radextra BOF
Internet-Draft
Intended status: Standards Track
Expires: 11 January 2024

M. Cullen
Painless Security
A. DeKok
FreeRADIUS
M. Donnelly
Painless Security
J. Howlett
Federated Solutions
10 July 2023

Status-Realm and Loop Prevention for the Remote Dial-In User Service (RADIUS)
draft-cullen-radextra-status-realm-01

Abstract

This document describes extension to the Remote Authentication Dial-In User Service (RADIUS) protocol to allow participants in a multi-hop RADIUS proxy fabric to check the status of a remote RADIUS authentication realm, gain visibility into the path that a RADIUS request will take across the RADIUS proxy fabric, and mitigate or prevent RADIUS proxy loops.

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 11 January 2024.

Copyright Notice

Table of Contents

1. Introduction

This document describes an extension to the Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865], to allow participants in a multi-hop RADIUS proxy fabric to check the status of a remote RADIUS authentication realm, gain visibility into the path that a RADIUS request will take across the RADIUS proxy fabric, and mitigate or prevent RADIUS proxy forwarding loops.

This document defines two new RADIUS Packet Type Codes:

* Status-Realm-Request (TBD)

* Status-Realm-Response (TBD)

This document also defines the following RADIUS Attributes:

* Status-Realm-Response-Code (TBD)

* Max-Hop-Count (TBD)

* Server-Identifier (TBD)

2. 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].

3. Terminology

The following terms are used throughout this document. Their definitions are included here for consistency and clarity.

RADIUS Request	A RADIUS Request is the first message in a RADIUS message exchange. RADIUS request message types include: Access-Request, Accounting-Request, and Status-Server. This document defines a new RADIUS Request message type: Status-Realm-Request.
RADIUS Response	A RADIUS Response is any RADIUS message sent in reply to a RADIUS Request. RADIUS reponse message types include: Access-Accept, Access-Challenge, Access-Reject, Accounting-Response. This document defines a new RADIUS Response message type: Status-Realm-Response.

RADIUS Instance	A single device or software module that implements the RADIUS protocol.
RADIUS Client	A RADIUS Client is a RADIUS Instance that sends RADIUS Request messages and recevies RADIUS Reponse messages in reply.
RADIUS Server	A RADIUS Server is a RADIUS Instance that receives RADIUS Requests and sends RADIUS Response messages in reply.
Authentication Request	An Authentication Request is sent to authenticate a particular user within a particular realm. The user and realm information are typically included in a User-Name Attribute [RFC2865] within the Authentication Request.
Authentication Server	An Authentication Server is a RADIUS Server that receives Access-Requests for a given RADIUS Realm, and sends Access-Access, Access-Challenge or Access-Reject messages in response. A single Authentication Server may serve more than one Authentication Realm.
Authentication Realm	An Authentication Realm consists of a group of users within a single organization that can be authenticated using RADIUS. A single Authentication Realm MAY be served by more than one Authentication Server.
Target Realm	The Target Realm of a RADIUS Request is the RADIUS Realm toward which the Request is directed. The Target Realm is typically contained within the "User-Name" attribute of a Request.
RADIUS Proxy	A RADIUS Proxy receives RADIUS Requests and forwards then towards the Target Realm included in the RADIUS Request message. It also receives the corresponding RADIUS Respone message and fowards them back towards the RADIUS Client that originated the request. In this context forwarding a RADIUS Requst consists of generating a new RADIUS Request containing information from the original Request, and sending it to the

	configured next-hop RADIUS server for the Target Realm. Forwarding a RADIUS Response consists of sending it to the RADIUS Server from which the corresponding Request was received.
RADIUS Proxy Fabric	A multi-hop group of inter-connected RADIUS Servers that Proxy requests among themselves towards a set of Target Realms.
RADIUS Proxy Path	The RADIUS Server Path is a the set of RADIUS Servers that a RADIUS Request traverses from the first RADIUS Server that is contacted by the RADIUS Client to the final RADIUS Server that responds to the Request.
Proxy Loop	A Proxy Loop may occur when two or more RADIUS Proxies are configured such that a RADIUS Request follow a circular path through the Proxy Fabric, never reaching the Target Realm. This is a pathological and potentially damaging misconfiguration.
First-Hop Server	The First-Hop Server is the first RADIUS Server within a Proxy Fabric to recieve a RADIUS Request. In some cases, the First-Hop RADIUS Server may receive the request from a separate RADIUS Client. In other case, the First-Hop RADIUS Server and the RADIUS Client may be running in a single RADIUS Instance.
Last-Hop Proxy	The Last-Hop Proxy is the last RADIUS Proxy to forward a RADIUS Request before it reaches the Authentication Server. Depending on its configuraiton, the Last-Hop Proxy may or may not know that is the Last-Hop Proxy for a given RADIUS Request.

Note: It is possible for a single RADIUS instance to server in multiple roles. For example, it is common for a RADIUS Server to act as an Authentication Server for some Realms, while acting as a Proxy for other Realms. A RADIUS Proxy will, by its nature, act as a RADIUS Server for some RADIUS messages while acting as a RADIUS Client for others. The requirements in this document apply to all RADIUS instances whenever they are acting in the role to which the requirement applies.

4. Overview

This document defines two functional extensions to RADIUS: Querying the status of a remote RADIUS Realm (Status-Realm), and mitigating, detecting and preventing loops in a RADIUS Proxy forwarding loops (Proxy Loop Prevention). This section contains a short overview of each function. Detailed definitions and requirements are covered in later sections of this document.

4.1. Status-Realm Overview

Status-Realm-Request messages are sent by RADIUS Clients to to query the reachability and status of a particular Target Realm. In some cases, the Status-Realm RADIUS Client may be able to reach an Authentication Server for the Target Realm directly. In other cases, the RADIUS Client will send the initial Status-Realm request to a RADIUS Proxy, which will forward the Status-Realm-Request toward the indicated realm.

Status-Realm-Requests may be sent to the RADIUS authentication port or the RADIUS accounting port of the first-hop RADIUS server. RADIUS proxies should forward Status-Realm-Requests received on the authentication port to the authentication port of the next-hop RADIUS server. Status-Realm-Requests received on the accounting port should, similarly, be forwarded to the accounting port of the next-hop server.

When a Status-Realm-Request packet is received by an Authentication Server for the Target Realm, the Authentication Server MUST respond with a Status-Realm-Response packet.

If an intermediate RADIUS Proxy is unable to forward a Status-Realm-Request packet towards the Target Realm, either because it has no information about how to reach the Target Realm, or because there are no reachable Authentication Servers for the Target Realm, the RADIUS Proxy MUST return a Status-Realm-Response packet containing a Status-Realm-Response-Code attribute.

Status-Realm packets allow the sender to determine the reachability and status of a Authentication Realm, without requiring a direct RADIUS connection to a RADIUS Server for the Target realm, and without requiring credentials for an authorized user within that realm. This can be useful for debugging RADIUS authentication issues, identifying routing issues within a RADIUS proxy fabric, or monitoring realm availability.

Using the Max-Hop-Count attribute defined in this document, RADIUS Clients can also implement "traceroute-like" functionality, discovering a series of proxies on route to a target realm.

4.2. RADIUS Loop Prevention Overview

RADIUS Proxies are configured to know which next-hop RADIUS Server to use for a given Target Realm. There is no dynamic routing protocol or tree-spanning protocol in use, so Proxy Loops are a common occurence due to misconfiguration. These loops can be controlled or prevented using implementation-specific or operator-specific mechanisms, but it would be useful to have well-defined, common mechanism.

The Max-Hop-Count attribute described in this document can be used to mitigate the damage caused by Proxy Loops. The Max-Hop-Count attribute is set to a small integer by the RADIUS Client or First-Hop RADIUS Server. The value is decremented each time a RADIUS message is proxied. When the Max-Hop-Count reaches zero, the request is discarded, ending the loop.

This document also defines a more effective method of detecting and preventing Proxy Forwarding Loops: RADIUS Loop Prevention. This document defines a RADIUS Server-Identifier attribute that is used to uniquely identify a RADIUS Server. When a RADIUS Proxy receives a RADIUS Request packet, it checks to see if the Request contains a Server-Identifier attribute indicating that it has already processed this packet. If so, it discards the packet. If not, it adds its own Server Identifier to the packet before forwarding it.

5. Packet Formats

This section describes the RADIUS packet formats for Status-Realm-Request and Status-Realm-Response packets. Status-Realm-Requests are sent in the same format, whether they are sent to the authentication port or the accounting port.

5.1. Status-Realm-Request Packet

Status-Realm-Request packets reuse the RADIUS packet format, with the fields and values for those fields as defined in [RFC2865], Section 3.

A Status-Realm-Request packet MUST include a Message-Authenticator attribute, as defined in [RFC2869], section 5.14. The Message-Authenticator provides per-packet authentication and integrity protection. The Authenticator field of a Status-Realm-Request packet MUST be generated using the same method as that used for the Request Authenticator field of Access-Request packets.

A Status-Realm-Request packets MUST include a User-Name Attribute, containing the Target Realm for the Request. The 'user' portion of the User-Name SHOULD be ignored, if present.

A Status-Realm-Request message MUST also include a Max-Hop-Count attribute, as defined above.

Status-Realm-Requests MAY include NAS-Identifier, and one of (NAS-IP-Address or NAS-IPv6-Address). These attributes are not necessary for the operation of Status-Realm, but may be useful information to a server that receives those packets.

Status-Realm-Request packets MUST NOT contain authentication credentials (such as User-Password, CHAP-Password, EAP-Message) or User or NAS accounting attributes (such as Acct-Session-Id, Acct-Status-Type, Acct-Input-Octets).

5.2. Status-Realm-Response Packet

Status-Realm-Response packets reuse the RADIUS packet format, with the fields and values for those fields as defined in [RFC2865], Section 3.

The Response Authenticator field of a Status-Realm-Response packet MUST be generated using the same method used for calculating the Response Authenticator of an Access-Accept or an Access-Reject sent in response to an Access-Request, with the Status-Realm-Request Request Authenticator taking the place of the Access-Request Request Authenticator.

The Status-Realm-Response packet MUST contain a Status-Realm-Response-Code attribute, as defined below, indicating the results of the Status-Realm request.

The Status-Realm-Response packet MAY contain the following attributes: Reply-Message, Message-Authenticator, Server-Information.

Note that when a server responds to a Status-Realm-Request packet, it MUST NOT send more than one Status-Realm-Response packet.

6. Max-Hop-Count Attribute

This section defines a new RADIUS attribute, Max-Hop-Count (TBD). The value of the Max-Hop-Count attribute is an integer, as defined in [RFC8044], Section 3.1. Valid values are small positive integers, 0 to 255.

This attribute is used to limit the number of RADIUS servers that will proxy a packet before it reaches its final destination. When a RADIUS server that implements the Max-Hop-Count Attribute determines that it wants to proxy a RADIUS Request to another RADIUS Server, it will check the Max-Hop-Count attribute. If the Max-Hop-Count attribute is present and the value is zero, the Request MUST NOT be forwarded and an error response SHOULD be returned, as appropriate to the request type. If the Max-Hop-Count is greater than zero, the proxy server MUST decrement the hop count by 1 before forwarding the request.

In the context of Status-Realm-Requests, this attribute can be used to implement "traceroute-like" functionality. By sending a series of Status-Realm-Requests with incremented values of Max-Hop-Count, starting with a Max-Hop-Count value of O, the RADIUS Client will receive a series of Status-Realm-Responses from the RADIUS Proxies on the Proxy Path to a given Target Realm.

When used on other types of RADIUS Request messages, this option can mitigate the damage caused by RADIUS proxy loops. It is therefore possible that a RADIUS Client or a RADIUS proxy server will support the Max-Hop-Count attribute, even if they do not support Status-Realm. When used to limit RADIUS proxy loops, it is RECOMMENDED that the value of the Max-Hop-Count attribute be set to 32, by default.

For any type of RADIUS request message, setting the Max-Hop-Count attribute to 0 effectively requests that the request message not be proxied. Setting the attribute to a value greater than 0 requests that the request message be proxied across at most that many intermediate proxies between the visited and home server.

If this attribute is not present on a RADIUS Request received from a RADIUS Client, the First-Hop RADIUS Server MAY add this option, setting it to the default value of 32, or to any valid, configured value.

7. Status-Realm-Response-Code Attribute

This section defines a new RADIUS attribute, Status-Realm-Response-Code (TBD). This is of type tlv, as defined in [RFC8044], section 3.13. It contains 3 sub-attributes:

* Response-Code (Type = 1)

* Hop-Count (Type = 2)

* Responding-Server (Type = 3)

Response-Code is of type 'integer', as defined in [RFC8044], Section 3.1. Exactly one Response-Code sub-attribute MUST be included in in every Status-Realm-Response-Code attribute. It will contain one of the following values:

0	The target realm is available
1	No proxy route to the target realm
2	No available servers for the target realm
3	The target realm is missing or invalid
4	Max-Hop-Count exceeded
5-255	Unspecified error, the target realm is unreachable
256	Administratively prohibited, target realm status unknown
257	Internal error, target realm status unknown
258	Bad Status-Realm-Request, missing or invalid Target Realm in the request message, target realm status unknown
259	Bad Status-Realm-Request, missing or invalid Max-Hop-Count, target realm status unknown
260-511	Unspecified error, Target Realm status unknown
512+	Reserved

Response-Code values from 0 to 255 indicate the status of the target realm on the RADIUS network. Response-Code values from 256 to 511 indicate errors in processing the Status-Realm request, and cannot indicate the status of the target realm.

Hop-Count is of type 'integer'. Valid values are 0-255. The value of this sub-attribute MUST be set to the value of the Max-Hop-Count attribute in the received Status-Realm-Request. If no Max-Hop-Count is included in the Status-Realm-Request message, this sub-attribute MUST be omitted.

Responding-Server is of type 'tlv', as defined in [RFC8044], Section 3.13. This sub-attribute MUST be returned in every Status-Realm-Response attribute. The value field of this sub-attribute contains a Server-Information Attribute for the responding server, as described below.

8. Server-Information Attribute

The Server-Information attribute is used to identify a specific RADIUS Server. It MAY be added to any RADIUS Request message to indicate that a particular RADIUS Server has processed the Request. If present in a RADIUS Request, it SHOULD be copied into the corresponding RADIUS Response. RADIUS Servers SHOULD NOT add Server-Information attributes to Response messages when processing Responses.

This attribute is of type 'tlv', as defined in [RFC8044], Section 3.13. The value of this attribute consists of a set of sub-attributes, all of type 'tlv'. Each sub-attribute contains an identifier for a RADIUS proxy. The Server-Identifier MUST have at least one sub-attribute and MAY have more than one sub-attribute. If multiple sub-attributes are present, a RADIUS proxy MUST match all of the sub-attributes in order to match the identifier.

The following sub-attributes may be included in the value field of a Server-Information Attribute. The Type code for each sub-attribute is included in parenthesis.

* Server-Operator (Type = 1)

* Server-Identifier (Type = 2)

* Hop-Count (Type = 3)

* Time-Delta (Type = 4)

The Server-Operator is of type 'string'. It is the analogue of the Operator-Name, as defined in [RFC5580].

The Server-Identifier in an analogue of the NAS-Identifier defined in [RFC2865]. It indicates the name of this particular proxy server. This field is used to identify which server processed the Request, among those operated by the organization indicated in the Server-Operator sub-attribute.

The Time-Delta attribute is of type 'integer'. It represents the number of milliseconds the request took to return through this proxy server. For the target server, this value SHOULD be 0.

8.1. Status-Realm Responding-Server

This attribute is also included as the sub-attribute Responding-Server within the Status-Realm-Response-Code attribute, defined above, to indicate which RADIUS Server has sent the Status-Realm-Response message. Thus, a Status-Realm response may contain many Server-Information attributes, as well as a Status-Realm-Response-Code attribute with the Responding-Server sub-attribute, which has the same structure.

If a Status-Realm request targeting "target-realm" is routed over proxy servers P1 and P2 before reaching the "target-realm" home server, then the response message will contain these attributes:

* Server-Information:
  - Server-Operator: P1
  - Server-Identifier: P1
  - Hop-Count: 32
  - Time-Delta: 90
* Server-Information:
  - Server-Operator: P2
  - Server-Identifier: P2-Alpha
  - Hop-Count: 31
  - Time-Delta: 60
* Status-Realm-Response-Code:
  - Response-Code: 0 (Available)
  - Hop-Count: 30
  - Responding-Server:
    - Server-Operator: target-realm
    - Server-Identifier: radius1.target-realm
    - Hop-Count: 30

- Time-Delta: 0

9. Status-Realm Implementation Requirements

This section describes implementation details and requirements for RADIUS Clients and servers that support Status-Realm.

9.1. RADIUS Client Requirements

When Status-Realm-Request packets are sent from a RADIUS Client, they MUST NOT be retransmitted. Instead, the Identity field MUST be changed every time a packet is transmitted. The old packet should be discarded, and a new Status-Realm-Request packet should be generated and sent, with new Identity and Authenticator fields.

RADIUS Clients MUST include the Message-Authenticator attribute in all Status-Realm-Request packets. Failure to do so would mean that the packets could be trivially spoofed, leading to potential denial-of-service (DoS) attacks.

The RADIUS Client MUST include a User-Name attribute in the request. The "user" portion of the username SHOULD be omitted. The "realm" portion of the username is the target realm for the Status-Realm request.

RADIUS Clients that support Status-Realm-Requests SHOULD allow a user or administrator to set or configure the Count value of the Max-Hop-Count Attribute described above. If a different value is not indicated, the RADIUS Client SHOULD include a Max-Hop-Count attribute with a Count value of 32 in the Status-Realm-Request packet to prevent the possibility that Status-Realm-Requests will loop indefinitely.

The RADIUS Client MAY increment packet counters as a result of sending a Status-Realm-Resquest or receiving a Status-Realm-Response. The RADIUS Client MUST NOT perform any other action that is normally performed when it receives a Response packet, such as permitting a user to have login access to a port.

RADIUS Clients MAY send Status-Realm-Request packets to the RADIUS destination ports from the same source port(s) used to send other Request packets. Other RADIUS Clients MAY choose to send Status-Realm-Request packets from a unique source port that is not used to send other Request packets.

In the case where a RADIUS Client sends a Status-Realm-Request packets from a source port also used to send other Request packets, the Identifier field MUST be unique across all outstanding Request

packets for that source port, independent of the value of the RADIUS Code field for those outstanding requests. Once the RADIUS Client has either received a corresponding Status-Realm-Response packet or determined that the Status-Realm-Request has timed out, it may reuse the Identifier in another Request packet.

The RADIUS Client MUST validate the Response Authenticator in the Status-Realm-Response. If the Response Authenticator is not valid, the packet MUST be silently discarded. If the Response Authenticator is valid, then the packet MUST be deemed to be a valid response.

9.2. Server Requirements

Servers SHOULD permit administrators to globally enable or disable the acceptance of Status-Realm-Request packets. The default SHOULD be that acceptance is enabled. Servers SHOULD also permit administrators to enable or disable acceptance of Status-Realm-Request packets on a per-RADIUS Client basis. The default SHOULD be that acceptance is enabled.

If a server does not support Status-Realm, or if it is configured not to respond to Status-Realm-Requests, then it MUST silently discard any Status-Realm-Requests messages that it receives. If a server receives a Status-Realm-Request packet from a RADIUS Client from which it is configured not to accept Status-Realm-Requests, then it MUST silently discard the message.

If a server supports Status-Realm, is configured to respond to Status-Realm-Requets, and receives a Status-Realm-Request packet from a permitted RADIUS Client, it MUST first validate the Message-Authenticator attribute as defined in [RFC3579], Section 3.2. Packets failing this validation MUST be silently discarded.

If the Status-Realm-Request passes Message-Authenticator validation, then the server should check if the Target Realm matches a local realm served by this Server. If it does match, the server should send a Status-Realm-Response packet indicating that status of the Target Realm, reachable or unreachable (Status-Server-Response-Code = 0 or 2).

If the Target Realm does not match a local realm, then the server should determine whether it is configured to proxy packets towards the Target Realm. If so, the server should implement the Proxy Server Requirements, below. Servers SHOULD ignore the value of the "user" portion of the User-Name attribute, if any.

Servers SHOULD NOT discard Status-Realm packets merely because they have recently sent the RADIUS Client a response packet. The query may have originated from an administrator who does not have access to the response packet stream or one who is interested in obtaining additional information about the server.

The server MAY decide to send an error response to a Status-Realm-Request packet based on local-site policy. For example, a server that is running but is unable to perform its normal duties SHOULD send a Status-Realm-Response packet indicating an internal error (Status-Server-Response-Code = 257). This situation can happen, for example, when a server requires access to a database for normal operation, but the connection to that database is down. Or, it may happen when the accepted load on the server is lower than the current load.

The server MAY increment packet counters or create log entries as a result of receiving a Status-Realm-Request packet or sending a Status-Realm-Response packet. The server SHOULD NOT perform any other action that is normally performed when it receives a Request packet, other than sending a Response packet.

If the Status-Realm-Request packet includes a Max-Hop-Count attribute, that attribute (with its current value) MUST be returned in any corresponding Status-Realm-Response packet.

Note that [RFC2865], Section 3, defines a number of RADIUS Codes, but does not make statements about which Codes are valid for port 1812. In contrast, [RFC2866], Section 3, specifies that only RADIUS Accounting packets are to be sent to port 1813. This specification is compatible with the standards-track specification [RFC2865], as it defines a new Message Type Code for packets to port 1812. This specification is not compatible with the informational document [RFC2866], as it adds a new Code (Status-Realm-Request) that is valid for port 1813.

9.3. Proxy Server Requirements

Many RADIUS servers act as RADIUS proxies, forwarding requests to other RADIUS servers. Such servers SHOULD proxy Status-Realm-Request packets to enable RADIUS Clients to determine the status of Authentication Realms that are not directly connected to the RADIUS Client.

RADIUS proxies that support Status-Realm-Requests MUST support the Max-Hop-Count attribute defined above. Before forwarding a Status-Realm-Request packet, a proxy MUST check the Max-Hop-Count Attribute. If the Max-Hop-Count attribute is present and the Count is zero (0),

the proxy MUST send a Status-Realm-Response indicating that the hop count has been exceeded (Status-Server-Response-Code = 4), and MUST NOT forward the packet. If the Max-Hop-Count attribute is present, and the Count value is not zero, the proxy MUST decrement the Max-Hop-Count value before forwarding the packet.

The RADIUS proxy MUST check the "realm" portion of the User-Name attribute in the Status-Realm-Request to determine the Target Realm for the request. If the target realm is missing or malformed, the RADIUS proxy MUST send a Status-Realm-Response indicating an invalid realm (Status-Server-Response-Code = 3). If the realm is properly formed, the Status-Realm-Request packet should be proxied toward the Target Realm, using the same next-hop RADIUS server that the proxy server would use for other request packets received on the same port.

In some cases, a RADIUS proxy may not have an available next-hop RADIUS server for the Target Realm. In that case, the RADIUS proxy server MUST send a Status-Realm-Response packet indicating that there is no proxy route to the Target Realm (Status-Server-Response-Code = 1).

In cases where a RADIUS proxy is configured to have a direct connection to the RADIUS server(s) of the Target Realm, but is configured not to forward Status-Realm-Request packets to the target server(s), the proxy MAY use other methods to determine the status of the Target Realm (such as Status-Server packets or recent Access-Request state information), and send a Status-Realm-Response indicating the determined state of the Target Realm (Status-Server-Response-Code = 0 or 2). If the proxy is configured not to forward Status-Realm-Request packet to the Target Realm and does not have other methods to detect the status of the Target Realm, it SHOULD return a Status-Realm-Response packet indicating that the request is administrative prohibited (Status-Server-Response-Code = 257).

If the Status-Realm-Request packet includes a Max-Hop-Count attribute, that attribute (with its current value) MUST be returned in any corresponding Status-Realm-Response packet.

10. Status-Realm Implementation Status

There is an initial implementation of Status-Realm available here:

https://github.com/alandekok/freeradius-server/tree/Status-Realm

10.1. Status-Realm Message Exchange Examples

Message exchange examples are TBD.

11. Proxy Loop Detection Implementation Requirements

This section describes implementation details and requirements for RADIUS Clients, Servers and Proxies that support Proxy Loop Detection.

