Future specifications may define additional meaning for this field. Implementations therefore MUST NOT treat this field as invalid if it is non-zero.

Ext-Data

The contents of this field MUST be a valid data type as defined in the RADIUS Data Type registry. The Ext-Data field MUST NOT contain any of the following data types: "concat", "vsa", "extended", "long-extended", or "evs".

The Ext-Data field is one or more octets.

Implementations supporting this specification MUST use the Identifier of "Type.Extended-Type" to determine the interpretation of the Ext-Data field.

The length of the data MUST be taken as the sum of the lengths of the fragments (i.e. Ext-Data fields) from which it is constructed. Any interpretation of the resulting data MUST occur after the fragments have been reassembled. If the reassembled data does not match the expected format, each fragment MUST be treated as an "invalid attribute", and the reassembled data MUST be discarded.

We note that the maximum size of a fragmented attribute is limited only by the RADIUS packet length limitation. Implementations MUST be able to handle the case where one fragmented attribute completely

fills the packet.

3.17. evs

The "evs" data type encodes an "Extended Vendor-Specific" attribute, as given in [RFC6929] Section 2.4. The "evs" data type is used solely to extend the Vendor Specific space. It MAY appear inside of an "extended" or a "long-extended" data type. It MUST NOT appear in the contents of any other data type.

Where an implementation determines that an attribute of data type "evs" contains data which does not match the expected format, it SHOULD treat that attribute as being an "invalid attribute".

Name

evs

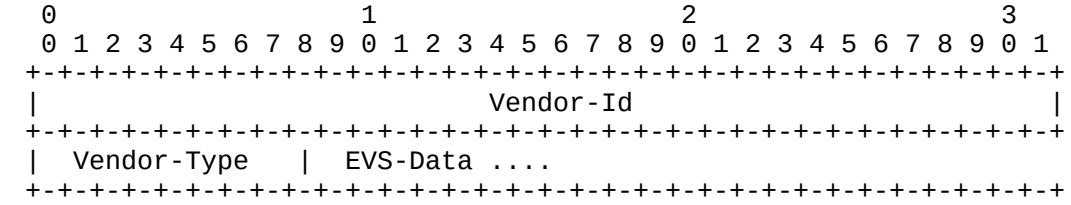
Value

17

Length

Six or more octets

Format



Subfields

Vendor-Id

The 4 octets are the Network Management Private Enterprise Code [PEN] of the Vendor in network byte order.

Vendor-Type

The Vendor-Type field is one octet. Values are assigned at the sole discretion of the Vendor.

EVS-Data

The EVS-Data field is one or more octets. It SHOULD encapsulate a previously defined RADIUS data type. Non-standard data types SHOULD NOT be used. We note that the EVS-Data field may be of data type "tlv".

The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. We recognise that Vendors have complete control over the contents and format of the Ext-Data field, while at the same time recommending that good practices be followed.

Further codification of the range of allowed usage of this field is outside the scope of this specification.

4. Updated Registries

This section defines a new IANA registry for RADIUS data types, and then updates the existing RADIUS Attribute Type registry to use the data types from the new registry.

4.1. Create a Data Type Registry

This section defines a new registry located under "RADIUS Types", called "Data Type". The "Registration Procedures" for the Data Type registry are "Standards Action".

The Data Type registry contains three columns of data, as follows.

Value

The number of the data type. The value field is an artifact of the registry, and has no on-the-wire meaning.

Name

The name of the data type. The name field is used only for the registry, and has no on-the-wire meaning.

Reference

The specification where the data type was defined.

The initial contents of the registry are as follows.

Value	Description	Reference
1	integer	[RFC2865], TBD
2	enum	[RFC2865], TBD
3	time	[RFC2865], TBD
4	text	[RFC2865], TBD
5	string	[RFC2865], TBD
6	concat	TBD
7	ifid	[RFC3162], TBD
8	ipv4addr	[RFC2865], TBD
9	ipv6addr	[RFC3162], TBD
10	ipv6prefix	[RFC3162], TBD
11	ipv4prefix	[RFC6572], TBD
12	integer64	[RFC6929], TBD
13	tlv	[RFC6929], TBD
14	evs	[RFC6929], TBD
15	extended	[RFC6929], TBD
16	long-extended	[RFC6929], TBD

4.2. Updates to the Attribute Type Registry

This section updates the RADIUS Attribute Type registry to have a new column, which is inserted in between the existing "Description" and "Reference" columns. The new column is named "Data Type". The contents of that column are the name of a data type, corresponding to the attribute in that row, or blank if the attribute type is unassigned. The name of the data type is taken from the RADIUS Data Type registry, as defined above.

The existing registration requirements for the RADIUS Attribute Type registry are otherwise unchanged.

NOTE TO RFC EDITOR: Before the document is published, please remove this note, and the following text in this section.

The updated registry follows in CSV format.

```
Value,Description,Data Type,Reference
1,User-Name,text,[RFC2865]
2,User-Password,string,[RFC2865]
3,CHAP-Password,string,[RFC2865]
4,NAS-IP-Address,ipv4addr,[RFC2865]
5,NAS-Port,integer,[RFC2865]
6,Service-Type,enum,[RFC2865]
7,Framed-Protocol,enum,[RFC2865]
8,Framed-IP-Address,ipv4addr,[RFC2865]
9,Framed-IP-Netmask,ipv4addr,[RFC2865]
10,Framed-Routing,enum,[RFC2865]
```

```
11,Filter-Id,text,[RFC2865]
12,Framed-MTU,integer,[RFC2865]
13,Framed-Compression,enum,[RFC2865]
14,Login-IP-Host,ipv4addr,[RFC2865]
15,Login-Service,enum,[RFC2865]
16,Login-TCP-Port,integer,[RFC2865]
17,Unassigned,,
18,Reply-Message,text,[RFC2865]
19,Callback-Number,text,[RFC2865]
20,Callback-Id,text,[RFC2865]
21,Unassigned,,
22,Framed-Route,text,[RFC2865]
23,Framed-IPX-Network,ipv4addr,[RFC2865]
24,State,string,[RFC2865]
25,Class,string,[RFC2865]
26,Vendor-Specific,vsa,[RFC2865]
27,Session-Timeout,integer,[RFC2865]
28,Idle-Timeout,integer,[RFC2865]
29,Termination-Action,enum,[RFC2865]
30,Called-Station-Id,text,[RFC2865]
31,Calling-Station-Id,text,[RFC2865]
32,NAS-Identifier,text,[RFC2865]
33,Proxy-State,string,[RFC2865]
34,Login-LAT-Service,text,[RFC2865]
35,Login-LAT-Node,text,[RFC2865]
36,Login-LAT-Group,string,[RFC2865]
37,Framed-AppleTalk-Link,integer,[RFC2865]
38,Framed-AppleTalk-Network,integer,[RFC2865]
39,Framed-AppleTalk-Zone,text,[RFC2865]
40,Acct-Status-Type,enum,[RFC2866]
41,Acct-Delay-Time,integer,[RFC2866]
42,Acct-Input-Octets,integer,[RFC2866]
43,Acct-Output-Octets,integer,[RFC2866]
44,Acct-Session-Id,text,[RFC2866]
45,Acct-Authentic,enum,[RFC2866]
46,Acct-Session-Time,integer,[RFC2866]
47,Acct-Input-Packets,integer,[RFC2866]
48,Acct-Output-Packets,integer,[RFC2866]
49,Acct-Terminate-Cause,enum,[RFC2866]
50,Acct-Multi-Session-Id,text,[RFC2866]
51,Acct-Link-Count,integer,[RFC2866]
52,Acct-Input-Gigawords,integer,[RFC2869]
53,Acct-Output-Gigawords,integer,[RFC2869]
54,Unassigned,,
55,Event-Timestamp,time,[RFC2869]
56,Egress-VLANID,integer,[RFC4675]
57,Ingress-Filters,enum,[RFC4675]
58,Egress-VLAN-Name,text,[RFC4675]
```

```
59,User-Priority-Table,string,[RFC4675]
60,CHAP-Challenge,string,[RFC2865]
61,NAS-Port-Type,enum,[RFC2865]
62,Port-Limit,integer,[RFC2865]
63,Login-LAT-Port,text,[RFC2865]
64,Tunnel-Type,enum,[RFC2868]
65,Tunnel-Medium-Type,enum,[RFC2868]
66,Tunnel-Client-Endpoint,text,[RFC2868]
67,Tunnel-Server-Endpoint,text,[RFC2868]
68,Acct-Tunnel-Connection,text,[RFC2867]
69,Tunnel-Password,string,[RFC2868]
70,ARAP-Password,string,[RFC2869]
71,ARAP-Features,string,[RFC2869]
72,ARAP-Zone-Access,enum,[RFC2869]
73,ARAP-Security,integer,[RFC2869]
74,ARAP-Security-Data,text,[RFC2869]
75,Password-Retry,integer,[RFC2869]
76,Prompt,enum,[RFC2869]
77,Connect-Info,text,[RFC2869]
78,Configuration-Token,text,[RFC2869]