11.1. Server Requirements

A RADIUS Server that implements Proxy Loop Prevention add its own Server-Information Attribute to any RADIUS message that it generates, including RADIUS Response messages. It MUST also copy all Server-Information atributes from a received RADIUS Request into any RADIUS Response that it generates in reply to that Request.

11.2. Proxy Requirements

A RADIUS Proxy that implements the Loop Prevention mechanism defined in this document MUST be configured with information to populate a Server-Information attribute, and matching criteria to determine if a Server-Information attribute in an incoming request indicates the existence of a Proxy Loop.

Before forwarding a RADIUS Request towards the Target Realm, a RADIUS Proxy that implements Proxy Loop Prevention MUST examine each of the Server-Information attributes included in the Request message to determine whether the message is caught in a Proxy Loop. If so, the Proxy should discard the message. If a Proxy Loop is not detected, the RADIUS Proxy MUST add its own Server-Information attribute to any RADIUS Request that they forward toward the Target Realm.

12. Proxy Loop Detection Implementation Status

The Proxy Loop Detection mechanism is similar to RADIUS Vendor-Specific attribute used today to detect RADIUS Proxy Loops. Unlike the Vendor-Specific attributes in use today, this mechanism includes server information within a single, globally-defrined attribute, rather than requiring that a unique vendor identifiers be allocated for each RADIUS Server operator.

12.1. Loop Detection Message Exchange Examples

Message exchange examples are TBD.

13. Management Information Base (MIB) Considerations

Status-Realm-Request packets are sent to the defined RADIUS ports, so they can affect the [RFC4669] and [RFC4671] RADIUS server MIB modules. [RFC4669] defines a counter named radiusAuthServTotalUnknownTypes that counts the number of RADIUS packets of unknown type that were received. [RFC4671] defines a similar counter named radiusAccServTotalUnknownTypes. Implementations not supporting Status-Realm-Requests or implementations that are configured not to respond to Status-Realm-Request packets MUST use these counters to track received Status-Realm packets.

If, however, Status-Realm-Requests are supported and the server is configured to respond as described above, then the counters defined in [RFC4669] and [RFC4671] MUST NOT be used to track Status-Realm-Request or Status-Realm-Response packets. That is, when a server fully implements Status-Realm, the counters defined in [RFC4669] and [RFC4671] MUST be unaffected by the transmission or reception of packets relating to Status-Realm-Requests.

If a server supports Status-Realm-Request and the [RFC4669] or [RFC4671] MIB modules, then it SHOULD also support vendor-specific MIB extensions dedicated solely to tracking Status-Realm-Request and Status-Realm-Response packets. Any definition of the server MIB modules for Status-Realm-Requests is outside of the scope of this document.

14. Interaction with RADIUS Client MIB Modules

RADIUS Clients implementing Status-Realm-Request MUST NOT increment [RFC4668] or [RFC4670] counters upon reception of Status-Realm-Response packets. That is, when a RADIUS Client fully implements Status-Realm-Request, the counters defined in [RFC4668] and [RFC4670] MUST be unaffected by the transmission or reception of packets relating to Status-Realm.

If an implementation supports Status-Realm-Request and the [RFC4668] or [RFC4670] MIB modules, then it SHOULD also support vendor-specific MIB extensions dedicated solely to tracking Status-Realm requests and responses. Any definition of the RADIUS Client MIB modules for Status-Realm-Requests is outside of the scope of this document.

15. Table of Attributes

The following table provides a guide to which attributes may be found in Status-Realm-Request and Status-Realm-Response packets, and in what quantity. Attributes other than the ones listed below SHOULD NOT be found in a Status-Realm-Request packet.

Status-Realm-Request	Status-Realm-Response		
1	1	1	User-Name
0	0	2	User-Password
0	0	3	CHAP-Password
0-1	0	4	NAS-IP-Address (Note 1)
0	0+	18	Reply-Message
0+	0+	26	Vendor-Specific
0-1	0	32	NAS-Identifier (Note 1)
0	0	79	EAP-Message
1	0-1	80	Message-Authenticator
0-1	0	95	NAS-IPv6-Address (Note 1)
0	1	(TBD)	Status-Realm-Response-Code
1	0	(TBD)	Max-Hop-Count
0+	0+	(TBD)	Server-Information
0	0	103-121	Digest-*

Note 1: Status-Realm-Request packet SHOULD contain one of (NAS-IP-Address or NAS-IPv6-Address), or NAS-Identifier, or both NAS-Identifier and one of (NAS-IP-Address or NAS-IPv6-Address).

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

0	This attribute MUST NOT be present in packet.
0+	Zero or more instances of this attribute MAY be present in the packet.
0-1	Zero or one instance of this attribute MAY be present in the packet.
1	Exactly one instance of this attribute MUST be present in the packet.

16. IANA Considerations

This document defines the Status-Realm-Request (TBD) and the Status-Realm-Response (TBD) RADIUS Packet Type Codes, both of which should be assigned by IANA from the Unassigned block of RADIUS Packet Type Codes.

This document defines three new RADIUS attributes, Max-Hop-Count (TBD) and Status-Realm-Response-Code (TBD) and Server-Identifier (TBD), which should be assigned by IANA from an Unassigned block of RADIUS Attribute Types, such as the Unassigned block for Extended-Attribute-1.

This document also defines two new Protocol Registries that need to be created: "Values for RADIUS Attribute (TBD), Status-Realm-Response-Code" and "Valies for RADIUS Attribute (TBD), Server-Identifier". Initial values for these registries are defined above.

17. Security Considerations

Status-Realm-Request packets are similar to Access-Request packets, and are therefore subject to the same security considerations as described in [RFC2865], Section 8. Status-Realm packets also use the Message-Authenticator attribute, and are therefore subject to the same security considerations as [RFC3579], Section 4.

We reiterate that all Status-Realm-Request packets MUST contain a Message-Authenticator. Servers not checking the Message-Authenticator attribute could respond to Status-Realm packets from an attacker, potentially enabling a reflected DoS attack onto a real RADIUS Client.

Where this document differs from [RFC2865] is that it defines a new request/response method in RADIUS: the Status-Realm-Request and Status-Realm-Response. The Status-Realm-Request is similar to the previously described and widely implemented Status-Server message [RFC5997], and no additional security considerations are known to relate to the implementation or use of Status-Server. This option differs from Status-Server because it is forwarded through proxies, so it can be sent to a RADIUS Server that does not have a direct connection to the Status-Realm RADIUS Client. However, Access-Request packets are also forwarded, and there should be no additional attacks other than those incurred by forwarding Status-Realm-Request packets.

Attacks on cryptographic hashes are well known [RFC4270] and getting better with time. RADIUS uses the MD5 hash [RFC1321] for packet authentication and attribute obfuscation. There are ongoing efforts in the IETF to analyze and address these issues for the RADIUS protocol.

Security Considerations for Loop Prevention are TBD.

18. Acknowledgements

This document was written using xml2rfc, as described in [RFC7991]

Some of the sections in this document were adapted from the description of the Status-Server RADIUS Packet Type Code in [RFC5997].

19. References

19.1. 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>.

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

19.2. Informative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC2869] Rigney, C., Willats, W., and P. Calhoun, "RADIUS Extensions", RFC 2869, DOI 10.17487/RFC2869, June 2000, <https://www.rfc-editor.org/info/rfc2869>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <https://www.rfc-editor.org/info/rfc3579>.

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, DOI 10.17487/RFC4270, November 2005, <https://www.rfc-editor.org/info/rfc4270>.

[RFC4668] Nelson, D., "RADIUS Authentication Client MIB for IPv6", RFC 4668, DOI 10.17487/RFC4668, August 2006, <https://www.rfc-editor.org/info/rfc4668>.

[RFC4669] Nelson, D., "RADIUS Authentication Server MIB for IPv6", RFC 4669, DOI 10.17487/RFC4669, August 2006, <https://www.rfc-editor.org/info/rfc4669>.

[RFC4670] Nelson, D., "RADIUS Accounting Client MIB for IPv6", RFC 4670, DOI 10.17487/RFC4670, August 2006, <https://www.rfc-editor.org/info/rfc4670>.

[RFC4671] Nelson, D., "RADIUS Accounting Server MIB for IPv6", RFC 4671, DOI 10.17487/RFC4671, August 2006, <https://www.rfc-editor.org/info/rfc4671>.

[RFC5580] Tschofenig, H., Ed., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, DOI 10.17487/RFC5580, August 2009, <https://www.rfc-editor.org/info/rfc5580>.

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, DOI 10.17487/RFC5997, August 2010, <https://www.rfc-editor.org/info/rfc5997>.

[RFC7991] Hoffman, P., "The "xml2rfc" Version 3 Vocabulary", RFC 7991, DOI 10.17487/RFC7991, December 2016, <https://www.rfc-editor.org/info/rfc7991>.

Authors' Addresses

Margaret Cullen
Painless Security
Phone: +1 (781)405-7464
Email: margaret@painless-security.com

Alan DeKok
FreeRADIUS
Email: aland@freeradius.org

Mark Donnelly
Painless Security
Phone: +1 (857)928-5967
Email: mark@painless-security.com

Josh Howlett
Federated Solutions
Phone: +44 (0)7510 666 950
Email: josh@federated-solutions.com

RADEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2024

A. DeKok
FreeRADIUS

23 October 2023

Deprecating Insecure Practices in RADIUS
draft-dekok-radext-deprecating-radius-05

Abstract

RADIUS crypto-agility was first mandated as future work by RFC 6421. The outcome of that work was the publication of RADIUS over TLS (RFC 6614) and RADIUS over DTLS (RFC 7360) as experimental documents. Those transport protocols have been in wide-spread use for many years in a wide range of networks. They have proven their utility as replacements for the previous UDP (RFC 2865) and TCP (RFC 6613) transports. With that knowledge, the continued use of insecure transports for RADIUS has serious and negative implications for privacy and security.

This document formally deprecates using the User Datagram Protocol (UDP) and of the Transmission Control Protocol (TCP) as transport protocols for RADIUS. These transports are permitted inside of secure networks, but their use in secure networks is still discouraged. For all other environments, the use of secure transports such as IPsec or TLS is mandated. We also discuss additional security issues with RADIUS deployments, and give recommendations for practices which increase security and privacy.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekok-radext-deprecating-radius/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/deprecating-radius.git.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The RADIUS protocol [RFC2865] was first standardized in 1997, though its roots go back much earlier to 1993. The protocol uses MD5 [RFC1321] to sign some packets types, and to obfuscate certain attributes such as User-Password. As originally designed, Access-Request packets were entirely unauthenticated, and could be trivially spoofed as discussed in [RFC3579] Section 4.3.2. In order to prevent such spoofing, that specification defined the Message-Authenticator attribute ([RFC3579] Section 3.2) which allowed for packets to carry a signature based on HMAC-MD5.

The state of MD5 security was discussed in [RFC6151], which led to the state of RADIUS security being reviewed in [RFC6421] Section 3. The outcome of that review was the remainder of [RFC6421], which created crypto-agility requirements for RADIUS.

RADIUS was historically secured with IPSec, as described in [RFC3579] Section 4.2:

> To address the security vulnerabilities of RADIUS/EAP, implementations of this specification SHOULD support IPsec (RFC2401) along with IKE (RFC2409) for key management. IPsec ESP (RFC2406) with non-null transform SHOULD be supported, and IPsec ESP with a non-null encryption transform and authentication support SHOULD be used to provide per-packet confidentiality, authentication, integrity and replay protection. IKE SHOULD be used for key management.

The use of IPSec allowed RADIUS to be sent privately, and securely, across the Internet. However, experience showed that TLS was in many ways simpler for implementations and deployment than IPSec. While IPSec required operating system support, TLS was an application-space library. This difference, coupled with the wide-spread adoption of TLS for HTTPS ensures that it was often easier for applications to use TLS than IPSec.

RADIUS/TLS [RFC6614] and RADIUS/DTLS [RFC7360] were then defined in order to meet the crypto-agility requirements of [RFC6421]. RADIUS/TLS has been in wide-spread use for about a decade, including eduroam [EDUROAM], and more recently OpenRoaming [OPENROAMING] and [I-D.tomas-openroaming]. RADIUS/DTLS has seen less use across the public Internet, but it nonetheless has multiple implementations.

As of the writing of this specification, RADIUS/UDP is still widely used, even though it depends on MD5 and "ad hoc" constructions for security. While MD5 has been broken, it is a testament to the design of RADIUS that there have been (as yet) no attacks on RADIUS Authenticator signatures which are stronger than brute-force.

However, the problems with MD5 means that if a someone can view unencrypted RADIUS traffic, even a hobbyist attacker can crack all possible RADIUS shared secrets of eight characters or less. Such attacks can also result in compromise of all passwords carried in the User-Password attribute.

Even if a stronger packet signature method was used as in [RFC6218], it would not fully address the issues with RADIUS. Most information in RADIUS is sent in clear-text, and only a few attributes are hidden via obfuscation methods which rely on more "ad hoc" MD5 constructions. The privacy implications of this openness are severe.

Any observer of non-TLS RADIUS traffic is able to obtain a substantial amount of personal identifiable information (PII) about users. The observer can tell who is logging in to the network, what devices they are using, where they are logging in from, and their approximate location (usually city). With location-based attributes as defined in [RFC5580], a users location may be determined to within

15 or so meters outdoors, and with "meter-level accuracy indoors" [WIFILOC]. An observer can also use RADIUS accounting packets to determine how long a user is online, and to track a summary of their total traffic (upload and download totals).

When RADIUS/UDP is used across the public Internet, the location of individuals can potentially be tracked in real-time (usually 10 minute intervals), to within 15 meters. Their devices can be identified, and tracked. Any passwords they send via the User-Password attribute can be be compromised. Even using CHAP-Password offers minimal protection, as the cost of cracking the underlying password is similar to the cost of cracking the shared secret. MS-CHAP ([RFC2433] and [RFC2759]) is significantly worse for security, as it can be trivially cracked with minimal resources even if the shared secret is not known (Section 7.4).

The implications for security and individual safety are large, and negative.

These issues are only partly mitigated when the authentication methods carried within RADIUS define their own processes for increased security and privacy. For example, some authentication methods such EAP-TLS, EAP-TTLS, etc. allow for User-Name privacy and for more secure transport of passwords via the use of TLS. The use of MAC address randomization can limit device information identification to a particular manufacturer, instead of to a unique device.

However, these authentication methods are not always used, or are not always available. Even if these methods were used ubiquitously, they do not protect all of the information which is publicly available over RADIUS/UDP or RADIUS/TCP transports. And even when TLS-based EAP methods are used, implementations have historically often skipped certificate validation, leading to password compromise ([SPOOFING]). In many cases, users were not even aware that the server certificate was incorrect or spoofed, which meant that there was no way for the user to detect that anything was wrong. Their passwords were simply handed to a spoofed server, with little possibility for the user to take any action to stop it.

It is no longer acceptable for RADIUS to rely on MD5 for security. It is no longer acceptable to send device or location information in clear text across the wider INternet. This document therefore deprecates insecure uses of RADIUS, and mandates the use of secure TLS-based transport layers. We also discuss related security issues with RADIUS, and give many recommendations for practices which increase security and privacy.

1.1. Simply using IPSec or TLS is not enough

The use of a secure transport such as IPSec or TLS ensures complete privacy and security for all RADIUS traffic. An observer is limited to knowing rough activity levels of a client or server. That is, an observer can tell if there are a few users on a NAS, or many users on a NAS. All other information is hidden from all observers. However, it is not enough to say "use IPSec" and then move on to other issues. There are many issues which can only be addressed via an informed approach.

For example it is possible for an attacker to record the session traffic, and later crack the TLS session key or IPSec parameters. This attack could comprise all traffic sent over that connection, including EAP session keys. If the cryptographic methods provide forward secrecy ([RFC7525] Section 6.3), then breaking one session provides no information about other sessions. As such, it is RECOMMENDED that all cryptographic methods used to secure RADIUS conversations provide forward secrecy. While forward secrecy will not protect individual sessions from attack, it will prevent attack on one session from being leveraged to attack other, unrelated, sessions.

AAA servers should minimize the impact of such attacks by using a total throughput (recommended) or time based limit before replacing the session keys. The session keys can be replaced though a process of either rekeying the existing connection, or by opening a new connection and deprecating the use of the original connection. Note that if the original connection if closed before a new connection is open, it can cause spurious errors in a proxy environment.

The final attack possible in a AAA system is where one party in a AAA conversation is compromised or run by a malicious party. This attack is made more likely by the extensive use of RADIUS proxy forwarding chains. In that situation, every RADIUS proxy has full visibility into, and control over, the traffic it transports. The solution here is to minimize the number of proxies involved, such as by using Dynamic Peer Discovery ([RFC7585].

There are many additional issues on top of simply adding a secure transport. The rest of this document addresses those issues in detail.

1.2. Overview

The rest of this document begins a summary of issues with RADIUS, and shows just how trivial it is to crack RADIUS/UDP security. We then mandate the use of secure transport, and describe what that requirement means in practice. We give recommendations on how current systems can be migrated to using TLS. We give suggestions for increasing the security of existing RADIUS transports, including a discussion of the authentication protocols carried within RADIUS. We conclude with privacy and security considerations.

As IPSec has been discussed previously in the context of RADIUS, we do not discuss it in detail to it here, other than to say it is an acceptable solution for securing RADIUS traffic. As the bulk of the current efforts are focused on TLS, this document likewise focuses on TLS. However, all of the issues raised here about the RADIUS protocol also apply to IPSec transport.

While this document tries to be comprehensive, it is necessarily imperfect. There may be issues which should have been included, but which were missed due to oversight or accident. Any reader should be aware that there are good practices which are perhaps not documented here, and bad behaviors which are likewise not forbidden.

There is also a common tendency to suggest that a particular practice is "allowed" by a specification, simply because the specification does not forbid that practice. This belief is wrong. That is, a behavior which is not mentioned in the specification cannot honestly be said to be "permitted" or "allowed" by that specification. Instead, the correct description for such behaviors is that they are not forbidden. In many cases, documents such as [RFC5080] are written to both correct errors in earlier documents, and to address harmful behaviors have been seen in practice.

By their very nature, documents include a small number of permitted, required, and/or forbidden behaviors. There are a much larger set of behaviors which are undefined. That is, behaviors which are neither permitted nor forbidden. Those behaviors may be good or bad, independent of what the specification says.

Outside of published specifications, there is also a large set of common practices and behaviors which have grown organically over time, but which have not been written into a specification. These practices have been found to be valuable by implementers and administrators. Deviations from these practices generally result in instabilities and incompatibilities between systems. As a result, implementers should exercise caution when creating new behaviors which have not previously been seen in the industry. Such behaviors are likely to be wrong.

It is RECOMMENDED that implementations follow widely accepted practices which have been proven to work, even if those practices are not written down in a public specification. Failure to follow common industry practices usually results in interoperability failures.

2. Terminology

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.

* RADIUS

  The Remote Authentication Dial-In User Service protocol, as defined in [RFC2865], [RFC2865], and [RFC5176] among others.

* RADIUS/UDP

  RADIUS over the User Datagram Protocol as define above.

* RADIUS/TCP

  RADIUS over the Transport Control Protocol [RFC6613]

* RADIUS/TLS

  RADIUS over the Transport Layer Security protocol [RFC6614]

* RADIUS/DTLS

  RADIUS over the Datagram Transport Layer Security protocol [RFC7360]

* TLS

the Transport Layer Security protocol. Generally when we refer to TLS in this document, we are referring to RADIUS/TLS and/or RADIUS/DTLS.

* NAS

  Network Access Server, which is a RADIUS client.

3. Overview of issues with RADIUS

There are a large number of issues with RADIUS. The most serious is that RADIUS sends most information "in the clear", with obvious privacy implications.

Further, MD5 has been broken for over a decade, as summarized in [RFC6151]. For traffic sent across the Internet, no protocol should depend on MD5 for security. Even if MD5 was not broken, computers have gotten substantially faster in the past thirty years. This speed increase makes it possible for the average hobbyist to perform brute-force attacks to crack even seemingly complex shared secrets.

We address each of these issues in detail below.

3.1. Information is sent in Clear Text

Other than a few attributes such as User-Password, all RADIUS traffic is sent "in the clear". The resulting data exposure has a large number of privacy issues. We refer to [RFC6973], and specifically to Section 5 of that document for detailed discussion. RADIUS/UDP and RADIUS/TCP are vulnerable to all of the issues raised by [RFC6973].

There are clear privacy and security information with sending user identifiers, and user locations [RFC5580] in clear-text across the Internet. As such, the use of clear-text protocols across insecure networks is no longer acceptable.

3.2. MD5 has been broken

Attacks on MD5 are summarized in part in [RFC6151]. While there have not been many new attacks in the decade since [RFC6151] was published, that does not mean that further attacks do not exist. It is more likely that no one is looking for new attacks.

It is reasonable to expect that new research can further break MD5, but also that such research may not be publicly available.

3.3. Complexity of cracking RADIUS shared secrets

The cost of cracking a a shared secret can only go down over time as computation becomes cheaper. The issue is made worse because of the way MD5 is used to sign RADIUS packets. The attacker does not have to calculate the hash over the entire packet, as the hash prefix can be calculated once, and then cached. The attacker can then begin the attack with that hash prefix, and brute-force only the shared secret portion.

At the time of writing this document, an "off the shelf" commodity computer can calculate at least 100M MD5 hashes per second. If we limit shared secrets to upper/lowercase letters, numbers, and a few "special" characters, we have 64 possible characters for shared secrets. Which means that for 8-character secrets, there are 2^48 possible combinations.

The result is that using consumer-grade machine, it takes approximately 32 days to brute-force the entire 8 octet / 64 character space for shared secrets. The problem is even worse when graphical processing units (GPUs) are used. A high-end GPU is capable of performing more than 64 billion hashes per second. At that rate, the entire 8 character space described above can be searched in approximately 90 minutes.

This is an attack which is feasible today for a hobbyist. Increasing the size of the character set raises the cost of cracking, but not enough to be secure. Increasing the character set to 93 characters means that the hobbyist using a GPU could search the entire 8 character space in about a day.

Increasing the length of the shared secret has a larger impact on the cost of cracking. For secrets ten characters long, one GPU can search a 64-character space in about six months, and a 93 character space would take approximately 24 years.

This brute-force attack is also trivially parallelizable. Nation-states have sufficient resources to deploy hundreds to thousands of systems dedicated to these attacks. That realization means that a "time to crack" of 24 years is simply expensive, but is not particularly difficult. A thousand commodity CPUs are enough to reduce the crack time from 24 years to a little over a week.

Whether the above numbers are precise, or only approximate is immaterial. These attacks will only get better over time. The cost to crack shared secrets will only go down over time.

Even worse, administrators do not always derive shared secrets from secure sources of random numbers. The "time to crack" numbers given above are the absolute best case, assuming administrators follow best practices for creating secure shared secrets. For shared secrets created manually by a person, the search space is orders of magnitude smaller than the best case outlined above. Rather than brute-forcing all possible shared secrets, an attacker can create a local dictionary which contains common or expected values for the shared secret. Where the shared secret used by an administrator is in the dictionary, the cost of the attack can drop by multiple orders of magnitude.

It should be assumed that a hobbyist attacker with modest resource can crack most shared secrets created by people in minutes, if not seconds.

Despite the ease of attacking MD5, it is still a common practice for some "cloud" and other RADIUS providers to send RADIUS/UDP packets over the Internet "in the clear". It is also common practice for administrators to use "short" shared secrets, and to use shared secrets created by a person, or derived from a limited character set. Theses practice are easy to implement and follow, but they are highly insecure and SHOULD NOT be used.

Further requirements in shared secrets are given below in Section 6.1.

3.4. Tunnel-Password and CoA-Request packets

There are a number of security problems with the Tunnel-Password attribute, at least in CoA-Request and Disconnect-Request packets. A full explanation requires a review of the relevant specifications.

[RFC5176] Section 2.3 describes how to calculate the Request Authenticator field for these packets:

Request Authenticator

> In Request packets, the Authenticator value is a 16-octet MD5 [RFC1321] checksum, called the Request Authenticator. The Request Authenticator is calculated the same way as for an Accounting-Request, specified in [RFC2866].

Where [RFC2866] Section 3 says:

> The NAS and RADIUS accounting server share a secret. The Request Authenticator field in Accounting-Request packets contains a one-way MD5 hash calculated over a stream of octets consisting of the Code + Identifier + Length + 16 zero octets + request attributes + shared secret (where + indicates concatenation). The 16 octet MD5 hash value is stored in the Authenticator field of the Accounting-Request packet.

Taken together, these definitions mean that for CoA-Request packets, all attribute obfuscation is calculated with the Reply Authenticator being all zeroes. In contrast for Access-Request packets, the Request Authenticator is mandated there to be 16 octets of random data. This difference has negative impacts on security.

For Tunnel-Password, [RFC5176] Section 3.6 allows it to appear in CoA-Request packets:

> ...
> Change-of-Authorization Messages
>
> Request ACK NAK # Attribute
> ...
> 0+ 0 0 69 Tunnel-Password (Note 5)
> ...
> (Note 5) When included within a CoA-Request, these attributes represent an authorization change request. Where tunnel attributes are included within a successful CoA-Request, all existing tunnel attributes are removed and replaced by the new attribute(s).

However, [RFC2868] Section 3.5 says that Tunnel-Password is encrypted with the Request Authenticator:

> Call the shared secret S, the pseudo-random 128-bit Request Authenticator (from the corresponding Access-Request packet) R,

The assumption that the Request Authenticator is random data is true for Access-Request packets. That assumption is not true for CoA-Request packets.

That is, when the Tunnel-Password attribute is used in CoA-Request packets, the only source of randomness in the obfuscation is the salt, as defined in [RFC2868] Section 3.5;

Salt
: The Salt field is two octets in length and is used to ensure the uniqueness of the encryption key used to encrypt each instance of the Tunnel-Password attribute occurring in a given Access-Accept packet. The most significant bit (leftmost) of the Salt field MUST be set (1). The contents of each Salt field in a given Access-Accept packet MUST be unique.

This chain of unfortunate definitions means that there is only 15 bits of entropy in the Tunnel-Password obfuscation (plus the secret). It is not known if this limitation makes it sufficiently easy for an attacker to determine the contents of the Tunnel-Password. However, such limited entropy cannot be a good thing, and it is one more reason to deprecate RADIUS/UDP.

Due to the above issues, implementations and new specifications SHOULD NOT permit obfuscated attributes to be used in CoA-Request or Disconnect-Request packets.

4. All short Shared Secrets have been compromised

Unless RADIUS packets are sent over a secure network (IPsec, TLS, etc.), administrators SHOULD assume that any shared secret of 8 characters or less has been immediately compromised. Administrators SHOULD assume that any shared secret of 10 characters or less has been compromised by an attacker with significant resources. Administrators SHOULD also assume that any private information (such as User-Password) which depends on such shared secrets has also been compromised.

In conclusion, if a User-Password, or CHAP-Password, or MS-CHAP password has been sent over the Internet via RADIUS/UDP or RADIUS/TCP in the last decade, you should assume that underlying password has been compromised.

5. Deprecating Insecure transports

The solution to an insecure protocol which uses thirty year-old cryptography is to deprecate the use insecure cryptography, and to mandate modern cryptographic transport.

5.1. Deprecating UDP and TCP as transports

RADIUS/UDP and RADIUS/TCP MUST NOT be used outside of secure networks. A secure network is one which is known to be safe from eavesdroppers, attackers, etc. For example, if IPsec is used between two systems, then those systems may use RADIUS/UDP or RADIUS/TCP over the IPsec connection.

Similarly, RADIUS/UDP and RADIUS/TCP could be used in secure management networks. However, administrators should not assume that such uses are always secure. An attacker who breaks into a key system could use that access to view RADIUS traffic, and thus be able to attack it. Similarly, a network misconfiguration could result in the RADIUS traffic being sent over an insecure network.

Neither the RADIUS client nor the RADIUS server would be aware of any network misconfiguration (e.g. such as could happen with IPSec). Neither the RADIUS client nor the RADIUS server would be aware of any attacker snooping on RADIUS/UDP or RADIUS/TCP traffic.

In contrast, when TLS is used, the RADIUS endpoints are aware of all security issues, and can enforce any necessary security policies.

Using RADIUS/UDP and RADIUS/TCP in any environment is therefore NOT RECOMMENDED.

5.2. Mandating Secure transports

All systems sending RADIUS packets outside of secure networks MUST use either IPSec, RADIUS/TLS, or RADIUS/DTLS. It is RECOMMENDED, for operational and security reasons that RADIUS/TLS or RADIUS/DTLS are preferred over IPSec.

Unlike (D)TLS, use of IPSec means that applications are generally unaware of transport-layer security. Any problem with IPSec such as configuration issues, negotiation or re-keying problems are typically presented to the RADIUS servers as 100% packet loss. These issues may occur at any time, independent of any changes to a RADIUS application using that transport. Further, network misconfigurations which remove all security are completely transparent to the RADIUS application: packets can be sent over an insecure link, and the RADIUS server is unaware of the failure of the security layer.

In contrast, (D)TLS gives the RADIUS application completely knowledge and control over transport-layer security. The failure cases around (D)TLS are therefore often clearer, easier to diagnose and faster to resolve than failures in IPSec. For example, a failed TLS connection may return a "connection refused" error to the application, or any one of many TLS errors indicating which exact part of the TLS conversion failed during negotiation.

5.3. Crypto-Agility

The crypto-agility requirements of [RFC6421] are addressed in [RFC6614] Appendix C, and in Section 10.1 of [RFC7360]. For clarity, we repeat the text of [RFC7360] here, with some minor modifications to update references, but not content.

Section 4.2 of [RFC6421] makes a number of recommendations about security properties of new RADIUS proposals. All of those recommendations are satisfied by using TLS or DTLS as the transport layer.

Section 4.3 of [RFC6421] makes a number of recommendations about backwards compatibility with RADIUS. [RFC7360] Section 3 addresses these concerns in detail.

Section 4.4 of [RFC6421] recommends that change control be ceded to the IETF, and that interoperability is possible. Both requirements are satisfied.

Section 4.5 of [RFC6421] requires that the new security methods apply to all packet types. This requirement is satisfied by allowing TLS and DTLS to be used for all RADIUS traffic. In addition, [RFC7360] Section 3, addresses concerns about documenting the transition from legacy RADIUS to crypto-agile RADIUS.

Section 4.6 of [RFC6421] requires automated key management. This requirement is satisfied by using TLS or DTLS key management.

We can now finalize the work began in [RFC6421]. This document updates [RFC2865] et al. to state that any new RADIUS specification MUST NOT introduce new "ad hoc" cryptographic primitives to sign packets as was done with the Request / Response Authenticator, or to obfuscate attributes as was done with User-Password and Tunnel-Password. That is, RADIUS-specific cryptographic methods existing as of the publication of this document can continue to be used for historical compatibility. However, all new cryptographic work in the RADIUS protocol is forbidden.

We recognize that RADIUS/UDP will still be in use for many years, and that new standards may require some modicum of privacy. As a result, it is a difficult choice to forbid the use of these constructs. If an attack is discovered which breaks RADIUS/UDP (e.g. by allowing attackers to forge Request Authenticators or Response Authenticators, or by allowing attackers to de-obfuscate User-Password), the solution would be to simply deprecate the use of RADIUS/UDP entirely. It would not be acceptable to design new cryptographic primitives in an attempt to "secure" RADIUS/UDP.

All new security and privacy requirements in RADIUS MUST be provided by a secure transport layer such as TLS or IPSec. As noted above, simply using IPsec is not always enough, as the use (or not) of IPsec is unknown to the RADIUS application.

The restriction forbidding new cryptographic work in RADIUS does not apply to the data being transported in RADIUS attributes. For example, a new authentication protocol could use new cryptographic methods, and would be permitted to be transported in RADIUS. This protocol could be a new EAP method, or it could use updates to TLS. In those cases, RADIUS serves as a transport layer for the authentication method. The authentication data is treated as opaque data for the purposes of Access-Request, Access-Challenge, etc. packets. There would be no need for RADIUS to define any new cryptographic methods in order to transport this data.

Similarly, new specifications MAY define new attributes which use the obfuscation methods for User-Password as defined in [RFC2865] Section 5.2, or for Tunnel-Password as defined in [RFC2868] Section 3.5. However, due to the issues noted above in Section 3.4, the Tunnel-Password obfuscation method MUST NOT be used for packets other than Access-Request, Access-Challenge, and Access-Accept. If the attribute needs to be send in another type of packet, then the protocol design is likely wrong, and needs to be revisited. It is again a difficult choice to forbid certain uses of the Tunnel-Password obfuscation method, but we believe that doing so is preferable to allowing sensitive data to be obfuscated with less security than the original design intent.

6. Migration Path and Recommendations

We recognize that it is difficult to upgrade legacy devices with new cryptographic protocols and user interfaces. The problem is made worse because the volume of RADIUS devices which are in use. The exact number is unknown, and can only be approximated. Our best guess is that at the time of this writing, there could be in the order of hundreds of thousands, if not millions of RADIUS/UDP devices in daily use.

We therefore need to define a migration path to using secure transports. We give a a number of migration steps which could be done independently. We recommend increased entropy for shared secrets. We also mandate the use of Message-Authenticator in all Access-Request packets for RADIUS/UDP and RADIUS/TCP. Finally, where [RFC6614] Section 2.3 makes support for TLS-PSK optional, we suggest that RADIUS/TLS and RADIUS/DTLS implementations SHOULD support TLS-PSK.

6.1. Shared Secrets

[RFC2865] Section 3 says:

> It is preferred that the secret be at least 16 octets. This is to ensure a sufficiently large range for the secret to provide protection against exhaustive search attacks. The secret MUST NOT be empty (length 0) since this would allow packets to be trivially forged.

This recommendation is no longer adequate, so we strengthen it here.

RADIUS implementations MUST support shared secrets of at least 32 octets, and SHOULD support shared secrets of 64 octets. Implementations MUST warn administrators that the shared secret is insecure if it is 10 octets or less in length.

Administrators SHOULD use shared secrets of at least 24 octets, generated using a source of secure random numbers. Any other practice is likely to lead to compromise of the shared secret, user information, and possibly of the entire network.

Creating secure shared secrets is not difficult. One solution is to use a simple script given below. While the script is not portable to all possible systems, the intent here is to document a concise and simple method for creating secrets which are secure, and humanly manageable.

```
#!/usr/bin/env perl use MIME::Base32; use Crypt::URandom(); print
join('-', unpack("(A4)*", lc
encode_base32(Crypt::URandom::urandom(12)))), "\n";
```

This script reads 96 bits of random data from a secure source, encodes it in Base32, and then makes it easier for people to work with. The generated secrets are of the form "2nw2-4cfi-nicw-3g2i-5vxq". This form of secret will be accepted by all implementation which supports at least 24 octets for shared secrets.

Given the simplicity of creating strong secrets, there is no excuse for using weak shared secrets with RADIUS. The management overhead of dealing with complex secrets is less than the management overhead of dealing with compromised networks.

Over all, the security analysis of shared secrets is similar to that for TLS-PSK. It is therefore RECOMMENDED that implementors manage shared secrets with same the practices which are recommended for TLS-PSK, as defined in [RFC8446] Section E.7 and [RFC9257] Section 4.

On a practical node, RADIUS implementers SHOULD provide tools for administrators which can create and manage secure shared secrets. The cost to do so is minimal for implementors. Providing such a tool can further enable and motivate administrators to use secure practices.

6.2. Message-Authenticator

The Message-Authenticator attribute was defined in [RFC3579] Section 3.2. The "Note 1" paragraph at the bottom of [RFC3579] Section 3.2 required that Message-Authenticator be added to Access-Request packets when the EAP-Message as present, and suggested that it should be present in a few other situations. Experience has shown that these recommendations are inadequate.

Some RADIUS clients never use the Message-Authenticator attribute, even for the situations where the [RFC3579] text suggests that it should be used. When the Message-Authenticator attribute is missing from Access-Request packets, it is often possible to trivially forge or replay those packets.

For example, an Access-Request packet containing CHAP-Password but which is missing Message-Authenticator can be trivially forged. If an attacker sees one packet such packet, it is possible to replace the CHAP-Password and CHAP-Challenge (or Request Authenticator) with values chosen by the attacker. The attacker can then perform brute-force attacks on the RADIUS server in order to test passwords.

This document therefore requires that RADIUS clients MUST include the Message-Authenticator in all Access-Request packets when UDP or TCP transport is used.

In contrast, when TLS-based transports are used, the Message-Authenticator attribute serves no purpose, and can be omitted, even when the Access-Request packet contains an EAP-Message attribute. Servers receiving Access-Request packets over TLS-based transports SHOULD NOT silently discard a packet if it is missing a Message-Authenticator attribute. However, if the Message-Authenticator attribute is present, it still MUST be validated as discussed in [RFC7360] and [RFC3579].

6.3. Recommending TLS-PSK

Given the insecurity of RADIUS/UDP, the absolute minimum acceptable security is to use strong shared secrets. However, administrator overhead for TLS-PSK is not substantially higher than for shared secrets, and TLS-PSK offers significantly increased security and privacy.

It is therefore RECOMMENDED that implementations support TLS-PSK. In some cases TLS-PSK is preferable to certificates. It may be difficult for RADIUS clients to upgrade all of their interfaces to support the use of certificates, and TLS-PSK more closely mirrors the historical use of shared secrets, with similar operational considerations.

Implementation and operational considerations for TLS-PSK are given in [I-D.ietf-radext-tls-psk], and we do not repeat them here.

7. Increasing the Security of RADIUS

While we still permit the use of UDP and TCP transports in secure environments, there are opportunities for increasing the security of RADIUS when those transport protocols are used. The amount of personal identifiable information sent in packets should be minimized. Information about the size, structure, and nature of the visited network should be omitted or anonymized. The choice of authentication method also has security and privacy impacts.

The recommendations here for increasing the security of RADIUS transports also applies when TLS is used. TLS transports protect the RADIUS packets from observation by from third-parties. However, TLS does not hide the content of RADIUS packets from intermediate proxies, such as ones uses in a roaming environment. As such, the best approach to minimizing the information sent to proxies is to minimize the number of proxies which see the RADIUS traffic.

Implementers and administrators need to be aware of all of these issues, and then make the best choice for their local network which balances their requirements on privacy, security, and cost. Any security approach based on a simple "checklist" of "good / bad" practices is likely to result in decreased security, as compared to an end-to-end approach which is based on understanding the issues involved.

7.1. Minimizing Personal Identifiable Information

One approach to increasing RADIUS privacy is to minimize the amount of PII which is sent in packets. Implementers of RADIUS products and administrators of RADIUS systems SHOULD ensure that only the minimum necessary PII is sent in RADIUS.

Where possible, identities should be anonymized (e.g. [RFC7542] Section 2.4). The use of anonymized identities means that the the Chargeable-User-Identifier [RFC4372] should also be used. Further discussion on this topic is below.

Device information SHOULD be either omitted, or randomized. e.g. MAC address randomization could be used on end-user devices. The details behind this recommendation are the subject of ongoing research and development. As such, we do not offer more specific recommendations here.

Information about the visited network SHOULD be replaced or anonymized before packets are proxied outside of the local organization. The attribute Operator-NAS-Identifier [RFC8559] can be used to anonymize information about NASes in the local network.

Location information ([RFC5580] SHOULD either be omitted, or else it SHOULD be limited to the broadest possible information, such as country code. For example, [I-D.tomas-openroaming] says:

> All OpenRoaming ANPs MUST support signalling of location information

This location information is required to include at the minimum the country code. We suggest the country code SHOULD also be the maximum amount of location information which is sent over third-party networks.

7.1.1. Chargeable-User-Identity

Where the Chargeable-User-Identity (CUI) [RFC4372] is used, it SHOULD be unique per session. This practice will help to maximize user privacy, as it will be more difficult to track users across multiple sessions. Due to additional constraints which we will discuss below, we cannot require that the CUI change for every session.

What we can do is to require that the home server MUST provide a unique CUI for each combination of user and visited network. That is, if the same user visits multiple networks, the home server MUST provide different CUIs to each visited network for that user. The CUI MAY be the same across multiple sessions for that user on one particular network. The CUI MAY be the same for multiple devices used by that user on one particular network.

We note that the MAC address is likely the same across multiple user sessions on one network. Therefore changing the CUI offers little additional benefit, as the user can still be tracked by the unchanging MAC address. Never the less, we believe that having a unique CUI per session can be useful, because there is ongoing work on increasing user privacy by allowing more MAC address randomization. If we were to recommend that the CUI remain constant across multiple sessions, that would in turn negate much of the effort being put into MAC address randomization.

One reason to have a constant CUI value for a user (or user devices) on one network is that network access providers may need to enforce limits on simultaneous logins. Network providers may also need to correlate user behavior across multiple sessions in order to track and prevent abuse. Both of these requirements are impossible if the CUI changes for every user session.

The result is that there is a trade-off between user privacy and the needs of the local network. While perfect user privacy is an admirable goal, perfect user privacy may also allow anonymous users to abuse the visited network. The network would then likely simply refuse to provide network access. Users may therefore have to accept some limitations on privacy, in order to obtain network access.

We spend some time here in order to give recommendations for creating and managing of CUI. We believe that these recommendations will help implementers satisfy the preceding requirements, while not imposing undue burden on the implementations.

In general, the simplest way to track CUIs long term is to associate the CUI to user identity in some kind of cache or database. This association could be created at the tail end of the authentication process, and before any accounting packets were received. This association should generally be discarded after a period of time if no accounting packets are received. If accounting packets are received, the CUI to user association should then be tracked along with the normal accounting data.

The above method for tracking CUI works no matter how the CUI is generated. If the CUI can be unique per session, or it could be tied to a particular user identity across a long period of time. The same CUI could also be associated with multiple devices.

Where the CUI is not unique for each session, the only minor issue is the cost of the above method is that the association is stored on a per-session basis when there is no need for that to be done. Storing the CUI per session means that is it possible to arbitrarily change how the CUI is calculated, with no impact on anything else in the system. Designs such as this which decouple unrelated architectural elements are generally worth the minor extra cost.

For creating the CUI, that process should be done in a way which is scalable and efficient. For a unique CUI per user, implementers SHOULD create a value which is unique both to the user, and to the visited network. There is no reason to use the same CUI for multiple visited networks, as that would enable the tracking of a user across multiple networks.

Before suggesting a method for creating the CUI, we note that [RFC4372] Section 2.1 defines the CUI as being of data type 'string' ([RFC8044] Section 3.5). [RFC4372] Section 2.1 further suggests that the value of the CUI is interpreted as an opaque token, similar to the Class attribute ([RFC2865] Section 5.25). Some organizations create CUI values which use the Network Access Identifier (NAI) format as defined in [RFC7542]. This format can allow the home network to be identified to the visited network, where the User-Name does not contain a realm. Such formats SHOULD NOT be used unless all parties involved have agreed to this behavior.

The CUI SHOULD be created via a construct similar to what is given below, where "+" indicates concatenation:

CUI = HASH(visited network data + user identifier + key)

This construct has the following conceptual parameters.

HASH

A cryptographic hash function.

visited network data

Data which identifies the visited network.

This data could be the Operator-Name attribute ([RFC5580] Section 4.1).

user identifier

The site-local user identifier. For tunnelled EAP methods such as PEAP or TTLS, this could be the user identity which is sent inside of the TLS tunnel.

key

A secret known only to the local network. The key is generally a large random string. It is used to help prevent dictionary attacks on the CUI.

Where the CUI needs to be constant across multiple user sessions or devices, the key can be a static value. It is generated once by the home network, and then stored for use in further CUI derivations.

Where the CUI needs to be unique per session, the above derivation SHOULD still be used, except that the "key" value will instead be a random number which is different for each session. Using such a

design again decouples the CUI creation from any requirement that it is unique per session, or constant per user. That decision can be changed at any time, and the only piece which needs to be updated is the derivation of the "key" field. In contrast, if the CUI is generated completely randomly per session, then it may be difficult for a system to later change that behavior to allow the CUI to be constant for a particular user.

If an NAI format is desired, the hash output can be converted to printable text, truncated if necessary to meet length limitations, and then an "@" character and a realm can be appended to it. The resulting text string is then in NAI form.

We note that the above recommendation is not invertible. That is, given a particular CUI, it is not possible to determine which visited network or user identifier was used to create it. If it is necessary to use the CUI to determine which user is associated with it, the local network still needs to store the full set of CUI values which are associated with each user.

If this tracking is too complex for a local network, it is possible to create the CUI via an invertible encryption process as follows:

CUI = ENCRYPT(key, visited network data + user identifier)

This construct has the following conceptual parameters.

ENCRYPT

A cryptographically secure encryption function

key

The encryption key. Note that the same key must not be used for more both hashing and encryption.

visited network data

Data which identifies the visited network.

This data could be the Operator-Name attribute ([RFC5580] Section 4.1).

user identifier

The site-local user identifier. For tunnelled EAP methods such as PEAP or TTLS, this could be the user identity which is sent inside of the TLS tunnel.

However, the use of a hash-based method is RECOMMENDED.

In short, the intent is for CUI to leak as little information as possible, and ideally be different for every session. However, business agreements, legal requirements, etc. may mandate different behavior. The intention of this section is not to mandate complete CUI privacy, but instead to clarify the trade-offs between CUI privacy and business realities.

7.2. User-Password and Proxying

The design of RADIUS means that when proxies receive Access-Request packets, the clear-text contents of the User-Password attribute are visible to the proxy. Despite various claims to the contrary, the User-Password attribute is never sent "in the clear" over the network. Instead, the password is protected by TLS (RADIUS/TLS) or via the obfuscation methods defined in [RFC2865] Section 5.2. However, the nature of RADIUS means that each proxy must first undo the password obfuscation of [RFC2865], and then re-do it when sending the outbound packet. As such, the proxy has the clear-text password visible to it, and stored in its application memory.

It is therefore possible for every intermediate proxy to snoop and record all user identities and passwords which they see. This exposure is most problematic when the proxies are administered by an organization other than the one which operates the home server. Even when all of the proxies are operated by the same organization, the existence of clear-text passwords on multiple machines is a security risk.

It is therefore NOT RECOMMENDED for organizations to send User-Password attributes in packets which are sent outside of the local organization. If RADIUS proxying is necessary, another authentication method SHOULD be used.

Client and server implementations SHOULD use programming techniques to securely wipe passwords from memory when they are no longer needed.

Organizations MAY still use User-Password attributes within their own systems, for reasons which we will explain in the next section.

7.3. Password Visibility and Storage

Some organizations may desire to increase the security of their network by using alternate authentication methods such as CHAP or MS-CHAP, instead of PAP. These attempts are largely misguided. If simple password-based methods must be used, in almost all situations, the security of the network as a whole is increased by using PAP in preference to CHAP or MS-CHAP. The reason is found through a simple risk analysis, which we explain in more detail below.

When PAP is used, any compromise of a system which sees the User-Password will result in that password leaking. In contrast, when CHAP or MS-CHAP is used, those methods do not share the password, but instead a hashed transformation of it. That hash output is in theory secure from attackers. However, the hashes used (MD5 and MD4 respectively) are decades old, have been broken, and are known to be insecure. Any security analysis which makes the claim that "User-Password insecure because it is protected with MD5" ignores the fact that the CHAP-Password attribute is constructed through substantially similar methods.

The difference between the two constructs is that the CHAP-Password depends on the hash of a visible Request Authenticator (or CHAP-Challenge) and the users password, while the obfuscated User-Password depends on the same Request Authenticator, and on the RADIUS shared secret. For an attacker, the difference between the two calculations is minimal. They can both be attacked with similar amounts of effort.

Further, any security analysis can not stop with the wire protocol. It must include all related systems which are affected by the choice of authentication methods. In this case, the most important piece of the system affected by these choices is the database which stores the passwords.

When PAP is used, the information stored in the database can be salted, and/or hashed in a form is commonly referred to as being in "crypt"ed form. The incoming clear-text password then undergoes the "crypt" transformation, and the two "crypt"ed passwords are compared. The passwords in the database are stored securely at all times, and any compromise of the database results in the disclosure of minimal information to an attacker. That is, the attacker cannot easily obtain the clear-text passwords from the database compromise.

The process for CHAP and MS-CHAP is inverted from the process for PAP. Using similar terminology as above for illustrative purposes, the "crypt"ed passwords are sent to the server. The server must obtain the clear-text (or NT hashed) password from the database, and

then perform the "crypt" operation on the password from the database. The two "crypt"ed passwords are then compared as was done with PAP. This inverted process has substantial and negative impacts on security.