79,EAP-Message,concat,[RFC2869]
80,Message-Authenticator,string,[RFC2869]
81,Tunnel-Private-Group-ID,text,[RFC2868]
82,Tunnel-Assignment-ID,text,[RFC2868]
83,Tunnel-Preference,integer,[RFC2868]
84,ARAP-Challenge-Response,string,[RFC2869]
85,Acct-Interim-Interval,integer,[RFC2869]
86,Acct-Tunnel-Packets-Lost,integer,[RFC2867]
87,NAS-Port-Id,text,[RFC2869]
88,Framed-Pool,text,[RFC2869]
89,CUI,string,[RFC4372]
90,Tunnel-Client-Auth-ID,text,[RFC2868]
91,Tunnel-Server-Auth-ID,text,[RFC2868]
92,NAS-Filter-Rule,text,[RFC4849]
93,Unassigned,,
94,Originating-Line-Info,string,[RFC7155]
95,NAS-IPv6-Address,ipv6addr,[RFC3162]
96,Framed-Interface-Id,ifid,[RFC3162]
97,Framed-IPv6-Prefix,ipv6prefix,[RFC3162]
98,Login-IPv6-Host,ipv6addr,[RFC3162]
99,Framed-IPv6-Route,text,[RFC3162]
100,Framed-IPv6-Pool,text,[RFC3162]
101,Error-Cause Attribute,enum,[RFC3576]
102,EAP-Key-Name,string,[RFC4072][RFC7268]
103,Digest-Response,text,[RFC5090]
104,Digest-Realm,text,[RFC5090]
105,Digest-Nonce,text,[RFC5090]
106,Digest-Response-Auth,text,[RFC5090]
```

```
   107,Digest-Nextnonce,text,[RFC5090]
   108,Digest-Method,text,[RFC5090]
   109,Digest-URI,text,[RFC5090]
   110,Digest-Qop,text,[RFC5090]
   111,Digest-Algorithm,text,[RFC5090]
   112,Digest-Entity-Body-Hash,text,[RFC5090]
   113,Digest-CNonce,text,[RFC5090]
   114,Digest-Nonce-Count,text,[RFC5090]
   115,Digest-Username,text,[RFC5090]
   116,Digest-Opaque,text,[RFC5090]
   117,Digest-Auth-Param,text,[RFC5090]
   118,Digest-AKA-Auts,text,[RFC5090]
   119,Digest-Domain,text,[RFC5090]
   120,Digest-Stale,text,[RFC5090]
   121,Digest-HA1,text,[RFC5090]
   122,SIP-AOR,text,[RFC5090]
   123,Delegated-IPv6-Prefix,ipv6prefix,[RFC4818]
   124,MIP6-Feature-Vector,string,[RFC5447]
   125,MIP6-Home-Link-Prefix,ipv6prefix,[RFC5447]
   126,Operator-Name,text,[RFC5580]
   127,Location-Information,string,[RFC5580]
   128,Location-Data,string,[RFC5580]
   129,Basic-Location-Policy-Rules,string,[RFC5580]
   130,Extended-Location-Policy-Rules,string,[RFC5580]
   131,Location-Capable,enum,[RFC5580]
   132,Requested-Location-Info,enum,[RFC5580]
   133,Framed-Management-Protocol,enum,[RFC5607]
   134,Management-Transport-Protection,enum,[RFC5607]
   135,Management-Policy-Id,text,[RFC5607]
   136,Management-Privilege-Level,integer,[RFC5607]
   137,PKM-SS-Cert,concat,[RFC5904]
   138,PKM-CA-Cert,concat,[RFC5904]
   139,PKM-Config-Settings,string,[RFC5904]
   140,PKM-Cryptosuite-List,string,[RFC5904]
   141,PKM-SAID,text,[RFC5904]
   142,PKM-SA-Descriptor,string,[RFC5904]
   143,PKM-Auth-Key,string,[RFC5904]
   144,DS-Lite-Tunnel-Name,text,[RFC6519]
   145,Mobile-Node-Identifier,string,[RFC6572]
   146,Service-Selection,text,[RFC6572]
   147,PMIP6-Home-LMA-IPv6-Address,ipv6addr,[RFC6572]
   148,PMIP6-Visited-LMA-IPv6-Address,ipv6addr,[RFC6572]
   149,PMIP6-Home-LMA-IPv4-Address,ipv4addr,[RFC6572]
   150,PMIP6-Visited-LMA-IPv4-Address,ipv4addr,[RFC6572]
   151,PMIP6-Home-HN-Prefix,ipv6prefix,[RFC6572]
   152,PMIP6-Visited-HN-Prefix,ipv6prefix,[RFC6572]