When PAP is used, passwords are stored in clear-text only ephemerally in the memory of an application which receives and then verifies the password. Any compromise of that application results in the exposure of a small number of passwords which are visible at the time of compromise. If the compromise is undetected for an extended period of time, the number of exposed passwords would of course increase.

However, when CHAP or MS-CHAP are used, all of passwords are stored in clear-text in the database, all of the time. The database contents might be encrypted, but the decryption keys are necessarily accessible to the application which reads that database. Any compromise of the application means that the entire database can be immediately read and exfiltrated as a whole. The attacker then has complete access to all user identities, and all associated clear-text passwords.

The result is that when the system as a whole is taken into account, the risk of password compromise is less with PAP than with CHAP or MS-CHAP. It is therefore RECOMMENDED that administrators use PAP in preference to CHAP or MS-CHAP.

7.4. MS-CHAP

MS-CHAP (v1 in [RFC2433] and v2 in [RFC2759]) has major design flaws, and should not be used outside of a secure tunnel. As MS-CHAPv1 is not normally used, the discussion in this section will focus on MS-CHAPv2.

Recent developments demonstrate just how easy it is to attack MS-CHAPv2 exchanges, and obtain the "NT-hash" version of the password ([SENSEPOST]). The attack relies on a vulnerability in the protocol design in [RFC2759] Section 8.4. In that section, the response to the MS-CHAP challenge is calculated via three DES operations, which are based on the 16-octet NT-Hash form of the password. However, the DES operation requires 7 octet keys, so the 16-octet NT-Hash cannot be divided evenly into the 21 octets of keys required for the DES operation.

The solution in [RFC2759] Section 8.4 is to use the first 7 octets of the NT-Hash for the first DES key, the next 7 octets for the second DES key, leaving only 2 octets for the final DES key. The final DES key is padded with zeros. This construction means that an attacker who can observe the MS-CHAP2 exchange only needs to perform 2^16 DES operations in order to determine the final 2 octets of the original NT-Hash.

If the attacker has a database which correlates known passwords to NT-Hashes, then those two octets can be used as an index into that database, which returns a subset of candidate hashes. Those hashes are then checked via brute-force operations to see if they match the original MS-CHAPv2 data.

This process lowers the complexity of cracking MS-CHAP by nearly five orders of magnitude as compared to a brute-force attack. The attack has been demonstrated against databases containing tens to hundreds of millions of passwords. On a consumer-grade machine, the time required for such an attack to succeed is on the order of tens of milliseconds.

While this attack does require a database of known passwords, such databases are easy to find online, or to create locally from generator functions. Passwords created manually by people are notoriously predictable, and are highly likely to be found in a database of known passwords. In the extreme case of strong passwords, they will not be found in the database, and the attacker is still required to perform a brute-force dictionary search.

The result is that MS-CHAPv2 SHOULD be considered in most situations as being equivalent in security and privacy to PAP. It offers little benefit over PAP, and has many drawbacks as discussed here, and in the previous section.

There is one situation where MS-CHAP is significantly worse than PAP; where the MS-CHAP data is sent over the network in the clear. When the MS-CHAP data is not protected by TLS, it is visible to everyone who can observe the RADIUS traffic. Attackers who can see the MS-CHAP traffic can therefore obtain the underlying NT-Hash with essentially zero effort, as compared to cracking the RADIUS shared secret.

This document therefore mandates that MS-CHAP authentication data carried in RADIUS MUST NOT be sent in situations where the MS-CHAP data is visible to an observer. That is, MS-CHAP authentication MUST NOT be sent over RADIUS/UDP or RADIUS/TCP

7.5. EAP

If more complex authentication methods are needed, there are a number of EAP methods which can be used. These methods variously allow for the use of certificates (EAP-TLS), or passwords (EAP-TTLS [RFC5281], PEAP [I-D.josefsson-pppext-eap-tls-eap])) and EAP-pwd [RFC5931].

Where it is necessary to use intermediate proxies such as with eduroam [EDUROAM] and OpenRoaming [OPENROAMING], it is RECOMMENDED to use EAP instead of PAP, CHAP, or MS-CHAP. If passwords are used, they can be can be protected via TLS-based EAP methods such as EAP-TTLS or PEAP. Passwords can also be omitted entirely from being sent over the network, as with EAP-TLS [RFC9190] or EAP-pwd [RFC5931].

We also note that the TLS-based EAP methods which transport passwords also hide the passwords from intermediate RADIUS proxies. However, for the home authentication server, those EAP methods are still subject to the analysis above about PAP versus CHAP, along with the issues of storing passwords in a database.

7.6. Eliminating Proxies

The best way to avoid compromise of proxies is to eliminate proxies entirely. The use of dynamic peer discovery ([RFC7585]) means that the number of intermediate proxies is minimized.

However, the server on the visited network still acts as a proxy between the NAS and the home network. As a result, all of the above analysis still applies when [RFC7585] peer discovery is used.

8. Privacy Considerations

The primary focus of this document is addressing privacy and security considerations for RADIUS.

Deprecating insecure transport for RADIUS, and requiring secure transport means that personally identifying information is no longer sent "in the clear". As noted earlier in this document, such information can include MAC addresses, user identifiers, and user locations.

In addition, this document suggests ways to increase privacy by minimizing the use and exchange of PII.

9. Security Considerations

The primary focus of this document is addressing security and privacy considerations for RADIUS.

Deprecating insecure transport for RADIUS, and requiring secure transport means that many historical security issues with the RADIUS protocol no longer apply, or their impact is minimized.

We reiterate the discussion above, that any security analysis must be done on the system as a whole. It is not enough to put an expensive lock on the front door of a house while leaving the window next to it open, and then declare the house to be "secure". Any approach to security based on a simple checklist is at best naive, more truthfully is deeply misleading, and at worst such practices will serve to decrease security.

Implementers and administrators need to be aware of the issues raised in this document. They can then make the best choice for their local network which balances their requirements on privacy, security, and cost.

10. IANA Considerations

There are no IANA considerations in this document.

RFC Editor: This section may be removed before final publication.

11. Acknowledgements

Thanks to the many reviewers and commenters for raising topics to discuss, and for providing insight into the issues related to increasing the security of RADIUS. In no particular order, thanks to Margaret Cullen, Alexander Clouter, and Josh Howlett.

12. Changelog

* 01 - added more discussion of IPSec, and move TLS-PSK to its own document,
* 02 - Added text on Increasing the Security of Insecure Transports
* 03 - add text on CUI. Add notes on PAP vs CHAP security
* 04 - add text on security of MS-CHAP. Rearrange and reword many sections for clarity.
* 05 - Rework title to deprecating "insecure practices". Clarifications based on WG feedback.

13. References

13.1. Normative References

[BCP14] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC6421] Nelson, D., Ed., "Crypto-Agility Requirements for Remote Authentication Dial-In User Service (RADIUS)", RFC 6421, DOI 10.17487/RFC6421, November 2011, <https://www.rfc-editor.org/info/rfc6421>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[EDUROAM] eduroam, "eduroam", n.d., <https://eduroam.org>.

[I-D.ietf-radext-tls-psk] DeKok, A., "RADIUS and TLS-PSK", Work in Progress, Internet-Draft, draft-ietf-radext-tls-psk-03, 24 August 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-radext-tls-psk-03>.

[I-D.josefsson-pppext-eap-tls-eap] Palekar, A., Josefsson, S., Simon, D., and G. Zorn, "Protected EAP Protocol (PEAP) Version 2", Work in Progress, Internet-Draft, draft-josefsson-pppext-eap-tls-eap-10, 21 October 2004, <https://datatracker.ietf.org/doc/html/draft-josefsson-pppext-eap-tls-eap-10>.

[I-D.tomas-openroaming] Tomas, B., Grayson, M., Canpolat, N., Cockrell, B. A., and S. Gundavelli, "WBA OpenRoaming Wireless Federation", Work

in Progress, Internet-Draft, draft-tomas-openroaming-00, 14 June 2023, <https://datatracker.ietf.org/doc/html/draft-tomas-openroaming-00>.

[OPENROAMING] Alliance, W. B., "OpenRoaming: One global Wi-Fi network", n.d., <https://wballiance.com/openroaming/>.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2433] Zorn, G. and S. Cobb, "Microsoft PPP CHAP Extensions", RFC 2433, DOI 10.17487/RFC2433, October 1998, <https://www.rfc-editor.org/info/rfc2433>.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, DOI 10.17487/RFC2759, January 2000, <https://www.rfc-editor.org/info/rfc2759>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC2868] Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, DOI 10.17487/RFC2868, June 2000, <https://www.rfc-editor.org/info/rfc2868>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <https://www.rfc-editor.org/info/rfc3579>.

[RFC4372] Adrangi, F., Lior, A., Korhonen, J., and J. Loughney, "Chargeable User Identity", RFC 4372, DOI 10.17487/RFC4372, January 2006, <https://www.rfc-editor.org/info/rfc4372>.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, DOI 10.17487/RFC5080, December 2007, <https://www.rfc-editor.org/info/rfc5080>.

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

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <https://www.rfc-editor.org/info/rfc5281>.

[RFC5580] Tschofenig, H., Ed., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, DOI 10.17487/RFC5580, August 2009, <https://www.rfc-editor.org/info/rfc5580>.

[RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, DOI 10.17487/RFC5931, August 2010, <https://www.rfc-editor.org/info/rfc5931>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6218] Zorn, G., Zhang, T., Walker, J., and J. Salowey, "Cisco Vendor-Specific RADIUS Attributes for the Delivery of Keying Material", RFC 6218, DOI 10.17487/RFC6218, April 2011, <https://www.rfc-editor.org/info/rfc6218>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/info/rfc6613>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC6973] Cooper, A., Tschofenig, H., Aboba, B., Peterson, J., Morris, J., Hansen, M., and R. Smith, "Privacy Considerations for Internet Protocols", RFC 6973, DOI 10.17487/RFC6973, July 2013, <https://www.rfc-editor.org/info/rfc6973>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/info/rfc7360>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7525, DOI 10.17487/RFC7525, May 2015, <https://www.rfc-editor.org/info/rfc7525>.

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

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8559] DeKok, A. and J. Korhonen, "Dynamic Authorization Proxying in the Remote Authentication Dial-In User Service (RADIUS) Protocol", RFC 8559, DOI 10.17487/RFC8559, April 2019, <https://www.rfc-editor.org/info/rfc8559>.

[RFC9190] Preuß Mattsson, J. and M. Sethi, "EAP-TLS 1.3: Using the Extensible Authentication Protocol with TLS 1.3", RFC 9190, DOI 10.17487/RFC9190, February 2022, <https://www.rfc-editor.org/info/rfc9190>.

[RFC9257] Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External Pre-Shared Key (PSK) Usage in TLS", RFC 9257, DOI 10.17487/RFC9257, July 2022, <https://www.rfc-editor.org/info/rfc9257>.

[SENSEPOST] Sensepost, "Cracking MS-CHAP", n.d., <https://github.com/sensepost/assless-chaps>.

[SPOOFING] Cudbard-Bell, A., "Wi-Fi Spoofing for Fun and Profit", n.d., <https://networkradius.com/articles/2021/08/04/wifi-spoofing.html>.

[WIFILOC] Alliance, W.-F., "Accurate indoor location with Wi-Fi connectivity", n.d., <https://www.wi-fi.org/discover-wi-fi/wi-fi-location>.

Author's Address

Alan DeKok
FreeRADIUS
Email: aland@freeradius.org

RADEXT Working Group
Internet-Draft
Updates: 6613 7360 (if approved)
Intended status: Standards Track
Expires: 21 October 2023

A. DeKok
FreeRADIUS

19 April 2023

RADIUS Version 1.1
draft-dekok-radext-radiusv11-05

Abstract

This document defines Application-Layer Protocol Negotiation Extensions for use with RADIUS/TLS and RADIUS/DTLS. These extensions permit the negotiation of an additional application protocol for RADIUS over (D)TLS. No changes are made to RADIUS/UDP or RADIUS/TCP. The extensions allow the negotiation of a transport profile where the RADIUS shared secret is no longer used, and all MD5-based packet signing and attribute obfuscation methods are removed. When this extension is used, the previous Authenticator field is repurposed to contain an explicit request / response identifier, called a Token. The Token also allows more than 256 packets to be outstanding on one connection.

This extension can be seen as a transport profile for RADIUS, as it is not an entirely new protocol. It uses the existing RADIUS packet layout and attribute format without change. As such, it can carry all present and future RADIUS attributes. Implementation of this extension requires only minor changes to the protocol encoder and decoder functionality. The protocol defined by this extension is named "RADIUS version 1.1", or "RADIUS/1.1".

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekok-radext-radiusv11/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/radiusv11.git.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The RADIUS protocol [RFC2865] uses MD5 [RFC1321] to sign packets, and to obfuscate certain attributes. Decades of cryptographic research has shown that MD5 is insecure, and MD5 should no longer be used. In addition, the dependency on MD5 makes it impossible to use RADIUS in a FIPS-140 compliant system, as FIPS-140 forbids systems from relying on insecure cryptographic methods for security. There are many prior discussions of MD5 insecurities which we will not repeat here. These discussions are most notably in [RFC6151], and in Section 3 of [RFC6421], among others.

While additional transport protocols were defined for RADIUS in TCP ([RFC6613]), TLS ([RFC6614]), and DTLS ([RFC7360]), those transports still relied on MD5. That is, the shared secret was used along with MD5, even when the RADIUS packets were being transported in (D)TLS. At the time, the consensus of the RADEXT working group was that this continued use of MD5 was acceptable. TLS was seen as a simple "wrapper" around RADIUS, while using a fixed shared secret. The intention at the time was to allow the use of (D)TLS while making essentially no changes to the basic RADIUS encoding, decoding, signing, and packet validation.

The ensuing years have shown that it is important for RADIUS to remove its dependency on MD5. The continued use of MD5 is no longer acceptable in a security-conscious environment. The use of MD5 in [RFC6614] and [RFC7360] adds no security or privacy over that provided by TLS. It is time to remove the use of MD5 from RADIUS.

This document defines an Application-Layer Protocol Negotiation (ALPN) [RFC7301] extension for RADIUS which removes the dependency on MD5. Systems which implement this transport profile are therefore capable of being FIPS-140 compliant. This extension can best be understood as a transport profile for RADIUS, rather than a whole-sale revision of the RADIUS protocol. A preliminary implementation has shown that only minor changes are required to support RADIUS/1.1 on top of an existing RADIUS server.

The changes from traditional TLS-based transports for RADIUS are as follows:

* ALPN is used for negotiation of this extension,

* TLS 1.3 or later is required,

* all uses of the RADIUS shared secret have been removed,

* The now-unused Request and Response Authenticator fields have been repurposed to carry an opaque Token which identifies requests and responses,

* The Identifier field is no longer used, and has been replaced by the Token field,

* The Message-Authenticator attribute ([RFC3579] Section 3.2) is not sent in any packet, and if received is ignored,

* Attributes such as User-Password, Tunnel-Password, and MS-MPPE keys are sent encoded as "text" ([RFC8044] Section 3.4) or "octets" ([RFC8044] Section 3.5), without the previous MD5-based obfuscation. This obfuscation is no longer necessary, as the data is secured and kept private through the use of TLS,

* Future RADIUS specifications are forbidden from defining new cryptographic primitives.

The following items are left unchanged from traditional TLS-based transports for RADIUS:

* the RADIUS packet header is the same size, and the Code and Length fields ([RFC2865] Section 3) have the same meaning as before,

* All attributes which do not use MD5-based obfuscation methods are encoded using the normal RADIUS methods, and have the same meaning as before,

* As this extension is a transport profile for one "hop" (client to server connection), it does not impact any other connection used by a client or server. The only systems which are aware that this transport profile is in use are the client and server which have negotiated the use of this extension on a particular shared connection,

* This extension uses the same ports (2083/tcp and 2083/udp) which are defined for RADIUS/TLS [RFC6614] and RADIUS/DTLS [RFC7360].

A major benefit of this extensions is that a home server which implements it can also choose to also implement full FIPS-140 compliance. That is, a home server can remove all uses of MD4 and MD5. In that case, however, the home server will not support CHAP, MS-CHAP, or any authentication method which uses MD4 or MD5. We note that the choice of which authentication method to accept is always left to the home server. This specification does not change any authentication method carried in RADIUS, and does not mandate (or forbid) the use of any authentication method for any system.

As for proxies, there was never a requirement that proxies implement CHAP or MS-CHAP authentication. So far as a proxy is concerned, attributes relating to CHAP and MS-CHAP are simply opaque data that is transported unchanged to the next hop. As such, it is possible for a FIPS-140 compliant proxy to transport authentication methods which depend on MD4 or MD5, so long as that data is forwarded to a home server which supports those methods.

We reiterate that the decision to support (or not) any authentication method is entirely site local, and is not a requirement of this specification. The contents or meaning of any RADIUS attribute other than Message-Authenticator (and similar attributes) are not modified. The only change to the Message-Authenticator attribute is that is no longer used.

Unless otherwise described in this document, all RADIUS requirements apply to this extension. That is, this specification defines a transport profile for RADIUS. It is not an entirely new protocol, and it defines only minor changes to the existing RADIUS protocol. It does not change the RADIUS packet format, attribute format, etc. This specification is compatible with all RADIUS attributes, past, present, and future.

This specification is compatible with existing implementations of RADIUS/TLS and RADIUS/DTLS. There is no need to define an ALPN name for those protocols, as implementations can simply not send an ALPN name when those protocols are used. Backwards compatibility with existing implementations is both required, and assumed.

This specification is compatible with all past and future RADIUS specifications. There is no need for any RADIUS specification to mention this transport profile by name, or to make provisions for this specification. This specification defines how to transform RADIUS into RADIUS/1.1, and no further discussion of that transformation is necessary.

In short, when negotiated on a connection, this specification permits implementations to avoid MD5 when signing packets, or obfuscating certain attributes.

2. Terminology

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.

* ALPN

  Application-Layer Protocol Negotiation, as defined in [RFC7301].

* RADIUS

  The Remote Authentication Dial-In User Service protocol, as defined in [RFC2865], [RFC2865], and [RFC5176] among others.

  While this protocol can be viewed as "RADIUS/1.0", for simplicity and historical compatibility, we keep the name "RADIUS".

* RADIUS/UDP

  RADIUS over the User Datagram Protocol as define above.

* RADIUS/TCP

  RADIUS over the Transmission Control Protocol [RFC6613].

* RADIUS/TLS

  RADIUS over the Transport Layer Security protocol [RFC6614].

* RADIUS/DTLS

  RADIUS over the Datagram Transport Layer Security protocol [RFC7360].

* RADIUS over TLS

  Either RADIUS/TLS or RADIUS/DTLS. This terminology is used instead of alternatives such as "RADIUS/(D)TLS", or "either RADIUS/TLS or RADIUS/DTLS".

* RADIUS/1.1

  The transport profile defined in this document, which stands for "RADIUS version 1.1". We use RADIUS/1.1 to refer interchangeably to TLS and DTLS transport.

* TLS

  the Transport Layer Security protocol. Generally when we refer to TLS in this document, we are referring interchangeably to TLS or DTLS transport.

3. The RADIUS/1.1 Transport profile for RADIUS

This section describes the ALPN transport profile in detail. It first gives the name used for ALPN, and then describes how ALPN is configured and negotiated by client and server. It then concludes by discussing TLS issues such as what to do for ALPN during session resumption.

3.1. ALPN Name for RADIUS/1.1

The ALPN name defined for RADIUS/1.1 is as follows:

"radius/1.1"

The protocol defined by this specification.

Where ALPN is not configured or is not received in a TLS connection, systems supporting ALPN MUST not use RADIUS/1.1.

Where ALPN is configured, the client signals support by sending the ALPN string "radius/1.1". The server can accept this proposal and reply with the ALPN string "radius/1.1", or reject this proposal, and not reply with any ALPN string.

Implementations MUST signal ALPN "radius/1.1" in order for it to be used in a connection. Implementations MUST NOT have an administrative flag which causes a connection to use "radius/1.1" without signalling that protocol via ALPN.

The next step in defining RADIUS/1.1 is to review how ALPN works.

3.2. Operation of ALPN

Once a system has been configured to support ALPN, it is negotiated on a per-connection basis as per [RFC7301]. We give a brief overview here of ALPN in order to provide a high-level description ALPN for readers who do not need to understand [RFC7301] in detail. This section is not normative.

1) The client proposes ALPN by sending an ALPN extension in the ClientHello. This extension lists one or more application protocols by name.

2) The server receives the extension, and validates the application protocol name against the list it has configured.

   If the server finds no acceptable common protocols, it closes the connection.

3) Otherwise, the server return a ServerHello with either no ALPN extension, or an ALPN extension with only one named application protocol.

   If the client does not signal ALPN, or server does not accept the ALPN proposal, the server does not reply with any ALPN name.

4) The client receives the ServerHello, validates the application protocol (if any) against the name it sent, and records the application protocol which was chosen

   This check is necessary in order for the client to both know which protocol the server has selected, and to validate that the protocol sent by the server is acceptable to the client.

The next step in defining RADIUS/1.1 is to define how ALPN is configured on the client and server, and to give more detailed requirements on ALPN configuration and operation.

3.3. Configuration of ALPN for RADIUS/1.1

Clients or servers supporting this specification can do so by extending their TLS configuration through the addition of a new configuration flag, called "RADIUS/1.1" here. The exact name given below does not need to be used, but it is RECOMMENDED that administrative interfaces or programming interfaces use a similar name in order to provide consistent terminology. This flag controls how the implementations signal use of this protocol via ALPN.

Configuration Flag Name

RADIUS/1.1

Allowed Values

forbid - Forbid the use of RADIUS/1.1

A client with this configuration MUST NOT signal any protocol name via ALPN. The system MUST use RADIUS over TLS as defined in [RFC6614] and [RFC7360].

A server with this configuration MUST NOT signal any protocol name via ALPN. The system MUST use RADIUS over TLS as defined in [RFC6614] and [RFC7360].

A server with this configuration MUST NOT close the connection if it receives an ALPN name from the client. Instead, it simply does not reply with ALPN.

allow - Allow (or negotiate) the use of RADIUS/1.1

This value MUST be the default setting for implementations which support this specification.

A client with this configuration MUST use ALPN to signal that "radius/1.1" can be used. The client MUST use RADIUS/1.1 if the server responds signalling ALPN "radius/1.1". If no ALPN response is received from the server, the client MUST use RADIUS over TLS as defined in previous specifications.

A server with this configuration MAY reply to a client with an ALPN string of "radius/1.1", but only if the client first signals support for that protocol name via ALPN. If the client does not signal ALPN, the server MUST NOT reply with any ALPN name.

require - Require the use of RADIUS/1.1

A client with this configuration MUST use ALPN to signal that "radius/1.1" can be used. The client MUST use RADIUS/1.1 if the server responds signalling ALPN "radius/1.1". If no ALPN response is received from the server, the client MUST close the connection.

A server with this configuration MUST close the connection if the client does not signal "radius/1.1" via ALPN.

A server with this configuration MUST reply with the ALPN protocol name "radius/1.1" if the client signals "radius/1.1". The server and client both MUST then use RADIUS/1.1 as the application-layer protocol. There is no reason to signal support for a protocol, and then not use it.

Note that systems implementing this specification, but configured with "forbid" as above, will behave exactly the same as systems which do not implement this specification.

If a client or server determines that there are no compatible application protocol names, then as per [RFC7301] Section 3.2, it MUST send a TLS alert of "no_application_protocol" (120), which signals the other end that there is no compatible application protocol. It MUST then close the connection.

It is RECOMMENDED that a descriptive error is logged in this situation, so that an administrator can determine why a particular connection failed. The log message SHOULD include information about the other end of the connection, such as IP address, certificate information, etc. Similarly, a system receiving a TLS alert of "no_application_protocol" SHOULD log a descriptive error message. Such error messages are critical for helping administrators to diagnose connectivity issues.

Note that there is no way for a client to signal if its' RADIUS/1.1 configuration is set to "allow" or "require". The client MUST signal "radius/1.1" via ALPN when it is configured with either value. The difference between the two values for the client is only in how it handles reponses from the server.

Similarly, there is no way for a server to signal if its' RADIUS/1.1 configuration is set to "allow" or "require". In both cases if it receives "radius/1.1" from the client via ALPN, the server MUST reply with "radius/1.1", and agree to that negotiation. The difference between the two values for the server is how it handles the situation when no ALPN is signalled from the client.

3.3.1. Tabular Summary

The preceding text gives a large number of recommendations. In order to give a simpler description of the outcomes, a table of possible behaviors for client/server values of the RADIUS/1.1 flag is given below. This table and the names given below are for informational and descriptive purposes only. This section is not normative.

Client	Server: no ALPN	Server: forbid	Server: allow	Server: require
No ALPN	RADIUS	RADIUS	RADIUS	Close Note 1
forbid	RADIUS	RADIUS	RADIUS	Close Note 1
allow	RADIUS Note 3	RADIUS Note 3	OK	OK
require	Close Note 2	Close Note 2	OK	OK

Figure 1: Possible outcomes for ALPN Negotiation

The table entries above have the following meaning:

Close

Note 1 - the server closes the connection, as the client does not do RADIUS/1.1

Note 2 - the client closes the connection, as the server does not do RADIUS/1.1

RADIUS

RADIUS over TLS is used. RADIUS/1.1 is not used.

Note 3 - The client sends ALPN, but the server does not reply with ALPN.

OK

RADIUS/1.1 is used by both parties.

> The client sends "radius/1.1" via ALPN, and the server repies with "radius/1.1" via ALPN.

3.4. Additional TLS issues

Implementations of this specification MUST require TLS version 1.3 or later.

Implementations of this specification MUST support TLS-PSK.

3.5. Session Resumption

[RFC7301] Section 3.1 states that ALPN is negotiated on each connection, even if session resumption is used:

> When session resumption or session tickets [RFC5077] are used, the previous contents of this extension are irrelevant, and only the values in the new handshake messages are considered.

In order to prevent down-bidding attacks, RADIUS servers which negotiate the "radius/1.1" protocol MUST associate that information with the session ticket. On session resumption, the server MUST advertise only the capability to do "radius/1.1" for that session. That is, even if the server configuration is "allow" for new connections, it MUST signal "radius/1.1" when resuming a session which had previously negotiated "radius/1.1".

If a server sees that a client had previously negotiated RADIUS/1.1 for a session, but the client is now attempting to resume the sessions without signalling the use of RADIUS/1.1, the server MUST close the connection. The server SHOULD send an appropate TLS error, such as no_application_protocol (120), or insufficient_security (71). The server SHOULD log a descriptive message as described above.

4. RADIUS/1.1 Packet and Attribute Formats

This section describes the application-layer data which is sent inside of (D)TLS when using the RADIUS/1.1 protocol. Unless otherwise discussed herein, the application-layer data is unchanged from traditional RADIUS. This protocol is only used when "radius/1.1" has been negotiated by both ends of a connection.

4.1. RADIUS/1.1 Packet Format

When RADIUS/1.1 is used, the RADIUS header is modified from standard RADIUS. While the header has the same size, some fields have different meaning. The Identifier and the Request Authenticator and Response Authenticator fields are no longer used. Any operations which depend on those fields MUST NOT be performed. As packet signing and security are handled by the TLS layer, RADIUS-specific cryptographic primitives are no longer used.

A summary of the RADIUS/1.1 packet format is shown below. The fields are transmitted from left to right.

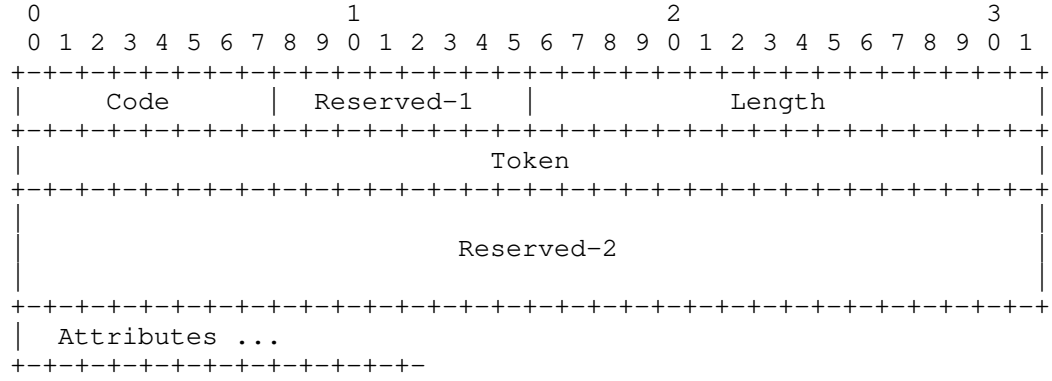


Figure 2: The RADIUS/1.1 Packet Format

Code

The Code field is one octet, and identifies the type of RADIUS packet.

The meaning of the Code field is unchanged from previous RADIUS specifications.

Reserved-1

The Reserved-1 field is one octet. It MUST be set to zero for all packets.

This field was previously called "Identifier" in RADIUS. It is now unused, as the Token field is now used to identify requests and responses.

Length

The Length field is two octets.

The meaning of the Length field is unchanged from previous RADIUS specifications.

Token

The Token field is four octets, and aids in matching requests and replies, as a replacement for the Identifier field. The RADIUS server can detect a duplicate request if it receives the same Token value for two packets on a particular connection.

Further requirements are given below in Section 4.2.1 for sending packets, and in Section 4.2.2 for receiving packets.

Reserved-2

The Reserved-2 field is twelve (12) octets in length.

These octets MUST be set to zero when sending a packet.

These octets MUST be ignored when receiving a packet.

These octets are reserved for future protocol extensions.

4.2. The Token Field

This section describes in more detail how the Token field is used.

4.2.1. Sending Packets

A client which sends packets uses the Token field to increase the number of RADIUS packets which can be sent over one connection.

The Token field MUST change for every new unique packet which is sent on the same connection. For DTLS transport, it is possible to retransmit duplicate packets, in which case the Token value MUST NOT be changed when a duplicate packet is (re)sent. When the contents of a retransmitted packet change for any reason (such changing Acct-Delay-Time as discussed in [RFC2866] Section 5.2), the Token value MUST be changed. Note that on reliable transports, packets are never retransmitted, and therefore every new packet sent has a unique Token value.

Systems generating the Token can do so via any method they choose, but for simplicity, it is RECOMMENDED that the Token values be generated from a 32-bit counter which is unique to each connection. Such a counter SHOULD be initialized to a random value, taken from a random number generator, whenever a new connection is opened. The counter can then be incremented for every new packet which is sent.

As there is no special meaning for the Token, there is no meaning when a counter "wraps" around from a high value back to zero. The originating system can simply continue to increment the Token value.

Once a RADIUS response to a request has been received and there is no need to track the packet any longer, the Token value MAY be reused. This SHOULD be after a suitable delay to ensure that Token values do not conflict with outstanding packets. Note that the counter method described above for generating Token values will automatically ensure a long delay between multiple uses of the same Token value, at the cost of maintaining a single 32-bit counter. Any other method of generating unique and non-conflicting Token values is likely to require substantially more resources to track outstanding Token values.

If a RADIUS client has multiple independent subsystems which send packets to a server, each subsystem MAY open a new port which is unique to that subsystem. There is no requirement that all packets go over one particular connection. That is, despite the use of a 32-bit Token field, RADIUS/1.1 clients are still permitted to open multiple source ports as discussed in [RFC2865] Section 2.5.

4.2.2. Receiving Packets

A server which receives RADIUS/1.1 packets MUST perform packet deduplication for all situations where it is required by RADIUS. Where RADIUS does not require deduplication (e.g. TLS transport), the server SHOULD NOT do deduplication.

We note that in previous RADIUS specifications, the Identifier field could have the same value for different types of packets on the same connection, e.g. for Access-Request and Accounting-Request. This overlap required that RADIUS clients and servers track the Identifier field, not only on a per-connection basis, but also on a per-packet type basis. This behavior adds complexity to implementations.

When using RADIUS/1.1, implementations MUST instead do deduplication only on the Token field, and not on any other field or fields in the packet header. A server MUST treat the Token as being an opaque field with no intrinsic meaning. While the recommendation above is for the sender to use a counter, other implementations are possible,

valid, and permitted. For example, a system could use a pseudo-random number generator with a long period to generate unique values for the Token field.

Where Token deduplication is done, it MUST be done on a per-connection basis. If two packets which are received on different connections contain the same Token value, then those packets MUST be treated as distinct (i.e. different) packets.

This change from RADIUS means that the Identifier field is no longer useful. The Reserved-1 field (previously used as the Identifier) MUST be set to zero for all RADIUS/1.1 packets. RADIUS/1.1 Implementations MUST NOT examine this field or use it for packet tracking or deduplication.

5. Attribute handling

Most attributes in RADIUS have no special encoding "on the wire", or any special meaning between client and server. Unless discussed in this section, all RADIUS attributes are unchanged in this specification. This requirement includes attributes which contain a tag, as defined in [RFC2868].

5.1. Obfuscated Attributes

As (D)TLS is used for this specification, there is no need to hide the contents of an attribute on a hop-by-hop basis. The TLS transport ensures that all attribute contents are hidden from any observer.

Attributes defined as being obfuscated via MD5 no longer have the obfuscation step applied when RADIUS/1.1 is used. Instead, those attributes are simply encoded as their values, as with any other attribute. Their encoding method MUST follow the encoding for the underlying data type, with any encryption / obfuscation step omitted.

There are often concerns where RADIUS is used, that passwords are sent "in cleartext" across the network. This allegation was never true for RADIUS, and definitely untrue when (D)TLS transport is used. While passwords are encoded in packets as strings, the packets (and thus passwords) are protected by TLS. For the unsure reader this protocol is the same TLS which protects passwords used for web logins, e-mail reception and sending, etc. As a result, any claims that passwords are sent "in the clear" are false.

There are risks from sending passwords over the network, even when they are protected by TLS. One such risk somes from the common practice of multi-hop RADIUS routing. As all security in RADIUS is

on a hop-by-hop basis, every proxy which receives a RADIUS packet can see (and modify) all of the information in the packet. Sites wishing to avoid proxies SHOULD use dynamic peer discovery [RFC7585], which permits clients to make connections directly to authoritative servers for a realm.

These others ways to mitigate these risks. One is by ensuring that the RADIUS over TLS session parameters are verified before sending the password, usually via a method such as verifying a server certificate. That is, passwords should only be sent to verified and trusted parties. If the TLS session parameters are not verified, then it is trivial to convince the RADIUS client to send passwords to anyone.

Another way to mitigate these risks is for the system being authenticated to use an authentication protocol which never sends passwords (e.g. EAP-PWD [RFC5931]), or which sends passwords protected by a TLS tunnel (e.g. EAP-TTLS [RFC5281]). The processes to choose and configuring an authentication protocol are strongly site-dependent, so further discussion of these issues are outside of the scope of this document. The goal here is to ensure that the reader has enough information to make an informed decision.

5.1.1. User-Password

The User-Password attribute ([RFC2865] Section 5.2) MUST be encoded the same as any other attribute of data type 'string' ([RFC8044] Section 3.5).

The contents of the User-Password field MUST be at least one octet in length, and MUST NOT be more than 128 octets in length. This limitation is maintained from [RFC2865] Section 5.2 for compatibility with legacy transports.

Note that the User-Password attribute is not of data type 'text'. The original reason in [RFC2865] was because the attribute was encoded as an opaque and obfuscated binary blob. We maintain that data type here, even though the attribute is no longer obfuscated. The contents of the User-Password attribute do not have to printable text, or UTF-8 data as per the definition of the 'text' data type in [RFC8044] Section 3.4.

However, implementations should be aware that passwords are often printable text, and where the passwords are printable text, it can be useful to store and display them as printable text. Where implementations can process non-printable data in the 'text' data type, they MAY use the data type 'text' for User-Password.

5.1.2. CHAP-Challenge

[RFC2865] Section 5.2 allows for the CHAP challenge to be taken from either the CHAP-Challenge attribute ([RFC2865] Section 5.40), or the Request Authenticator field. Since RADIUS/1.1 connections no longer use a Request Authenticator field, proxies may receive an Access-Request containing a CHAP-Password attribute ([RFC2865] Section 5.2) but without a CHAP-Challenge attribute ([RFC2865] Section 5.40). In this case, proxies which forward that CHAP-Password attribute over a RADIUS/1.1 connection MUST create a CHAP-Challenge attribute in the proxied packet using the contents from the Request Authenticator.

5.1.3. Tunnel-Password

The Tunnel-Password attribute ([RFC2868] Section 3.5) MUST be encoded the same as any other attribute of data type 'text' which contains a tag, such as Tunnel-Client-Endpoint ([RFC2868] Section 3.3). Since the attribute is no longer obfuscated, there is no need for a Salt field or Data-Length fields as described in [RFC2868] Section 3.5, and the textual value of the password can simply be encoded as-is.

Note that the Tunnel-Password attribute is not of data type 'text'. The original reason in [RFC2868] was because the attribute was encoded as an opaque and obfuscated binary blob. We maintain that data type here, even though the attribute is no longer obfuscated. The contents of the Tunnel-Password attribute do not have to printable text, or UTF-8 data as per the definition of the 'text' data type in [RFC8044] Section 3.4.

However, implementations should be aware that passwords are often printable text, and where the passwords are printable text, it can be useful to store and display them as printable text. Where implementations can process non-printable data in the 'text' data type, they MAY use the data type 'text' for Tunnel-Password.

5.1.4. Vendor-Specific Attributes

Any Vendor-Specific attribute which uses similar obfuscation MUST be encoded as per their base data type. Specifically, the MS-MPPE-Send-Key and MS-MPPE-Recv-Key attributes ([RFC2548] Section 2.4) MUST be encoded as any other attribute of data type 'text' ([RFC8044] Section 3.4).

We note that as the RADIUS shared secret is no longer used, it is no longer possible or necessary for any attribute to be obfuscated on a hop-by-hop basis using the previous methods defined for RADIUS.

5.2. Message-Authenticator

The Message-Authenticator attribute ([RFC3579] Section 3.2) MUST NOT be sent over a RADIUS/1.1 connection. That attribute is no longer used or needed.

If the Message-Authenticator attribute is received over a RADIUS/1.1 connection, the attribute MUST be silently discarded, or treated as an "invalid attribute", as defined in [RFC6929] Section 2.8. That is, the Message-Authenticator attribute is no longer used to sign packets. Its existence (or not) in this transport is meaningless.

We note that any packet which contains a Message-Authenticator attribute can still be processed. There is no need to discard an entire packet simply because it contains a Message-Authenticator attribute. Only the Message-Authenticator attribute itself is ignored.

5.3. Message-Authentication-Code

Similarly, the Message-Authentication-Code attribute defined in [RFC6218] Section 3.3 MUST NOT be sent over a RADIUS/1.1 connection. That attribute MUST be treated the same as Message-Authenticator, above.

As the Message-Authentication-Code attribute is no longer used, the related MAC-Randomizer attribute [RFC6218] Section 3.2 is also no longer used. It MUST also be treated the same was as Message-Authenticator, above.

5.4. CHAP, MS-CHAP, etc.

While some attributes such as CHAP-Password, etc. depend on insecure cryptographic primitives such as MD5, these attributes are treated as opaque blobs when sent between a RADIUS client and server. The contents of the attributes are not obfuscated, and they do not depend on the RADIUS shared secret. As a result, these attributes are unchanged in RADIUS/1.1.

A server implementing this specification can proxy CHAP, MS-CHAP, etc. without any issue. A home server implementing this specification can authenticate CHAP, MS-CHAP, etc. without any issue.

5.5. Original-Packet-Code

The Original-Packet-Code attribute ([RFC7930] Section 4) MUST NOT be sent over a RADIUS/1.1 connection. That attribute is no longer used or needed.

If the Original-Packet-Code attribute is received over a RADIUS/1.1 connection, the attribute MUST either be silently discarded, or be treated an as "invalid attribute", as defined in [RFC6929], Section 2.8. That is, existence of the Token field means that the Original-Packet-Code attribute is no longer needed to correlate Protocol-Error replies with outstanding requests. As such, the Original-Packet-Code attribute is not used in RADIUS/1.1.

We note that any packet which contains an Original-Packet-Code attribute can still be processed. There is no need to discard an entire packet simply because it contains an Original-Packet-Code attribute.

6. Other Considerations

Most of the differences between RADIUS and RADIUS/1.1 are in the packet header and attribute handling, as discussed above. The remaining issues are a small set of unrelated topics, and are discussed here.

6.1. Status-Server

[RFC6613] Section 2.6.5, and by extension [RFC7360] suggest that the Identifier value zero (0) be reserved for use with Status-Server as an application-layer watchdog. This practice MUST NOT be used for RADIUS/1.1, as the Identifier field is no longer used.

The rationale for reserving one value of the Identifier field was the limited number of Identifiers available (256), and the overlap in Identifiers between Access-Request packets and Status-Server packers. If all 256 Identifier values had been used to send Access-Request packets, then there would be no Identifier value available for sending a Status-Server Packet.

In contrast, the Token field allows for 2^32 outstanding packets on one RADIUS/1.1 connection. If there is a need to send a Status-Server packet, it is always possible to allocate a new value for the Token field. Similarly, the value zero (0) for the Token field has no special meaning. The edge condition is that there are 2^32 outstanding packets on one connection with no new Token value available for Status-Server. In which case there are other serious issues, such as allowing billions of packets to be oustanding. The safest way forward is likely to just close the connection.

6.2. Proxies

A RADIUS proxy normally decodes and then re-encodes all attributes, included obfuscated ones. A RADIUS proxy will not generally rewrite the content of the attributes it proxies (unless site-local policy requires such a rewrite). While some attributes may be modified due to administrative or policy rules on the proxy, the proxy will generally not rewrite the contents of attributes such as User-Password, Tunnel-Password, CHAP-Password, MS-CHAP-Password, MS-MPPE keys, etc. All attributes are therefore transported through a RADIUS/1.1 connection without changing their values or contents.

A proxy may negotiate RADIUS/1.1 (or not) with a particular client or clients, and it may negotiate RADIUS/1.1 (or not) with a server or servers it connect to, in any combination. As a result, this specification is fully compatible with all past, present, and future RADIUS attributes.

6.3. Crypto-Agility

The crypto-agility requirements of [RFC6421] are addressed in [RFC6614] Appendix C, and in Section 10.1 of [RFC7360]. This specification makes no changes from, or additions to, those specifications. The use of ALPN, and the removal of MD5 has no impact on security or privacy of the protocol.

RADIUS/TLS has been widely deployed in at least eduroam and in OpenRoaming. RADIUS/DTLS has seen less adoption, but it is known to be supported in many RADIUS clients and servers.

It is RECOMMENDED that all implementations of RADIUS over TLS be updated to support this specification. The effort to implement this specification is minimal. Once implementations support this specification, administrators can gain the benefit of it with little or no configuration changes. This specification is backwards compatible with [RFC6614] and [RFC7360]. It is only potentially subject to downbidding attacks if implementations do not enforce ALPN negotiation correctly on session resumption.

All crypto-agility needed or used by this specification is implemented in TLS. This specification also removes all cryptographic primitives from the application-layer protocol (RADIUS) being transported by TLS. As discussed in the next section below, this specification also bans the development of all new cryptographic or crypto-agility methods in the RADIUS protocol.

6.4. Future Standards

This specification defines a new transport profile for RADIUS. It does not define a completely new protocol. As such, any future attribute definitions MUST first be defined for RADIUS/UDP, after which those definitions can be applied to this transport profile.

New specifications MAY define new attributes which use the obfuscation methods for User-Password as defined in [RFC2865] Section 5.2, or for Tunnel-Password as defined in [RFC2868] Section 3.5. There is no need for those specifications to describe how those new attributes are transported in RADIUS/1.1. Since RADIUS/1.1 does not use MD5, any obfuscated attributes will by definition be transported as their underlying data type, ("text" ([RFC8044] Section 3.4) or "string" ([RFC8044] Section 3.5). a New RADIUS specifications MUST NOT define attributes which can only be transported via RADIUS over TLS. The RADIUS protocol has no way to signal the security requirements of individual attributes. Any existing implementation will handle these new attributes as "Invalid Attributes" ([RFC6929] Section 2.8), and could forward them over an insecure link. As RADIUS security and signalling is hop-by-hop, there is no way for a RADIUS client or server to even know if such forwarding is taking place. For these reasons and more, it is therefore inappropriate to define new attributes which are only secure if they use a secure transport layer.

New specifications do not need to mention this transport profile, or make any special provisions for dealing with it. This specification defines how RADIUS packet encoding, decoding, signing, and verification are performed when using RADIUS/1.1. So long as any future specification uses the existing encoding, etc. schemes defined for RADIUS, no additional text in future documents is necessary in order to be compatible with RADIUS/1.1.

To close the final loophole, this document updates [RFC2865] at al. to state that any new RADIUS specification MUST NOT introduce new "ad hoc" cryptographic primitives as was done with User-Password and Tunnel-Password. That is, RADIUS-specific cryptographic methods existing as of the publication of this document can continue to be used for historical compatibility. However, all new cryptographic work in RADIUS is forbidden. There is insufficient expertise in the RADIUS community to securely design new cryptography.

7. Implementation Status

(This section to be removed by the RFC editor.)

This specification is being implemented (client and server) in the FreeRADIUS project which is hosted on GitHub at https://github.com/FreeRADIUS/freeradius-server/tree/v3.2.x The code implementation "diff" is approximately 1,000 lines, including build system changes and changes to configuration parsers.

8. Privacy Considerations

This specification requires secure transport for RADIUS, and this has all of the privacy benefits of RADIUS/TLS [RFC6614] and RADIUS/DTLS [RFC7360]. All of the insecure uses of RADIUS have been removed.

9. Security Considerations

The primary focus of this document is addressing security considerations for RADIUS.

10. IANA Considerations

IANA is requested to update the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry with one new entry:

Protocol: radius/1.1
Id. Sequence: 0x72 0x61 0x64 0x69 0x75 0x73 0x2f 0x31 0x2e 0x31
("radius/1.1")
Reference: This document

11. Acknowledgements

In hindsight, the decision to retain MD5 for RADIUS over TLS was likely wrong. It was an easy decision to make in the short term, but it has caused ongoing problems which this document addresses.

Thanks to Bernard Aboba, Karri Huhtanen, Heikki Vatiainen, Alexander Clouter, Michael Richardons, Hannes Tschofenig, and Matthew Netwon for reviews and feedback.

12. Changelog

draft-dekok-radext-sradius-00

Initial Revision

draft-dekok-radext-radiusv11-00

Use ALPN from RFC 7301, instead of defining a new port. Drop the name "SRADIUS".

Add discussion of Original-Packet-Code

draft-dekok-radext-radiusv11-01

Update formatting.

draft-dekok-radext-radiusv11-02

Add Flag field and description.

Minor rearrangements and updates to text.

draft-dekok-radext-radiusv11-03

Remove Flag field and description based on feedback and expected use-cases.

Use "radius/1.0" instead of "radius/1"

Consistently refer to the specification as "RADIUSv11", and consistently quote the ALPN name as "radius/1.1"

Add discussion of future attributes and future crypto-agility work.

draft-dekok-radext-radiusv11-04

Remove "radius/1.0" as it is unnecessary.

Update Introduction with more historical background, which motivates the rest of the section.

Change Identifier field to be reserved, as it is entirely unused.

Update discussion on clear text passwords.

Clarify discussion of Status-Server, User-Password, and Tunnel-Password.

Give high level summary of ALPN, clear up client / server roles, and remove "radius/1.0" as it is unnecessary.

Add text on RFC6421.

draft-dekok-radext-radiusv11-05

Clarify naming. "radius/1.1" is the ALPN name. "RADIUS/1.1" is the transport profile.

Clarify that future specifications do not need to make provisions for dealing with this transport profile.

draft-dekok-radext-radiusv11-05

Typos and word smithing.

Define and use "RADIUS over TLS" instead of RADIUS/(D)TLS.

Many cleanups and rework based on feedback from Matthew Newton.

13. References

13.1. Normative References

[BCP14] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC6421] Nelson, D., Ed., "Crypto-Agility Requirements for Remote Authentication Dial-In User Service (RADIUS)", RFC 6421, DOI 10.17487/RFC6421, November 2011, <https://www.rfc-editor.org/info/rfc6421>.

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

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, DOI 10.17487/RFC1321, April 1992, <https://www.rfc-editor.org/info/rfc1321>.