   153,PMIP6-Home-Interface-ID,ifid,[RFC6572]
   154,PMIP6-Visited-Interface-ID,ifid,[RFC6572]
```

```
155,PMIP6-Home-IPv4-HoA,ipv4prefix,[RFC6572]
156,PMIP6-Visited-IPv4-HoA,ipv4prefix,[RFC6572]
157,PMIP6-Home-DHCP4-Server-Address,ipv4addr,[RFC6572]
158,PMIP6-Visited-DHCP4-Server-Address,ipv4addr,[RFC6572]
159,PMIP6-Home-DHCP6-Server-Address,ipv6addr,[RFC6572]
160,PMIP6-Visited-DHCP6-Server-Address,ipv6addr,[RFC6572]
161,PMIP6-Home-IPv4-Gateway,ipv4addr,[RFC6572]
162,PMIP6-Visited-IPv4-Gateway,ipv4addr,[RFC6572]
163,EAP-Lower-Layer,enum,[RFC6677]
164,GSS-Acceptor-Service-Name,text,[RFC7055]
165,GSS-Acceptor-Host-Name,text,[RFC7055]
166,GSS-Acceptor-Service-Specifics,text,[RFC7055]
167,GSS-Acceptor-Realm-Name,text,[RFC7055]
168,Framed-IPv6-Address,ipv6addr,[RFC6911]
169,DNS-Server-IPv6-Address,ipv6addr,[RFC6911]
170,Route-IPv6-Information,ipv6prefix,[RFC6911]
171,Delegated-IPv6-Prefix-Pool,text,[RFC6911]
172,Stateful-IPv6-Address-Pool,text,[RFC6911]
173,IPv6-6rd-Configuration,tlv,[RFC6930]
174,Allowed-Called-Station-Id,text,[RFC7268]
175,EAP-Peer-Id,string,[RFC7268]
176,EAP-Server-Id,string,[RFC7268]
177,Mobility-Domain-Id,integer,[RFC7268]
178,Preauth-Timeout,integer,[RFC7268]
179,Network-Id-Name,string,[RFC7268]
180,EAPoL-Announcement,concat,[RFC7268]
181,WLAN-HESSID,text,[RFC7268]
182,WLAN-Venue-Info,integer,[RFC7268]
183,WLAN-Venue-Language,string,[RFC7268]
184,WLAN-Venue-Name,text,[RFC7268]
185,WLAN-Reason-Code,integer,[RFC7268]
186,WLAN-Pairwise-Cipher,integer,[RFC7268]
187,WLAN-Group-Cipher,integer,[RFC7268]
188,WLAN-AKM-Suite,integer,[RFC7268]
189,WLAN-Group-Mgmt-Cipher,integer,[RFC7268]
190,WLAN-RF-Band,integer,[RFC7268]
191,Unassigned,,
192-223,Experimental Use,,[RFC3575]
224-240,Implementation Specific,,[RFC3575]
241,Extended-Attribute-1,extended,[RFC6929]
241.1,Frag-Status,integer,[RFC7499]
241.2,Proxy-State-Length,integer,[RFC7499]
241.3,Response-Length,integer,[RFC7930]
241.4,Original-Packet-Code,integer,[RFC7930]
241.{5-25},Unassigned,,
241.26,Extended-Vendor-Specific-1,evs,[RFC6929]
241.{27-240},Unassigned,,
241.{241-255},Reserved,,[RFC6929]
```

```
242,Extended-Attribute-2,extended,[RFC6929]
242.{1-25},Unassigned,,
242.26,Extended-Vendor-Specific-2,evs,[RFC6929]
242.{27-240},Unassigned,,
242.{241-255},Reserved,,[RFC6929]
243,Extended-Attribute-3,extended,[RFC6929]
243.{1-25},Unassigned,,
243.26,Extended-Vendor-Specific-3,evs,[RFC6929]
243.{27-240},Unassigned,,
243.{241-255},Reserved,,[RFC6929]
244,Extended-Attribute-4,extended,[RFC6929]
244.{1-25},Unassigned,,
244.26,Extended-Vendor-Specific-4,evs,[RFC6929]
244.{27-240},Unassigned,,
244.{241-255},Reserved,,[RFC6929]
245,Extended-Attribute-5,long-extended,[RFC6929]
245.{1-25},Unassigned,,
245.26,Extended-Vendor-Specific-5,evs,[RFC6929]
245.{27-240},Unassigned,,
245.{241-255},Reserved,,[RFC6929]
246,Extended-Attribute-6,long-extended,[RFC6929]
246.{1-25},Unassigned,,
246.26,Extended-Vendor-Specific-6,evs,[RFC6929]
246.{27-240},Unassigned,,
246.{241-255},Reserved,,[RFC6929]
247-255,Reserved,,[RFC3575]
```

5. Security Considerations

This specification is concerned solely with updates to IANA registries. As such, there are no security considerations with the document itself.

However, the use of inconsistent names and poorly-defined entities in a protocol is problematic. Inconsistencies in specifications can lead to security and interoperability problems in implementations. Further, having one canonical source for the definition of data types means an implementor has fewer specifications to read. The implementation work is therefore simpler, and is more likely to be correct.

The goal of this specification is to reduce ambiguities in the RADIUS protocol, which we believe will lead to more robust and more secure implementations.

6. IANA Considerations

IANA is instructed to create one new registry as described above in Section 4.1. The "TBD" text in that section should be replaced with the RFC number of this document when it is published.

IANA is instructed to update the RADIUS Attribute Type registry, as described above in Section 4.2.

IANA is instructed to require that all allocation requests in the RADIUS Attribute Type registry contain a "Data Type" field. That field is required to contain one of the "Data Type" names contained in the RADIUS Data Type registry.

IANA is instructed to require that updates to the RADIUS Data Type registry contain the following fields, with the associated instructions:

* Value. IANA is instructed to assign the next unused integer in sequence to new data type definitions.

* Name. IANA is instructed to require that this name be unique in the registry.

* Reference. IANA is instructed to update this field with a reference to the document which defines the data type.

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.

[RFC3162]
Aboba, B., Zorn, G., and D. Mitton, "RADIUS and IPv6", RFC 3162, August 2001.

[RFC3629]
Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 3629, November 2003.

[RFC4072]
Eronen, P., et al, "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, February 2013.

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

[RFC6572]
Xia, F., et al, "RADIUS Support for Proxy Mobile IPv6", RFC 6572, June 2012.

[RFC7499]
Perez-Mendez, A. Ed., et al, "Support of Fragmentation of RADIUS Packets", RFC 7499, April 2015.

7.2. Informative References

[RFC2868]
Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, June 2000.

[RFC2869]
Rigney, C., et al, "RADIUS Extensions", RFC 2869, June 2000.

[RFC5234]
Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 5234, January 2008.

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

[RFC7268]
Aboba, B, et al, "RADIUS Attributes for IEEE 802 Networks", RFC 7268, July 2015.

[RFC7499]
Perez-Mendez A., et al, "Support of Fragmentation of RADIUS Packets", RFC 7499, April 2015.