[RFC2548] Zorn, G., "Microsoft Vendor-specific RADIUS Attributes", RFC 2548, DOI 10.17487/RFC2548, March 1999, <https://www.rfc-editor.org/info/rfc2548>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC2868] Zorn, G., Leifer, D., Rubens, A., Shriver, J., Holdrege, M., and I. Goyret, "RADIUS Attributes for Tunnel Protocol Support", RFC 2868, DOI 10.17487/RFC2868, June 2000, <https://www.rfc-editor.org/info/rfc2868>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <https://www.rfc-editor.org/info/rfc3579>.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, DOI 10.17487/RFC5077, January 2008, <https://www.rfc-editor.org/info/rfc5077>.

[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>.

[RFC5281] Funk, P. and S. Blake-Wilson, "Extensible Authentication Protocol Tunneled Transport Layer Security Authenticated Protocol Version 0 (EAP-TTLSv0)", RFC 5281, DOI 10.17487/RFC5281, August 2008, <https://www.rfc-editor.org/info/rfc5281>.

[RFC5931] Harkins, D. and G. Zorn, "Extensible Authentication Protocol (EAP) Authentication Using Only a Password", RFC 5931, DOI 10.17487/RFC5931, August 2010, <https://www.rfc-editor.org/info/rfc5931>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC6218] Zorn, G., Zhang, T., Walker, J., and J. Salowey, "Cisco Vendor-Specific RADIUS Attributes for the Delivery of Keying Material", RFC 6218, DOI 10.17487/RFC6218, April 2011, <https://www.rfc-editor.org/info/rfc6218>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/info/rfc6613>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/info/rfc7360>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

[RFC7930] Hartman, S., "Larger Packets for RADIUS over TCP", RFC 7930, DOI 10.17487/RFC7930, August 2016, <https://www.rfc-editor.org/info/rfc7930>.

Author's Address

Alan DeKok
FreeRADIUS
Email: aland@freeradius.org

RADEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 January 2024

A. DeKok
FreeRADIUS
V. Cargatser
Cisco

27 July 2023

Reverse CoA in RADIUS
draft-dekok-radext-reverse-coa-01

Abstract

This document defines a "reverse change of authorization (CoA)" path for RADIUS packets. This specification allows a home server to send CoA packets in "reverse" down a RADIUS/TLS connection. Without this capability, it is impossible for a home server to send CoA packets to a NAS which is behind a firewall or NAT gateway. The reverse CoA functionality extends the available transport methods for CoA packets, but it does not change anything else about how CoA packets are handled.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekok-radext-reverse-coa/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/reverse-coa.git.

Status of This Memo

This Internet-Draft will expire on 28 January 2024.

Copyright Notice

Table of Contents

1. Introduction

[RFC5176] defines the ability to change a users authorization, or disconnect the user via what are generally called "Change of Authorization" or "CoA" packets. This term refers to either of the RADIUS packet types CoA-Request or Disconnect-Request. The initial transport protocol for all RADIUS was the User Datagram Protocol (UDP).

[RFC6614] updated previous specifications to allow packets to be sent over the Transport Layer Security (TLS) protocol. Section 2.5 of that document explicitly allows all packets (including CoA) to be sent over a TLS connection:

Due to the use of one single TCP port for all packet types, it is required that a RADIUS/TLS server signal which types of packets are supported on a server to a connecting peer. See also Section 3.4 for a discussion of signaling.

These specifications assume that a RADIUS client can directly contact a RADIUS server, which is the normal "forward" path for packets between a client and server. However, it is not always possible for the RADIUS server to send CoA packets to the RADIUS client. If a RADIUS server wishes to act as a CoA client, and send CoA packets to the NAS (CoA server), the "reverse" path can be blocked by a firewall, NAT gateway, etc. That is, a RADIUS server has to be reachable by a NAS, but there is usually no requirement that the NAS is reachable from a public system. To the contrary, there is usually a requirement that the NAS is not publicly accessible.

This scenario is most evident in a roaming / federated environment such as Eduroam or OpenRoaming. It is in general impossible for a home server to signal the NAS to disconnect a user. There is no direct reverse path from the home server to the NAS, as the NAS is not publicly addressible. Even if there was a public reverse path, it would generally be unknowable, as intermediate proxies can (and do) attribute rewriting to hide NAS identies.

These limitations can result in business losses and security problems, such as the inability to disconnect an online user when their account has been terminated.

As the reverse path is usally blocked, it means that it is in general possible only to send CoA packets to a NAS when the NAS and RADIUS server share the same private network (private IP space or IPSec). Even though [RFC8559] defines CoA proxying, that specification does not address the issue of NAS reachability.

This specification solves that problem. The solution is to simply allow CoA packets to go in "reverse" down an existing RADIUS/TLS connection. That is, when a NAS connects to a RADIUS server it normally sends request packets (Access-Request, etc.) and expects to receive response packets (Access-Accept, etc.). This specification extends RADIUS/TLS by permitting a RADIUS server to re-use an existing TLS connection to send CoA packets to the NAS, and permitting the NAS to send CoA response packets to the RADIUS server over that same connection.

We note that while this document specifically mentions RADIUS/TLS, it should be possible to use the same mechanisms on RADIUS/DTLS [RFC7360]. However at the time of writing this specification, no implementations exist for "reverse CoA" over RADIUS/DTLS. As such, when we refer to "TLS" here, or "RADIUS/TLS", we implicitly include RADIUS/DTLS in that description.

We also note that while this same mechanism could theoretically be used for RADIUS/UDP and RADIUS/TCP, there is no value in defining "reverse CoA" for those transports. Therefore for practial purposes, "reverse CoA" means RADIUS/TLS and RADIUS/DTLS.

There are additional considerations for proxies. While [RFC8559] describes CoA proxying, there are still issues which need to be addressed for the "reverse CoA" use-case. This specification describes how a proxy can implement "reverse CoA" proxying, including signalling necessary to negotiate this functionality.

2. Terminology

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.

* CoA

  Change of Authorization packets. For brevity, when this document refers to "CoA" packets, it means either or both of CoA-Request and Disconnect-Request packets.

* ACK

  Change of Authorization "positive acknowlegement" packets. For brevity, when this document refers to "ACK" packets, it means either or both of CoA-ACK and Disconnect-ACK packets.

* NAK

  Change of Authorization "negative acknowlegement" packets. For brevity, when this document refers to "ACK" packets, it means either or both of CoA-NAK and Disconnect-NAK packets.

* RADIUS/TLS

  RADIUS over the Transport Layer Security protocol [RFC6614]

* RADIUS/DTLS

  RADIUS over the Datagram Transport Layer Security protocol [RFC7360]

* TLS

  Either RADIUS/TLS or RADIUS/DTLS.

* reverse CoA

  CoA, ACK, or NAK packets sent over a RADIUS/TLS or RADIUS/DTLS connection which was made from a RADIUS client to a RADIUS server.

3. Concepts

The reverse CoA functionality is based on two additions to RADIUS. The first addition is a configuration and signalling, to indicate that a RADIUS client is capable of accepting reverse CoA packets. The second addition is an extension to the "reverse" routing table for CoA packets which was first described in Section 2.1 of [RFC8559].

4. Capability Configuration and Signalling

In order for a RADIUS server to send reverse CoA packets to a client, it must first know that the client is capable of accepting these packets.

This functionality can be enabled in one of two ways. The first is a simple static configuration between client and server, where both are configured to allow reverse CoA. The second method is via per-connection signalling between client and server.

The server manages this functionality with two boolean flags, one per-client, and one per-connection. The per-client flag can be statically configured, and if not present MUST be treated as having a "false" value. The per-connection flag MUST be initialized from the per-client flag, and then can be dynamically negotiated after that.

4.1. Configuration Flag

Clients and servers implementing reverse CoA SHOULD have a configuration flag which indicates that the other party supports the reverse CoA functionality. That is, the client has a per-server flag enabling (or not) reverse CoA functionality. The server has a similar per-client flag.

The flag can be used where the parties are known to each other. The flag can also be used in conjunction with dynamic discovery ([RFC7585]), so long as the server associates the flag with the client identity and not with any particular IP address. That is, the flag can be associated with any method of identifying a particular client such as TLS-PSK identity, information in a client certificate, etc.

For the client, the flag controls whether or not it will accept reverse CoA packets from the server, and whether the client will do dynamic signalling of the reverse CoA functionality.

Separately, each side also needs to have a per-connection flag, which indicates whether or not this connection supports reverse CoA. The per-connection flag is initialized from the static flag, and is then dynamically updated after that.

4.2. Dynamic Signalling

The reverse CoA functionality can be signalled on a per-connection basis by the client sending a Status-Server packet when it first opens a connection to a server. This packet contains a Capability attribute (see below), with value "Reverse-CoA". The existence of this attribute in a Status-Server packet indicates that the client supports reverse CoA over this connection. The Status-Server packet MUST be the first packet sent when the connection is opened, in order to perform per-connection signalling. A server which does not implement reverse CoA simply ignores this attribute, as per [RFC2865] Section 5.

A server implementing reverse CoA does not need to signal the NAS in any response, to indicate that it is supports reverse CoA. If the server never sends reverse CoA packets, then such signalling is unnecessary. If the server does send reverse CoA packets, then the packets themselves serve as sufficiant signalling.

The NAS may send additional Status-Server packets down the same connection, as per [RFC3539]. These packets do not need to contain the Capability attribute, so it can generally be omitted. That is, there is no need to signal the addition or removal of reverse CoA functionality during the lifetime of one connection. If a client decides that it no longer wants to support reverse CoA on a particular connection, it can simply tear down the connection, and open a new one which does not negotiate the reverse CoA functionality.

RADIUS client implementations which support reverse CoA MUST always signal that functionality in a Status-Server packet on any new connection. There is little reason to save a few octets, and having explicit signalling can help with implementations, deployment, and debugging.

The combination of static configuration and dynamic configuration means that it is possible for client and server to both agree on whether or not a particular connection supports reverse CoA.

5. Reverse Routing

The "reverse" routing table for CoA packets was first described in Section 2.1 of [RFC8559]. We extend that table here.

In our extension, the table does not map realms to home servers. Instead, it maps keys to connections. The keys will be defined in more detail below. For now, we say that keys can be derived from a RADIUS client to server connection, and from the contents of a CoA packet which needs to be routed.

When the server receives a TLS connection from a client, it derives a key for that connection, and associates the connection with that key. A server MUST support associating one particular key value with multiple connections. A server MUST support associating multiple keys for one connection. That is, the "key to connection" mapping is N to M. It is not one-to-one, or 1-N, or M-1.

When the server receives a CoA packet, it derives a key from that packet, and determines if there is a connection or connections which maps to that key. Where there is no available connection, the server MUST return a NAK packet that contains an Error-Cause Attribute having value 502 ("Request Not Routable").

As with normal proxying, a particular packet can sometimes have the choice more than one connection which can be used to reach a destination. In that case, issues of load-balancing, fail-over, etc. are implementation-defined, and are not discussed here. The server simply chooses one connection, and sends the reverse CoA packet down that connection.

The server then waits for a reply, doing retransmission if necessary. For all issues other than the connection being used, reverse CoA packets are handled as defined in [RFC5176] and in [RFC8559].

That is, when the NAS and server are known to each other, [RFC5176] is followed when sending CoA packets to the NAS. The difference is that instead of originating connections to the NAS, the server simply re-uses inbound TLS connections from the NAS. The NAS is identified by attributes such as NAS-Identifier, NAS-IP-Address, and NAS-IPv6-Address.

When a server is proxying to another server, [RFC8559] is following when proxying CoA packets. The "next hop" is identified either by Operator-Name for proxy-to-proxy connections. When the CoA packet reaches a visited network, that network identifies the NAS by examining the Operator-NAS-Identifier attribute.

5.1. Retransmits

Retransmissions of reverse CoA packets are handled identically to normal CoA packets. That is, the reverse CoA functionality extends the available transport methods for CoA packets, it does not change anything else about how CoA packets are handled.

6. Implementation Status

FreeRADIUS supports CoA proxying using Vendor-Specific attributes.

Cisco supports reverse CoA as of Cisco IOS XE Bengaluru 17.6.1 via Vendor-Specific attributes.
https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/17-6/configuration_guide/sec/b_176_sec_9300_cg/configuring_radsec.pdf

Aruba documentation states that "Instant supports dynamic CoA (RFC 3576) over RadSec and the RADIUS server uses an existing TLS connection opened by the Instant AP to send the request." https://www.arubanetworks.com/techdocs/Instant_83_WebHelp/Content/Instant_UG/Authentication/ConfiguringRadSec.htm

7. Privacy Considerations

This document does not change or add any privacy considerations over previous RADIUS specifications.

8. Security Considerations

This document increases network security by removing the requirement for non-standard "reverse" paths for CoA-Request and Disconnect-Request packets.

9. IANA Considerations

This document requests no action from IANA.

RFC Editor: This section may be removed before publication.

10. Acknowledgements

Thanks to Heikki Vatiainen for testing a preliminary implementation in Radiator, and for verifying interoperability with NAS equipment.

11. Changelog

12. References

12.1. Normative References

[BCP14] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <https://www.rfc-editor.org/info/rfc3539>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8559] DeKok, A. and J. Korhonen, "Dynamic Authorization Proxying in the Remote Authentication Dial-In User Service (RADIUS) Protocol", RFC 8559, DOI 10.17487/RFC8559, April 2019, <https://www.rfc-editor.org/info/rfc8559>.

12.2. Informative References

[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>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/info/rfc7360>.

Authors' Addresses

Alan DeKok
FreeRADIUS
Email: aland@freeradius.org

Vadim Cargatser
Cisco
Email: vcargats@cisco.com

RADEXT Working Group
Internet-Draft
Intended status: Informational
Expires: 7 January 2024

A. DeKok
FreeRADIUS

6 July 2023

RADIUS and TLS-PSK
draft-dekok-radext-tls-psk-01

Abstract

This document gives implementation and operational considerations for using TLS-PSK with RADIUS/TLS (RFC6614) and RADIUS/DTLS (RFC7360).

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekok-radext-tls-psk/.

Discussion of this document takes place on the RADEXT Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/.

Source for this draft and an issue tracker can be found at https://github.com/freeradius/radext-tls-psk.git.

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 7 January 2024.

Table of Contents

1. Introduction

The previous specifications "Transport Layer Security (TLS) Encryption for RADIUS" [RFC6614] and " Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS" [RFC7360] defined how (D)TLS can be used as a transport protocol for RADIUS. However, those documents do not provide guidance for using TLS-PSK with RADIUS. This docoument provides that missing guidance, and gives implementation and operational considerations.

2. Terminology

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.

TBD

3. History

TLS deployments usually rely on certificates in most common uses. However, we recognize that it may be difficult to fully upgrade client implementations to allow for certificates to be used with RADIUS/TLS and RADIUS/DTLS. These upgrades involve not only implementing TLS, but can also require significant changes to administration interfaces and application programming interfaces (APIs) in order to fully support certificates.

For example, unlike shared secrets, certificates expire. This expiration means that a working system using TLS can suddenly stop working. Managing this expiration can require additional notification APIs on RADIUS clients and servers which were previously not required when shared secrets were used.

Certificates also require the use of certification authorities (CAs), and chains of certificates. RADIUS implementations using TLS therefore have to track not just a small shared secret, but also potentially many large certificates. The use of TLS-PSK can therefore provide a simpler upgrade path for implementations to transition from RADIUS shared secrets to TLS.

4. General Discussion of PSKs and PSK Identies.

Before we define any RADIUS-specific use of PSKs, we must first review the current standards for PSKs, and give general advice on PSKs and PSK identies.

The requirements in this section apply to both client and server implementations which use TLS-PSK. Client-specific and server-specific issues are discussed in more detail later in this document.

4.1. Requirements on PSKs

Reuse of a PSK in multiple versions of TLS (e.g. TLS 1.2 and TLS 1.3) is considered unsafe ([RFC8446] Section E.7). Where TLS 1.3 binds the PSK to a particular key deriviation function, TLS 1.2 does not. This binding means that it is possible to use the same PSK in different hashes, leading to the potential for attacking the PSK by comparing the hash outputs. While there are no known insecurities, these uses are not known to be secure, and should therefore be avoided.

[RFC9258] adds a key derivation function to the import interface of (D)TLS 1.3, which binds the externally provided PSK to the protocol version. In particular, that document:

> ... describes a mechanism for importing PSKs derived from external PSKs by including the target KDF, (D)TLS protocol version, and an optional context string to ensure uniqueness. This process yields a set of candidate PSKs, each of which are bound to a target KDF and protocol, that are separate from those used in (D)TLS 1.2 and prior versions. This expands what would normally have been a single PSK and identity into a set of PSKs and identities.

If an implementation supports both TLS 1.2 and TLS 1.3, it MUST require that TLS 1.3 be negotiated in RADIUS/TLS and RADIUS/DTLS. This requirement prevents reuse of a PSK with multiple TLS versions, which prevents the attacks discussed in [RFC8446] Section E.7.

It is RECOMMENDED that systems follow the directions of [RFC9257] Section 4 for the use of external PSKs in TLS. That document provides extremely useful guidance on generating and using PSKs.

Implementations MUST support PSKs of at least 32 octets in length, and SHOULD support PSKs of 64 octets. Implementations MUST require that PSKs be at least 16 octets in length. That is, short PSKs MUST NOT be permitted to be used.

Administrators SHOULD use PSKs of at least 24 octets, generated using a source of cryptographically secure random numbers. Implementors needing a secure random number generator should see [RFC8937] for for further guidance. PSKs are not passwords, and administrators should not try to manually create PSKs.

Passwords are generally intended to be remembered and entered by people on a regular basis. In contrast, PSKs are intended to be entered once, and then automatically saved in a system configuration. As such, due to the limited entropy of passwords, they are not acceptable for use with TLS-PSK, and would only be acceptable for use with a password-authenticated key exchange (PAKE) TLS method.

We also incorporate by reference the requirements of Section 10.2 of [RFC7360] when using PSKs.

4.1.1. Interaction between PSKs and Shared Secrets

Any shared secret used for RADIUS/UDP or RADIUS/TLS MUST NOT be used for TLS-PSK.

It is RECOMMENDED that RADIUS clients and server track all used shared secrets and PSKs, and then verify that the following requirements all hold true:

* no shared secret is used for more than one RADIUS client

* no PSK is used for more than one RADIUS client

* no shared secret is used as a PSK

* no PSK is used as a shared secret

There may be use-cases for using one shared secret across multiple RADIUS clients. There may similarly be use-cases for sharing a PSK across multiple RADIUS clients. Details of the possible attacks on reused PSKs are given in [RFC9257] Section 4.1.

There are few, if any, use-cases for using a PSK as a shared secret, or vice-versa.

Implementaions MUST NOT provide user interfaces which allow both PSKs and shared secrets to be entered at the same time. Only one or the other must be present. Implementations MUST NOT use a "shared secret" field as a way for administrators to enter PSKs. The PSK entry fields MUST be labelled as being related to PSKs, and not to shared secrets.

4.2. PSK Identities

It is RECOMMENDED that systems follow the directions of [RFC9257] Section 6.1.1 for the use of external PSK identies in TLS. Note that the PSK identity is sent in the clear, and is therefore visible to attackers. Where privacy is desired, the PSK identity could be either an opaque token generated cryptographically, or perhaps in the form of a Network Access Identifier (NAI) [RFC7542], where the "user" portion is an opaque token. For example, an NAI could be "68092112@example.com". If the attacker already knows that the client is associated with "example.com", then using that domain name in the PSK identity offers no additional information. In constrast, the "user" portion needs to be both unique to the client and private, so using an opaque token there is a more secure approach.

Implementations MUST support PSK identies of 128 octets, and SHOULD support longer PSK identities. We note that while TLS provides for PSK identities of up to 2^16-1 octets in length, there are few practical uses for extremely long PSK identities.

4.3. PSK and PSK Identity Sharing

While administrators may desire to share PSKs and/or PSK identities across multiple systems, such usage is NOT RECOMMENDED. Details of the possible attacks on reused PSKs are given in [RFC9257] Section 4.1.

Implementations MUST support configuring a unique PSK and PSK identity for each possible client-server relationship. This configuration allows administrators desiring security to use unique PSKs for each such relationship. This configuration also allows administrators to re-use PSKs and PSK identies where local policies permit.

Implementations SHOULD warn administrators if the same PSK identity and/or PSK is used for multiple client-server relationships.

5. Guidance for RADIUS clients

TLS uses certificates in most common uses. However, we recognize that it may be difficult to fully upgrade client implementations to allow for certificates to be used with RADIUS/TLS and RADIUS/DTLS. Client implementations therefore MUST allow the use of a pre-shared key (TLS-PSK). The client implementation can then expose a flag "TLS yes / no", and then fields which ask for the PSK identity and PSK itself.

Implementations MUST use ECDH cipher suites. Implementations MUST implement the recommended cipher suites in [RFC9325] Section 4.2 for TLS 1.2, and in [RFC9325] Section 4.2 for TLS 1.3.

5.1. PSK Identities

[RFC6614] is silent on the subject of PSK identities, which is an issue that we correct here. Guidance is required on the use of PSK identities, as the need to manage identities associated with PSK is a new requirement for NAS management interfaces, and is a new requirement for RADIUS servers.

RADIUS systems implementing TLS-PSK MUST support identities as per [RFC4279] Section 5.3, and MUST enable configuring TLS-PSK identities in management interfaces as per [RFC4279] Section 5.4.

RADIUS shared secrets cannot safely be used as TLS-PSKs. To prevent confusion between shared secrets and TLS-PSKs, management interfaces and APIs need to label PSK fields as "PSK" or "TLS-PSK", rather than "shared secret

Where dynamic server lookups [RFC7585] are not used, RADIUS clients MUST still permit the configuration of a RADIUS server IP address.

6. Guidance for RADIUS Servers

The following section(s) describe guidance for RADIUS server implementations and deployments.

6.1. Identifying and filtering clients

RADIUS/UDP and RADIUS/TCP identify clients by source IP address. This practice is no longer needed when TLS transport is used, as the client can instead be identified via TLS information such as PSK identity, client certificate, etc.

When a RADIUS server implements TLS-PSK, it MUST use the PSK identity as the logical identifier for a RADIUS client instead of the IP address as was done with RADIUS/UDP. That is, instead of associating a source IP address with a shared secret, the RADIUS server instead associates a PSK identity with a pre-shared key. In effect, the PSK identity replaces the source IP address of the connection as the client identifier.

For example, when a RADIUS server receives a RADIUS/UDP packet, it normally looks up the source IP address, finds a client definition, and that client definition contains a shared secret. The packet is then authenticated (or not) using that shared secret.

When TLS-PSK is used, the RADIUS server instead receives a TLS connection request which contains a PSK identity. That identity is then used to find a client definition, and that client definition contains a PSK. The TLS connection is then authenticated (or not) using that PSK.

Each RADIUS client MUST be configured with a unique PSK, which implies a unique PSK identifier for each RADIUS client. To enforce the use of unique PSKs, RADIUS servers accepting TLS-PSK MUST require that a PSK identifier and PSK can be associated with each RADIUS client.

RADIUS servers MUST be able to look up PSK identity in a subsystem which then returns the actual PSK.

RADIUS servers MUST support IP address and network filtering of the source IP address for all TLS connections. In many situations a RADIUS server does not need to allow connections from the entire Internet. As such, it can increase security to limit permitted connections to a small list of networks.

For example, a RADIUS server be configured to be accept connections from a source network of 192.0.2/24. The RADIUS server could therefore discard any TLS connection request which comes from a source IP address outside of that network. In that case, there is no need to examine the PSK identity or to find the client definition. Instead, the IP source filtering policy would deny the connection before any TLS communication had been performed.

RADIUS servers SHOULD be able to limit certain PSK identifiers to certain network ranges or IP addresses. This filtering can catch configuration errors. That is, if a NAS is known to have a dynamic IP address within a particular subnet, the server should limit use of the NASes PSK to that subnet.

For example, as with the example above, the RADIUS server be configured to be accept connections from a source network of 192.0.2/24. The RADIUS server may be configured to with a PSK idrnity "system1", and then also configured to associate that PSK identity with the source IP address 192.0.2.16. In that case, if the server receives a connection request from the source IP address 192.0.2.16 with PSK identity other than "system1", then the connection could be rejected. Similarly, if the server receives a connection request from the source IP address other than 192.0.2.16 but which uses the PSK identity "system1", then the connection could also be rejected.

The use of PSK identities as client identifiers does not prevent RADIUS servers from performing source IP filtering of incoming packets or connections. Instead, the use of PSK identities as client identifiers means that source IP addresses are no longer required to be associated with RADIUS clients.