[PEN]
http://www.iana.org/assignments/enterprise-numbers

Acknowledgments

Thanks to the RADEXT WG for patience and reviews of this document.

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project

Email: aland@freeradius.org

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.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2017

B. Weis
Cisco Systems

October 25, 2016

RADIUS Extensions for Manufacturer Usage Description
draft-weis-radext-mud-00

Abstract

A Manufacturer Usage Description (MUD) is a file describing the expected use of a class of devices, usually an Internet of Things class of devices. It is prepared by a manufacturer and placed on a generally available web server, and is addressable via a Uniform Resource Identifier (URI). The URI is often included in a discovery protocol (e.g., DNS, LLDP). A Network Access Server (NAS) in the path of the discovery protocol can collect and forward the URI to a RADIUS server, which processes the URI. This draft defines the RADIUS extension needed for the NAS to forward the URI to the RADIUS server.

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 April 28, 2017.

Copyright Notice

Table of Contents

1. Introduction

Enterprise networks often use Port-Based Network Access Control [IEEE802.1X], where the Authentication Server is a RADIUS server [RFC2865]. In some cases a device will authenticate itself to the network using IEEE 802.1X with a digital certificate (e.g., an IEEE 802.1AR Secure Device ID [IEEE802.1AR]) that has been placed into the device by the manufacturer. Manufacturer Usage Description (MUD) [I-D.ietf-opsawg-mud] has defined an optional extension for digital certificates, which consists of a Uniform Resource Identifier (URI) that identifies the MUD file. A MUD file contains identification and network access information for a particular class of device. This information can be used to generate authorization policy such as an Access Control List (ACL) describing required network access for the device.

However, there are cases where a MUD URI is not included in a device's digital certificate, or it does not support the use of digital certificates, or may not even support an IEEE 802.1X Supplicant. This will often be the case with IoT devices, which is a primary use case for the use of MUD. In each of these situations, a device could benefit from distributing a MUD URI in a discovery message (e.g., a DHCP or LLDP message as defined in [I-D.ietf-opsawg-mud]), in hopes that a network element device will receive and consume it.

As shown in Figure 1, a Network Access Server (NAS) can observe the discovery message with the MUD URI and forward it to a RADIUS server. This can be done as part of a MAC Authentication Bypass (MAB) message. MAB is a common alternative approach of port-based network access control used for devices that cannot support a IEEE 802.1X Supplicant. The RADIUS server and an associated MUD Controller (defined in [I-D.ietf-opsawg-mud]) will work together to resolve the URI and translate the resulting MUD file into authorization policy. The RADIUS server distributes to the NAS authorization RADIUS attributes (e.g., an ACL describing required network access) to apply to messages received from the device.

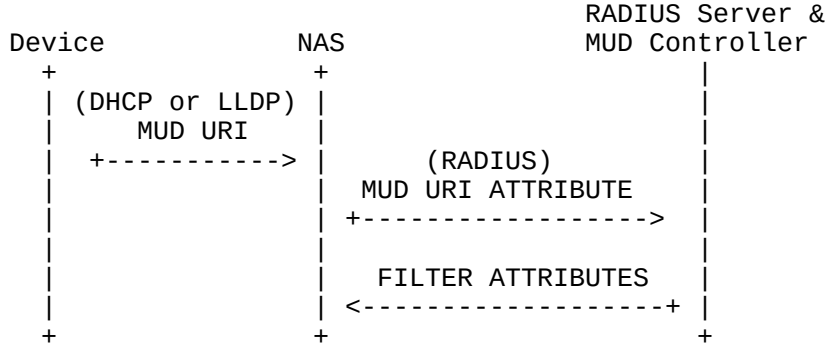


Figure 1: RADIUS Message Flow

The only missing piece in this workflow is the ability for the NAS to relay the MUD URI to the RADIUS server. This draft defines a new RADIUS attribute for this purpose. The expectation is that the MUD URI will be passed in Access Request or Accounting messages.

1.1. Requirements notation

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].

1.2. Terminology

The following key terms are used throughout this document:

MUD Controller An entity that requests a MUD file from the MUD server, and processes the MUD file upon receipt.

MUD file A file containing a MUD Yang file definition, as defined in [I-D.ietf-opsawg-mud]

MUD URI A URI pointing to a MUD file, typically located on a web server.

2. Acronyms and Abbreviations

The following acronyms and abbreviations are used throughout this document

DHCP Dynamic Host Configuration Protocol

IoT Internet of Things

LLDP Link Layer Discovery Protocol

MAB MAC Authentication Bypass

MUD Manufacturer Usage Description

NAS Network Access Server

3. Extended Attribute for the MUD URI

This attribute is of type "TLV" as defined in the RADIUS Protocol Extensions [RFC6929]. It is named the MUD-URI Attribute, and is defined in Figure 2.

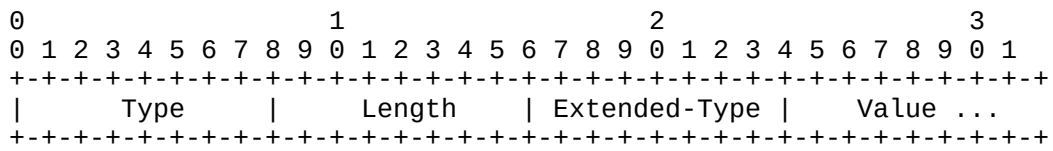


Figure 2: MUD TLV format

Type

TBD1

Length

This field indicates the total length in bytes of all fields of this attribute, including the Type, Length, Extended-Type, and the entire length of the Value.

Extended-Type

TBD2

Value

A MUD URI as defined in [I-D.ietf-opsawg-mud], and MUST conform to the syntax defined a URI [RFC3986].

4. MUD URI processing

When a NAS receives a MUD URI, it forwards it to a RADIUS server using the Extended Attribute described in Section 3.

When a RADIUS server receives a MUD URI, it works in conjunction with a MUD Controller to retrieve the MUD file and processes it as described in [I-D.ietf-opsawg-mud]. They determine filter policies based on the MUD file, and the RADIUS server passes these filter policies to the NAS using commonly used RADIUS filter attributes.

Finally, the NAS receives the RADIUS filter attributes and applies them to the network traffic associated with the new device.

5. Security Considerations

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

Security considerations regarding the integrity of the MUD URI are outside the scope of this document, but it may be helpful to consider how a network using MAB might use a MUD URI. When retrieved from an authenticated device a NAS does not absolutely know if this MUD file is correct for the device that proffers the MUD URI, but it can use the MUD file as a hint as to the type of device. A NAS may be able to correlate the claimed device type with other policy for this device using other mechanisms. It should also be noted that the intent of a MUD policy description is to severely limit the network access of the device (e.g., using filters), rather than grant wide access to a device. Therefore, the action of proffering a MUD URI indicates a willingness to have its network access restricted rather than opened.

6. IANA Considerations

TBD1: One of the RADIUS Types that indicates an Extended Type

TBD2: A RADIUS Extended Type value.

7. Acknowledgements

The author thanks Nancy Cam-Winget for her thoughtful review, which resulted in substantial improvements to the memo.

8. References

8.1. Normative References

[I-D.ietf-opsawg-mud]
Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", draft-ietf-opsawg-mud-01 (work in progress), September 2016.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

8.2. Informative Reference

[IEEE802.1AR]
IEEE Computer Society, "802.1AR-2009 - IEEE Standard for Local and metropolitan area networks--Secure Device Identity", February 2010, <https://standards.ieee.org/findstds/standard/802.1AR-2009.html>.

[IEEE802.1X]
IEEE Computer Society, "802.1X-2010 - IEEE Standard for Local and metropolitan area networks--Port-Based Network Access Control", February 2010, <https://standards.ieee.org/findstds/standard/802.1X-2010.html>.

[RFC6929] DeKok, A. and A. Lior, "Remote Authentication Dial In User Service (RADIUS) Protocol Extensions", RFC 6929, DOI 10.17487/RFC6929, April 2013, <http://www.rfc-editor.org/info/rfc6929>.

Author's Address

Brian Weis
Cisco Systems
170 W. Tasman Drive
San Jose, California 95134-1706
USA

Phone: +1 408 526 4796
Email: bew@cisco.com