Note that as some clients may have dynamic IP addresses, it is possible for a one PSK identity to appear at different source IP addresses over time. In addition, as there may be many clients behind one NAT gateway, there may be multiple RADIUS clients using one public IP address. RADIUS servers MUST support multiple PSKs at one source IP address, and MUST support a unique PSK identity for each unique client which is deployed in such a scenario.

In those use-cases, the RADIUS server should either not use source IP address filtering, or should apply source IP filtering rules which permit those use-cases. This filtering must therefore be flexible to allow all of the above behaviors, and be configurable by administrators to match their needs.

RADIUS servers SHOULD tie PSK identities to a particular permitted IP address or permitted network, as doing so will lower the risk if a PSK is leaked. RADIUS servers MUST permit multiple clients to share one permitted IP address or network.

7. Privacy Considerations

We make no changes over [RFC6614] and [RFC7360].

8. Security Considerations

The primary focus of this document is addressing security considerations for RADIUS.

9. IANA Considerations

There are no IANA considerations in this document.

RFC Editor: This section may be removed before final publication.

10. Acknowledgements

TBD.

11. Changelog

* 00 - initial version

* 01 - update examples

12. References

12.1. Normative References

[BCP14] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC4279] Eronen, P., Ed. and H. Tschofenig, Ed., "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, DOI 10.17487/RFC4279, December 2005, <https://www.rfc-editor.org/info/rfc4279>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

12.2. Informative References

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/info/rfc6613>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/info/rfc7360>.

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

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/info/rfc8937>.

[RFC9257] Housley, R., Hoyland, J., Sethi, M., and C. A. Wood, "Guidance for External Pre-Shared Key (PSK) Usage in TLS", RFC 9257, DOI 10.17487/RFC9257, July 2022, <https://www.rfc-editor.org/info/rfc9257>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

Author's Address

Alan DeKok
FreeRADIUS
Email: aland@freeradius.org

RADEXT Working Group
Internet-Draft
Intended status: Standards Track
Expires: 11 January 2024

M. Grayson
E. Lear
Cisco Systems
10 July 2023

RADIUS profile for Bonded Bluetooth Low Energy peripherals
draft-grayson-radext-rabble-01

Abstract

This document specifies an extension to the Remote Authentication Dial-In User Service (RADIUS) protocol that enables a Bluetooth Low Energy (BLE) peripheral device that has previously formed a bonded, secure trusted relationship with a first "home" Bluetooth Low Energy Central device to operate with a second "visited" Bluetooth Low Energy Central device.

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 11 January 2024.

Copyright Notice

Table of Contents

1. Introduction

This document specifies an extension to the Remote Authentication Dial-In User Service (RADIUS) protocol [RFC2865] that enables a Bluetooth Low Energy (BLE) peripheral device that has previously formed a bonded, secure trusted relationship with a first "home" Bluetooth Low Energy Central device to operate with a second "visited" Bluetooth Low Energy Central device that is integrated with a Network Access Server.

After being successfully authenticated, a signalling link is established that enables Bluetooth messages advertised by the BLE Peripheral to be forwarded from the Visited Bluetooth Low Energy Central device to a Home MQTT Broker. For connectable BLE Peripherals, the signalling link enables the Home MQTT Broker to send BLE Requests or Commands to the Visited Bluetooth Low Energy Central device that is then responsible for forwarding to the BLE peripheral.

The extensions allow administrative entities to collaborate to enable RADIUS authentication of BLE devices onto their respective networks, without requiring the peripheral to perform a re-pairing on the visited network.

1.1. 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.

1.2. Terminology

BLE Central Controller:

The BLE entity that implements the Bluetooth Link Layer and interacts with the Bluetooth Radio Hardware.

BLE Central Host:

A BLE entity that interacts with the BLE Central Controller to enable applications to communicate with peer BLE devices in a standard and interoperable way.

BLE Peripheral Device:

A BLE device that is configured to repeatedly send advertising messages.

BLE Security Database:

A database that stores the keying material associated with a bonded Bluetooth Connection.

Bluetooth Low Energy (BLE):

A wireless technology designed for low power operation and specified by the Bluetooth Special Interest Group.

Bonding:

A Bluetooth [BLUETOOTH] defined process that creates a relation between a Bluetooth Central device and a Bluetooth Peripheral device and which generates session keying material that is expected to be stored by both Bluetooth devices, to be used for future authentication.

Hash:

A Bluetooth [BLUETOOTH] specified 24-bit hash value which is calculated using a hash function operating on IRK and prand as its input parameters. The hash is encoded in the 24 least significant bits of a Resolvable Private Address.

Home:

A network that has access to the keying material necessary to support the pairing of a BLE peripheral and that is able to expose the keys generated as part of the BLE bonding process.

Identity Address (IA):

The 48-bit global (public) MAC address of a Bluetooth device.

Identity Resolving Key (IRK):

A Bluetooth [BLUETOOTH] specified key used in the Bluetooth privacy feature. The Resolvable Private Address hash value is calculated using a hash function of prand and the IRK.

Long-Term key (LTK):

A symmetric key which is generated during the Bluetooth bonding procedure and used to generate the session key used to encrypt a communication session between Bluetooth devices.

prand:

A 22-bit random number used by a BLE device to generate a Resolvable Private Address. The prand is encoded in the 24 most significant bits of a Resolvable Private Address.

Resolvable Private Address (RPA):

A Bluetooth [BLUETOOTH] specified private 48-bit address that can be resolved to a permanent Bluetooth Identity Address through the use of an Identity Resolving Key.

Visited:

A network that does not have access to the keying material necessary to support the pairing of a BLE peripheral, but that is able to support the RADIUS authentication of an already bonded BLE Peripheral.

2. BLE Roaming Overview

This section provides an overview of the RADIUS BLE mechanism, which is supported by the extensions described in this document. The RADIUS profile is intended to be used between a Visited BLE Central Host that is enhanced with Network Access Server (NAS) functionality which enables it to exchange messages with a RADIUS server.

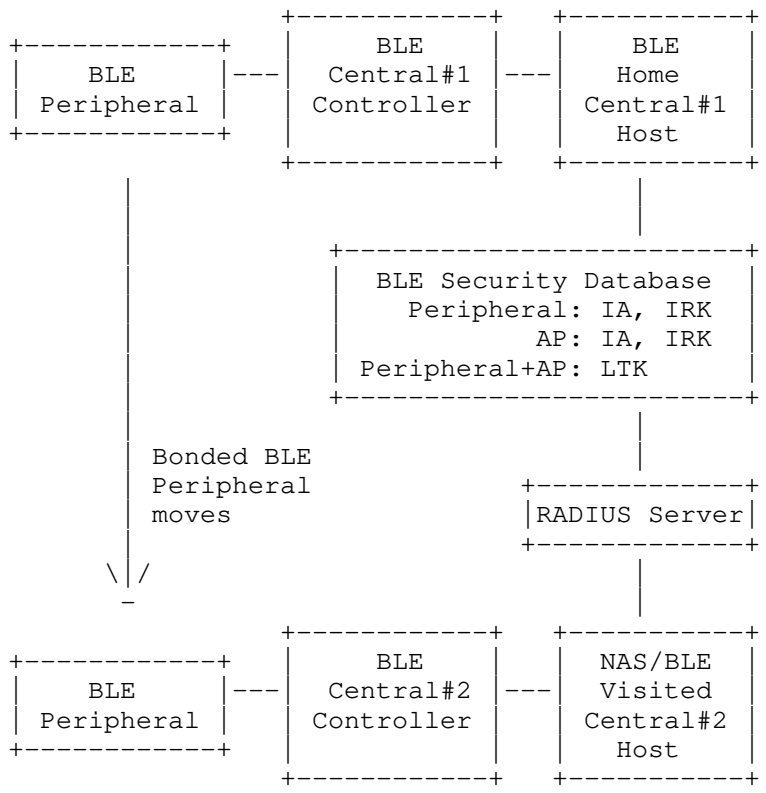


Figure 1: BLE RADIUS Authentication Overview

A BLE Peripheral is paired and bonded with the BLE Home Central Host. The pairing requires the BLE Home Central Host to have access to the keying material necessary to support the pairing of a BLE peripheral, e.g., by using techniques described in [I-D.shahzad-scim-device-model].

The bonding process generates new session specific keying material that MUST be exposed by the BLE Home Central Host to a RADIUS server, e.g., stored in a BLE Security Database which is accessible by the RADIUS server. The keying material MUST include the peripheral's IA and IRK, indicating that the BLE Peripheral has enabled the Bluetooth privacy feature and is operating with a Resolvable Private Address (RPA).

The BLE Peripheral then moves into the coverage of a second BLE Central device which comprises a second BLE Central Controller and a second BLE (Visited) Central Host which has been enhanced with Network Access Server (NAS) functionality. The BLE Peripheral MUST

be configured to send low duty cycle advertising events using the BLE Peripheral's RPA that are detected by the NAS/BLE Visited Central Host. The NAS/BLE Visited Central Host receives the Advertisement(s) sent by the BLE Peripheral and MAY use the presence and/or contents of specific Advertising Elements to decide whether to trigger a RADIUS exchange with a RADIUS Server which has access to the keying material exposed by the BLE Home Central Host.

The successful authentication of the BLE Peripheral onto the BLE Visited Central Host MUST include the signalling of the keying material exposed by the BLE Home Central Host to enable the re-establishment of the secured communication session with the BLE Peripheral. Bluetooth advertisements received from an authenticated BLE Peripheral are forwarded between the BLE Visited Central Host and a Home MQTT message broker.

If the BLE Peripheral is connectable, the Home MQTT Broker MAY send BLE Requests or Commands to the Visited Bluetooth Low Energy Central device that is then responsible for forwarding to the authenticated BLE peripheral. The Home MQTT Broker MAY be configured to forward the messages to/from a Bluetooth Application associated with the authenticated BLE Peripheral, either directly, or via the first Home Bluetooth Low Energy Central device.

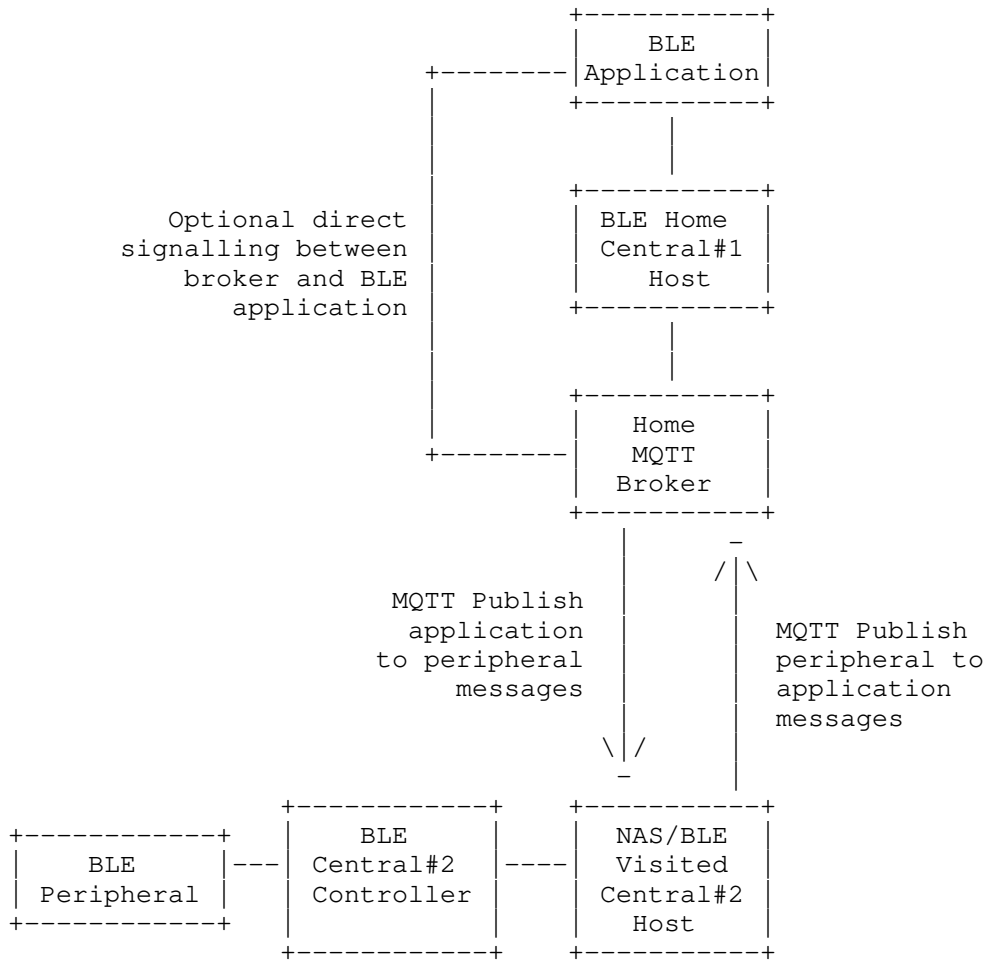


Figure 2: BLE Message Forwarding Overview

3. RADIUS Profile for BLE

3.1. User-Name

Contains a 6 character ASCII upper-case string corresponding to the hexadecimal encoding of the 22-bit prand value derived from the Bluetooth Resolvable Private Address, where the first string character represents the most significant hexadecimal digit, i.e., a prand value of 0x035fb2 is encoded as "035FB2".

3.2. NAS-IP-Address, NAS-IPv6-Address

The NAS-IP-Address contains the IPv4 address of the BLE Central Host acting as an Authenticator, and the NAS-IPv6-Address contains the IPv6 address.

3.3. NAS-Port

For use with BLE the NAS-Port will contain the port number of the BLE Central Host, if this is available.

3.4. Service-Type

For use with BLE, the Service-Type of Authenticate Only (8) is used.

3.5. State, Class, Proxy-State

These attributes are used for the same purposes as described in [RFC2865].

3.6. Vendor-Specific

Vendor-specific attributes are used for the same purposes as described in [RFC2865].

3.7. Session-Timeout

When sent in an Access-Accept without a Termination-Action attribute or with a Termination-Action attribute set to Default, the Session-Timeout attribute specifies the maximum number of seconds of service provided prior to session termination.

3.8. Idle-Timeout

The Idle-Timeout attribute indicates the maximum time that the BLE wireless device may remain idle.

3.9. Termination-Action

This attribute indicates what action should be taken when the service is completed. The value Default (0) indicates that the session should terminate.

3.10. Called-Station-Id

This attribute is used to store the public Identity Address (BD_ADDR) of the Bluetooth Access Point in ASCII formatted as specified in section 3.21 of [RFC3580].

3.11. NAS-Identifier

This attribute contains a string identifying the BLE Central Host originating the Access-Request.

3.12. NAS-Port-Type

TBA1: "Wireless - Bluetooth Low Energy"

3.13. Hashed-Password

Description

The Hashed-Password (TBA2) Attribute allows a RADIUS client to include a key and hashed password.

Type

TBA2

Length

Variable

Data Type

TLV

Value

The TLV data type is specified in section 3.13 of [RFC8044] and its value is determined by the TLV-Type field. Two TLV-Types are defined for use with the Hashed-Password Attribute.

3.13.1. Hashed-Password.Hmac-Sha256-128-Key

TLV-Type

0 (Hashed-Password.Hmac-Sha256-128-Key)

TLV-Value:

A string data type, as defined in section 3.1 of [RFC8044], encoding a sequence of octets representing a random 256-bit key. The value SHOULD satisfy the requirements of [RFC4086]. A new key value MUST be used whenever the value of Hashed-Password.Hmac-Sha256-128-Password is changed. The key MUST NOT be changed when a message is being retransmitted.

TLV-Length:

34 octets

3.13.2. Hashed-Password.Hmac-Sha256-128-Password

TLV-Type

1 (Hashed-Password.Hmac-Sha256-128-Password)

TLV-Value:

A string data type encoding a sequence of octets representing the first 128-bit (truncated) output of the HMAC-SHA-256-128 algorithm [RFC4868] where the input data corresponds to the 24-bit hash recovered from the Bluetooth Resolvable Private Address and the key corresponds to the value of the TLV-Type Hashed-Password.Hmac-Sha256-128-Key.

TLV-Length:

18 octets

3.13.3. Hashed-Password TLV-Type Usage

Two instances of the Hashed-Password Attribute MUST be included in an Access-Request packet. One instance MUST correspond to the TLV-Type 0 (Hashed-Password.Hmac-Sha256-128-Key) and one instance MUST correspond to the TLV-Type 1 (Hashed-Password.Hmac-Sha256-128-Password).

3.14. GATT-Service-Profile

Description

The GATT-Service-Profile (TBA3) Attribute allows a RADIUS client to include one or more GATT Service Profiles which are advertised by the BLE Peripheral.

Zero or more GATT-Service-Profile Attributes MAY be included in an Access-Request packet.

A summary of the GATT-Service-Profile Attribute format is shown below. The fields are transmitted from left to right.

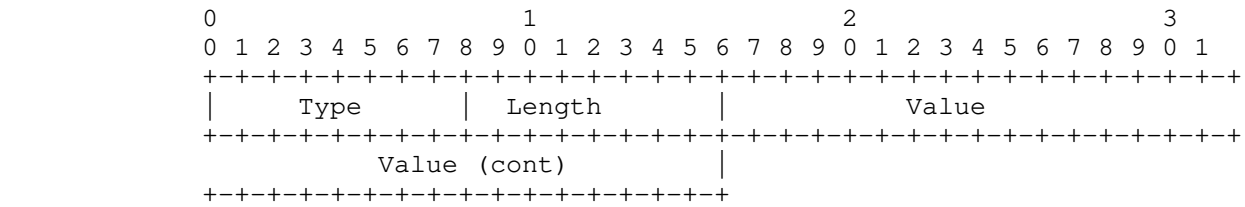


Figure 3: Encoding GATT-Service-Profile Attribute

Type

TBA3

Length

6 octet

Data Type

Integer

Value

The field is 4 octets, containing a 32-bit unsigned integer that represents a GATT Service Profile.

3.15. BLE-Keying-Material Attribute

Description

The BLE-Keying-Material (TBA3) Attribute allows the transfer of Identity Address(es) and cryptographic keying material from a RADIUS Server to the BLE Visited Central Host.

Type

TBA3

Length

Variable

Data Type

TLV

Value

The TLV data type is specified in section 3.13 of [RFC8044] and its value is determined by the TLV-Type field. Five TLV-Types are defined for use with the BLE-Keying-Material Attribute.

3.15.1. BLE-Keying-Material.Peripheral-IA

TLV-Type

0 (BLE-Keying-Material.Peripheral-IA)

TLV-Value:

A string data type encoding a sequence of octets representing the Peripheral's 6-octet Identity Address.

TLV-Length:

8 octets

3.15.2. BLE-Keying-Material.Central-IA

TLV-Type

1 (BLE-Keying-Material.Central-IA)

TLV-Value:

A string data type encoding a sequence of octets representing the Central's 6-octet Identity Address.

TLV-Length:

8 octets

3.15.3. BLE-Keying-Material.IV

TLV-Type

2 (BLE-Keying-Material.IV)

TLV-Value:

A string data type encoding a sequence of octets representing an 8-octet initial value (IV). The value MUST be as specified in section 2.2.3 of [RFC3394].

TLV-Length:

10 octets

3.15.4. BLE-Keying-Material.KEK-ID

TLV-Type

3 (BLE-Keying-Material.KEK-ID)

TLV-Value:

A string data type encoding a sequence of octets representing the identity of a Key Encryption Key (KEK). The combination of the BLE-Keying-Material.KEK-ID value and the RADIUS client and server IP addresses together uniquely identify a key shared between the RADIUS client and server. As a result, the BLE-Keying-Material.KEK-ID need not be globally unique. The BLE-Keying-Material.KEK-ID MUST refer to an encryption key for use with the AES Key Wrap with 128-bit KEK algorithm [RFC3394].
This key is used to protect the contents of the BLE-Keying-Material.KM-Data TLV (see Section 3.15.6).

The BLE-Keying-Material.KEK-ID is a constant that is configured through an out-of-band mechanism. The same value is configured on both the RADIUS client and server. If no BLE-Keying-Material.KEK-ID TLV-Type is signalled, then the field is set to 0. If only a single KEK is configured for use between a given RADIUS client and server, then 0 can be used as the default value.

TLV-Length:

18 octets

3.15.5. BLE-Keying-Material.KM-Type

TLV-Type:

4 (BLE-Keying-Material.KM-Type)

TLV-Value:

An integer data type identifying the type of keying material included in the BLE-Keying-Material.KM-Data TLV.
This allows for multiple keys for different purposes to be present in the same attribute. This document defines three values for the The BLE-Keying-Material.KM-Type

0 The BLE-Keying-Material.KM-Data TLV contains the 16-octet Peripheral IRK encrypted using the AES key wrapping process with 128-bit KEK defined in [RFC3394]. The Peripheral IRK is passed as input P1 and P2, with the plaintext P1 corresponding to octet 0 through to octet 7 of the IRK and plaintext P2 corresponding to octet 8 through to octet 15 of the IRK.

1 The BLE-Keying-Material.KM-Data TLV contains the encrypted 16-octet Peripheral IRK and the 16-octet LTK generated during an LE Secure Connection bonding procedure using the AES key wrapping process with 128-bit KEK defined in [RFC3394]. The Peripheral IRK is passed as the plaintext input P1 and P2, with P1 corresponding to octet 0 through to octet 7 of the IRK and P2 corresponding to octet 8 through to octet 15 of the IRK. The LTK is passed as the plaintext input P3 and P4, with P3 corresponding to octet 0 through to octet 7 of the LTK and P4 corresponding to octet 8 through to octet 15 of the LTK.

2 The BLE-Keying-Material.KM-Data TLV contains the encrypted 16-octet Peripheral IRK, the 16-octet LTK generated during an LE Secure Connection bonding procedure and the 16-octet Central IRK using the AES key wrapping process with 128-bit KEK defined in [RFC3394]. The Peripheral IRK is passed as the plaintext input P1 and P2, with P1 corresponding to octet 0 through to octet 7 of the IRK and P2 corresponding to octet 8 through to octet 15 of the IRK. The LTK is passed as the plaintext input P3 and P4, with P3 corresponding to octet 0 through to octet 7 of the LTK and P4 corresponding to octet 8 through to octet 15 of the LTK. The Central IRK is passed as plaintext input P5 and P6, with P5 corresponding to octet 0 through to octet 7 of the Central IRK and P6 corresponding to octet 8 through to octet 15 of the Central IRK.

TLV-Length:

6 octets

3.15.6. BLE-Keying-Material.KM-Data

TLV-Type:

5 (BLE-Keying-Material.KM-Data)

TLV-Value:

A string data type encoding a sequence of octets representing the actual encrypted keying material as identified using the BLE-Keying-Material.KM-Type.

TLV-Length:

Variable

3.15.7. BLE-Keying-Material TLV-Type Usage

At least four instances of the BLE-Keying-Material Attribute MUST be included in an Access-Accept packet, that include the following TLV-Types:

* TLV-Type 0 (BLE-Keying-Material.Peripheral-IA)
* TLV-Type 2 (BLE-Keying-Material.IV)
* TLV-Type 4 (BLE-Keying-Material.KM-Type)
* TLV-Type 5 (BLE-Keying-Material.KM-Data)

If a KEK is configured, then in addition the Access-Accept packet MUST include the BLE-Keying-Material Attribute with an instance of TLV-Type 3 (BLE-Keying-Material.KEK-ID). When not present, the NAS MUST use a default value of 0 for the KEK-ID.

If the BLE Peripheral is connectable and the RADIUS Server authorizes connections, then in addition the Access-Accept message MUST include the BLE-Keying-Material Attribute with an instance of TLV-Type 1 (BLE-Keying-Material.Central-IA).

3.16. Forwarding Bluetooth Messages

RADIUS attributes described in this section are used to exchange information to allow non-IP Bluetooth messages to be transferred between the BLE Visited Central Host and a Home MQTT Broker.

3.16.1. MQTT-Broker-URI

Description

The MQTT-Broker-URI (TBA5) Attribute allows a RADIUS server to specify the URI of the MQTT Broker. A single MQTT-Broker-URI Attributes MAY be included in an Access-Accept packet.

If the RADIUS server operates with NAS/BLE Visited Hosts that are deployed behind firewalls or NAT gateways, MQTT Messages SHOULD be transported using WebSocket [RFC6455] as a network transport as defined in MQTT [MQTT] and the the attribute SHOULD specify the URI of a WebSocket server that supports the 'mqtt' Sec-WebSocket-Protocol.

A summary of the MQTT-Broker-URI Attribute format is shown below. The fields are transmitted from left to right.

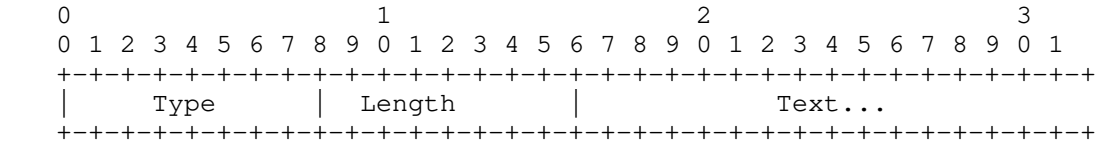


Figure 4: Encoding MQTT-Broker-URI Attribute

Type

TBA5

Length

>=3 octet

Data Type

Text

Value

The text field encodes a URI where the MQTT service can be accessed, e.g., "wss://broker.example.com:443".

3.16.2. MQTT-Token

Description

The MQTT-Token (TBA6) Attribute allows a RADIUS server to signal a token for use by an MQTT client in an MQTT CONNECT packet [MQTT]. The token can be used by an MQTT Broker to associate an MQTT Connection from an MQTT Client with a Network Access Server.

A MQTT-Token Attributes MAY be included in an Access-Accept packet.

A summary of the MQTT-Token Attribute format is shown below. The fields are transmitted from left to right.

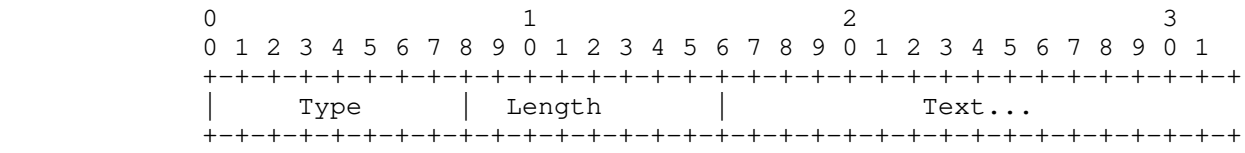


Figure 5: Encoding MQTT-Token Attribute

Type

TBA6

Length

>=3 octet

Data Type

Text

Value

The text field contains a token for use with an MQTT CONNECT packet.

3.17. RADIUS Accounting Attributes

With a few exceptions, the RADIUS accounting attributes defined in [RFC2866] have the same meaning within BLE sessions as they do in dialup sessions and therefore no additional commentary is needed.

3.17.1. Acct-Input-Octets and Acct-Output-Octets

These attributes are not not used by BLE Authenticators.

3.17.2. Acct-Input-Packets

This attribute is used to indicate how many MQTT messages that include the Peripheral Identity Address signalled in the BLE-Keying-Material attribute have been sent by the BLE Central Host.

3.17.3. Acct-Output-Packets

This attribute is used to indicate how many MQTT messages that include the Peripheral Identity Address signalled in the BLE-Keying-Material attribute have been received by the BLE Central Host.

3.17.4. Acct-Terminate-Cause

This attribute indicates how the session was terminated, as described in [RFC2866]. When the idle-timeout attribute is used by the NAS/BLE Visited Host to terminate a RADIUS Accounting session, it MUST set the Acct-Terminate-Cause set to Lost Carrier (2).

4. BLE RADIUS Exchange

The BLE Peripheral uses techniques defined in Bluetooth Core Specifications [BLUETOOTH] to establish a bonded, secure, trusted relationship with a BLE Home Central device in the network. The bonding procedure generates session specific keying material. The BLE Peripheral sends low duty cycle advertising events.

The BLE Peripheral moves into coverage of a second BLE Central device that is integrated with a NAS.

The BLE Peripheral sends Advertisements using its Resolvable Public Address. The contents of the Advertisements are signalled to a BLE Visited Central Host associated with the second BLE Central device. The received Advertisements sent by the BLE Peripheral are used by the BLE Visited Central Host to decide whether to trigger a RADIUS exchange, e.g., using the presence and/or contents of specific Advertising Elements.

The NAS associated with the BLE Visited Central Host is configured with the identity of the RADIUS server. The NAS/BLE Visited Host MAY be statically configured with the identity of a RADIUS Server. Alternatively, the NAS/BLE Visited Host MAY use the contents of an Advertisement Element received from the BLE Peripheral to derive an FQDN of the RADIUS sever and use RFC 7585 [RFC7585] to dynamically resolve the address of the RADIUS server. For example, the peripheral can use the Bluetooth URI data type Advertisement Element (0x24) to encode the Bluetooth defined ’empty scheme’ name tag together with a hostname that identifies the network which operates the BLE Home Central Host associated with the peripheral. Alternatively, a federation of operators of BLE Visited Centrals and RADIUS Servers can define the use of the Bluetooth defined Manufacturer Specific Advertisement Data Element (0xFF) together with a Company Identifier that identifies the federation to signal a federation defined sub-type that encodes information that enables the BLE Visited Central Host to derive an FQDN of the RADIUS sever associated with the advertising peripheral.

The NAS/BLE Host generates a RADIUS Access-Request message using the prand from the RPA as the User-Name attribute and the hash from the RPA to generate the TLV-Type Hashed-Password.Hmac-Sha256-128-Password. The NAS-Port-Type is set to "Wireless - Bluetooth Low Energy".

On receiving the RADIUS Access-Request message, the RADIUS Server uses the keying material exposed by the BLE Home Central Host and attempts to resolve the User-Name and the TLV-Type Hashed-Password.Hmac-Sha256-128-Password to a known BLE Identity Address (IA). If the RADIUS Server cannot resolve the User-Name and TLV-Type Hashed-Password.Hmac-Sha256-128-Password to a known BLE Identity Address, the RADIUS server MUST reject the Access-Request.

If the RADIUS Server resolves the User-Name and TLV-Type Hashed-Password.Hmac-Sha256-128-Password to a known BLE Identity Address, and the BLE Identity Address is authorized to access via the BLE Visited Host, the RADIUS server recovers the session specific keying material exposed by the BLE Home Central Host.

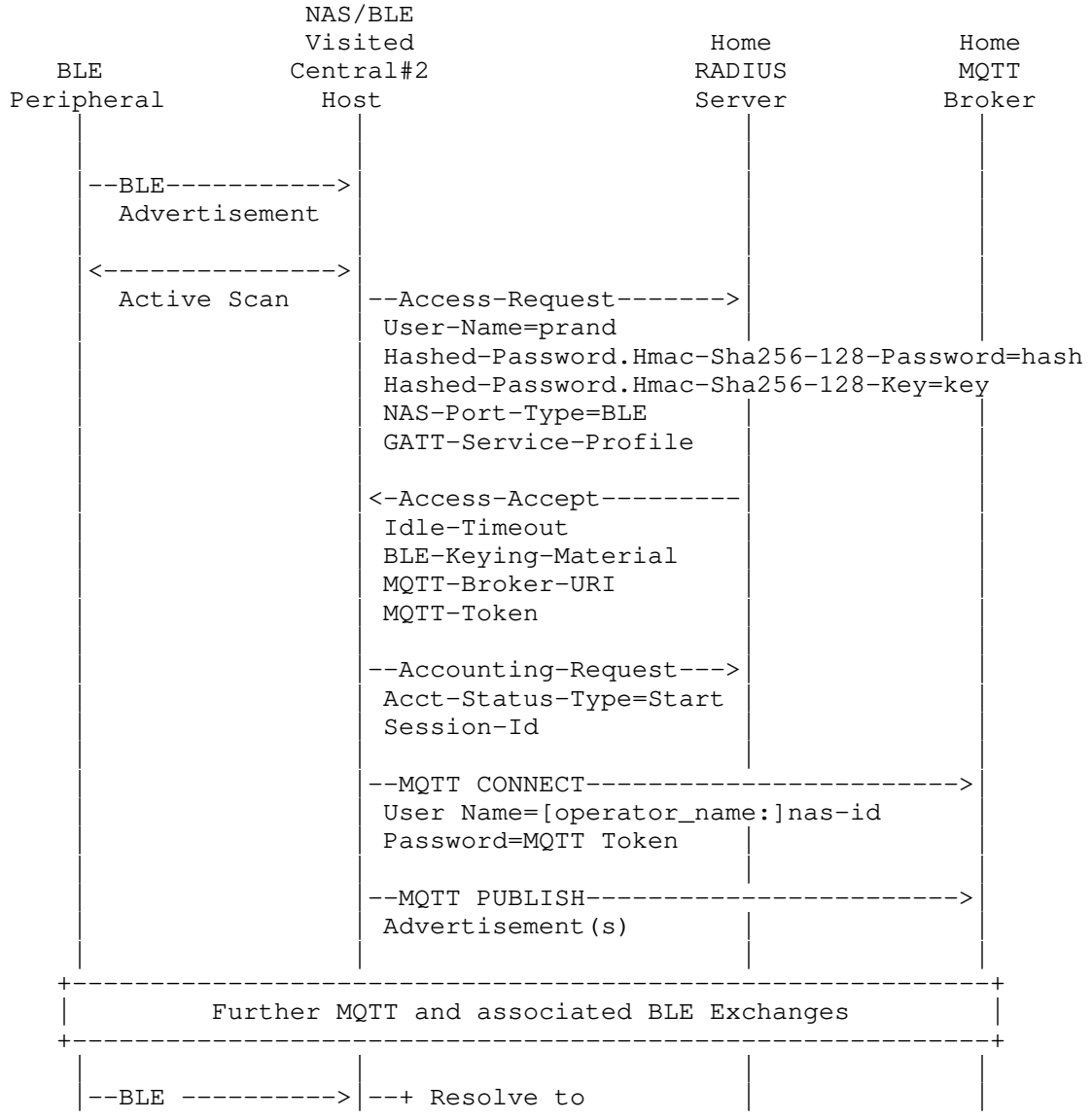
If the BLE Peripheral is not connectable or connections are not authorized, the RADIUS server signals the Peripheral Identity Address in the TLV-type BLE-Keying-Material.Peripheral-IA, sets the value of TLV-Type BLE-Keying-Material.KM-Type to 0 and encodes the Peripheral Identity Resolving Key in the TLV-Type BLE-Keying-Material.KM-Data. If the BLE Peripheral is connectable and connections are authorized via the BLE Visited Host, the RADIUS server additionally includes the Central Identity Address in the TLV-type BLE-Keying-Material.Central-IA, sets the value of TLV-Type BLE-Keying-Material.KM-Type to 1 and encodes the Peripheral Identity Resolving Key and the 16-octet Long Term Key in the TLV-Type BLE-Keying-Material.KM-Data. Finally, if the BLE Peripheral is connectable and connections are authorized via the BLE Visited Host and the security database indicates that the BLE Home Central Host operates using Bluetooth privacy, then the RADIUS server sets the value of TLV-Type BLE-Keying-Material.KM-Type to 2 and encodes the Peripheral Identity Resolving Key, the 16-octet Long Term Key and the 16-octet Central Identity Resolving Key in the TLV-Type BLE-Keying-Material.KM-Data.

The RADIUS Server SHOULD include the MQTT-Broker-URI attribute and MAY include the MQTT-Token attribute by which an MQTT client associated with the BLE Visited Host can establish an MQTT connection with a Home MQTT Broker for forwarding messages received to/from the BLE peripheral.

On receiving the Access-Accept, the NAS/BLE Visited Host recovers the keying material, including the BLE Peripheral’s Identity Address and then establishes an MQTT Connection with the Home MQTT Broker. The

NAS/BLE Visited Host SHOULD include its NAS-Id in the User Name field of the MQTT CONNECT message and MAY include an Operator Name, if for example the NAS has been configured with the operator-name attribute (#126) as specified in section 4.1 of RFC5580 [RFC5580].

If the advertisement that triggered the RADIUS exchange corresponds to an ADV_IND then the NAS/BLE Visited Host can subsequently establish a secure connection with the BLE Peripheral.



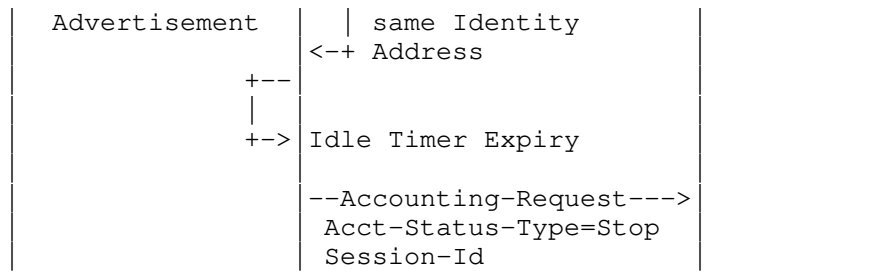


Figure 6: BLE RADIUS Exchange

5. Table of Attributes

The following table provides a guide to which of the attribute defined may be found in which kinds of packets, and in what quantity.

Request	Accept	Reject	Challenge	Acct-Request	#	Attribute
1+	0	0	0	0	TBA2	Hashed-Password
0+	0	0	0	0	TBA3	GATT-Service-Profile
0	1+	0	0	0	TBA4	BLE-Keying-Material
0	0-1	0	0	0	TBA5	MQTT-Broker-URI
0	0-1	0	0	0	TBA6	MQTT-Token

Table 1: Table of Attributes

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

Entry	Meaning
0	This attribute MUST NOT be present in packet.
0+	Zero or more instances of this attribute MAY be present in packet.
0-1	Zero or one instance of this attribute MAY be present in packet.
1	One instance of this attribute MUST be present in packet.

Table 2: Table of Attributes Entry Definition

6. Security Considerations

Use of this RADIUS profile for BLE can be between a NAS/BLE Visited Host and a RADIUS Server inside a secure network, or between a NAS/BLE Visited Host and RADIUS server operated in different administrative domains which are connected over the Internet. All implementations MUST follow [I-D.draft-dekok-radext-deprecating-radius].

The RADIUS profile for BLE devices is designed to operate when BLE devices operate their physical links with BLE Secure Connections [BLUETOOTH]. This approach uses a secure exchange of data over the Bluetooth connection, together with Elliptic Curve Diffie-Hellman (ECDH) public key cryptography, to create the session specific symmetric Long Term Key (LTK) which is then exchanged using the BLE-Keying-Material attribute in the RADIUS Access-Accept message.

Bluetooth [BLUETOOTH] specifies how an IRK can be generated from an Identity Root (IR) key. Removing the Bluetooth bond in a device will typically trigger the generation of a new IRK key for the device.

The RADIUS profile for BLE devices is designed to operate when BLE devices are configured to operate with Bluetooth Privacy Mode enabled [BLUETOOTH]. The BLE device defines the policy of how often it should generate a new Resolvable Private Address. This can be configured to be between every second and every hour, with a default value of every 15 minutes [BLUETOOTH]. This mode mitigates risks associated with a malicious third-party scanning for and collecting Bluetooth addresses over time and using such to build a picture of the movements of BLE devices and, by inference, the human users of those devices.

The Home MQTT broker can observe the Bluetooth messages exchanged with the BLE Peripheral. The Bluetooth GATT attributes SHOULD be cryptographically protected at the application-layer. The Home MQTT Broker MUST be configured with access control lists so that a NAS cannot subscribe to a topic that is intended for another NAS.

The WebSocket connection MUST operate using a WebSocket Secure connection. If the entropy of the MQTT-Token is known to be low, the WebSocket Secure TLS connection SHOULD be secured with certificate-based mutual TLS.

7. IANA Considerations

This document defines a new value of TBA1 for RADIUS Attribute Type #61 (NAS-Port-Type) defined in https://www.iana.org/assignments/radius-types/radius-types.xhtml#radius-types-13

Value	Description	Reference
TBA1	"Wireless - Bluetooth Low Energy"	Section 3.12

Table 3: New NAS-Port-Type value defined in this document

This document defines new RADIUS attributes, (see section Section 3), and assigns values of TBA2, TBA3, TBA4, TBA5 and TBA6 from the RADIUS Attribute Type space https://www.iana.org/assignments/radius-types.

Tag	Attribute	Reference
TBA2	Hashed-Password	Section 3.13
TBA3	GATT-Service-Profile	Section 3.14
TBA4	BLE-Keying-Material	Section 3.15
TBA5	MQTT-Broker-URI	Section 3.16.1
TBA6	MQTT-Token	Section 3.16.2

Table 4: New RADIUS attributes defined in this document

8. References

8.1. Normative References

[I-D.draft-dekok-radext-deprecating-radius]
DeKok, A., "Deprecating RADIUS/UDP and RADIUS/TCP", Work in Progress, Internet-Draft, draft-dekok-radext-deprecating-radius-01, 3 March 2023, <https://datatracker.ietf.org/doc/html/draft-dekok-radext-deprecating-radius-01>.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, DOI 10.17487/RFC4868, May 2007, <https://www.rfc-editor.org/info/rfc4868>.

[RFC5580] Tschofenig, H., Ed., Adrangi, F., Jones, M., Lior, A., and B. Aboba, "Carrying Location Objects in RADIUS and Diameter", RFC 5580, DOI 10.17487/RFC5580, August 2009, <https://www.rfc-editor.org/info/rfc5580>.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[BLUETOOTH]
Bluetooth Core Specification Working Group, "BLUETOOTH CORE SPECIFICATION v5.3", 13 July 2021, <https://www.bluetooth.com/specifications/bluetooth-core-specification/>.

[I-D.shahzad-scim-device-model]
Shahzad, M., Hassan, H., and E. Lear, "Device Schema Extensions to the SCIM model", Work in Progress, Internet-Draft, draft-shahzad-scim-device-model-05, 2 June 2023, <https://datatracker.ietf.org/doc/html/draft-shahzad-scim-device-model-05>.

[MQTT] OASIS, "MQTT Version 5.0", 7 March 2019, <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC3394] Schaad, J. and R. Housley, "Advanced Encryption Standard (AES) Key Wrap Algorithm", RFC 3394, DOI 10.17487/RFC3394, September 2002, <https://www.rfc-editor.org/info/rfc3394>.

[RFC3580] Congdon, P., Aboba, B., Smith, A., Zorn, G., and J. Roese, "IEEE 802.1X Remote Authentication Dial In User Service (RADIUS) Usage Guidelines", RFC 3580, DOI 10.17487/RFC3580, September 2003, <https://www.rfc-editor.org/info/rfc3580>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

Appendix A. MQTT Interworking

This section describes how a NAS/BLE Visited Host supporting the BLE RADIUS profile can interwork with a Home MQTT Message Broker in order to use MQTT topics to deliver Bluetooth messages to/from a BLE Peripheral. It is intended to move this material to another document - but is included here to describe, at a high level, the MQTT interworking established by the RADIUS exchange.

A.1. Establishing a Session to a MQTT-Broker-URI

If the NAS/BLE Visited Host is signalled a MQTT-Broker-URI in an Access-Accept with which it does not have an established MQTT connection, then it MUST establish an MQTT connection. It the NAS/BLE Visited Host is behind a firewall or NAT gateway it MUST use WebSocket transport for the MQTT connection. The user name in the MQTT CONNECT message SHOULD include the NAS-ID and MAY include the name of the operator of the NAS/BLE Visited Host.

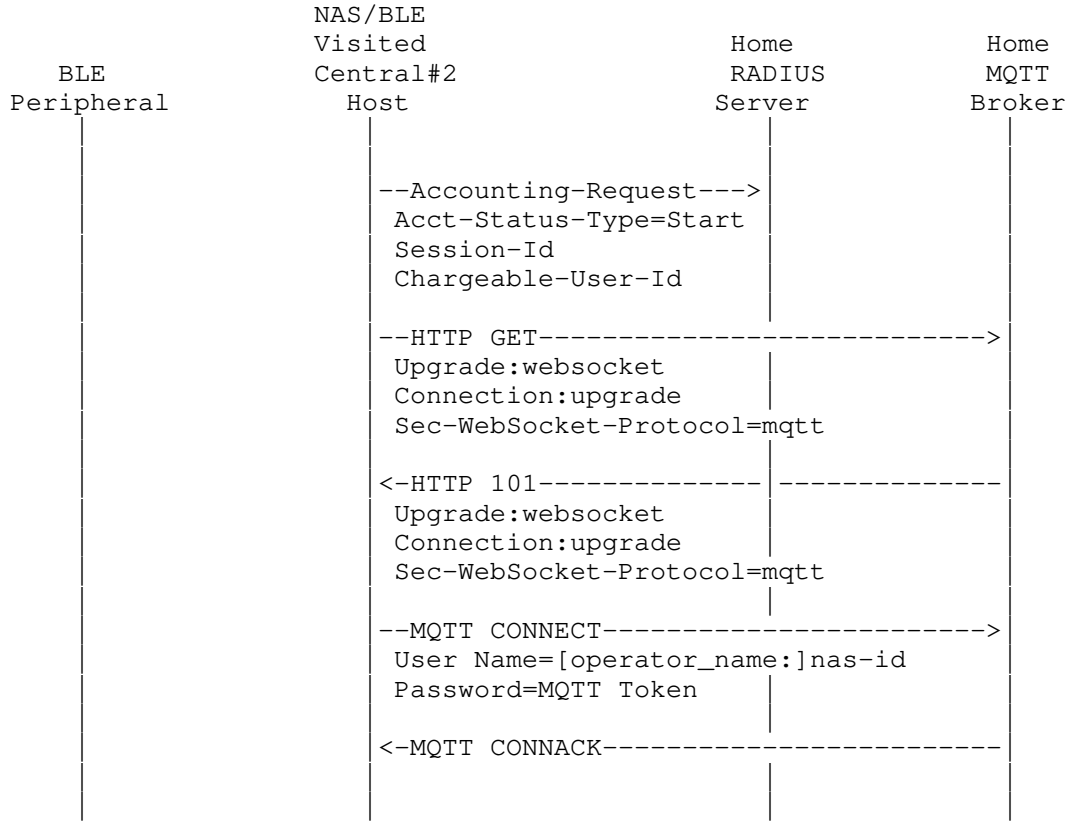


Figure 7: Establishing an MQTT connection to a Home Broker using WebSocket transport

A.2. MQTT topics

The following topic is used by the MQTT client of the BLE Visited Host to signal active and passive scan advertisements received from BLE Peripherals to the home MQTT Broker.

* {peripheral_identity_address}/advertisement/gatt-ind

If the BLE Peripheral is connectable, the MQTT client of the BLE Visited Host SHOULD subscribe to the following message topics to be able to receive GATT requests from the Home MQTT Broker:

1. {peripheral_identity_address}/connect/gatt-req : when publishing a message on the {peripheral_identity_address}/connect/gatt-req topic, an MQTT client SHOULD include the following as a response topic {peripheral_identity_address}/connect/gatt-res.

2. {peripheral_identity_address}/disconnect/gatt-req : when publishing a message on the {peripheral_identity_address}/disconnect/gatt-req topic, an MQTT client SHOULD include the following as a response topic {peripheral_identity_address}/disconnect/gatt-res.

3. {peripheral_identity_address}/read/gatt-req : when publishing a message on the {peripheral_identity_address}/read/gatt-req topic, an MQTT client SHOULD include the following as a response topic {peripheral_identity_address}/read/gatt-res.

4. {peripheral_identity_address}/write/gatt-req : when publishing a message on the {peripheral_identity_address}/write/gatt-req topic, an MQTT client SHOULD include the following as a response topic {peripheral_identity_address}/write/gatt-res.

5. {peripheral_identity_address}/service-discovery/gatt-req : when publishing a message on the {peripheral_identity_address}/service-discovery/gatt-req topic, an MQTT client SHOULD include the following as a response topic {peripheral_identity_address}/service-discovery/gatt-res.

6. {peripheral_identity_address}/notification/gatt-ind-res : when sending indications, the MQTT client of the NAS/BLE Visited Host SHOULD publish the message using the topic:{peripheral_identity_address}/notification/gatt-ind-req indication and SHOULD include the following as a response topic {peripheral_identity_address}/notification/gatt-ind-res.

A.3. MQTT Exchange for Non-Connectable BLE Peripherals

If the BLE Peripheral indicates in its scan that it is not connectable, the NAS/BLE Visited Host is responsible for publishing the received advertisements received from the authenticated BLE Peripheral.

On idle-timeout the NAS/BLE Visited Host MUST send an Accounting-Request message with Acct-Status-Type set to STOP and Acct-Terminate-Cause set to Lost Carrier (2).

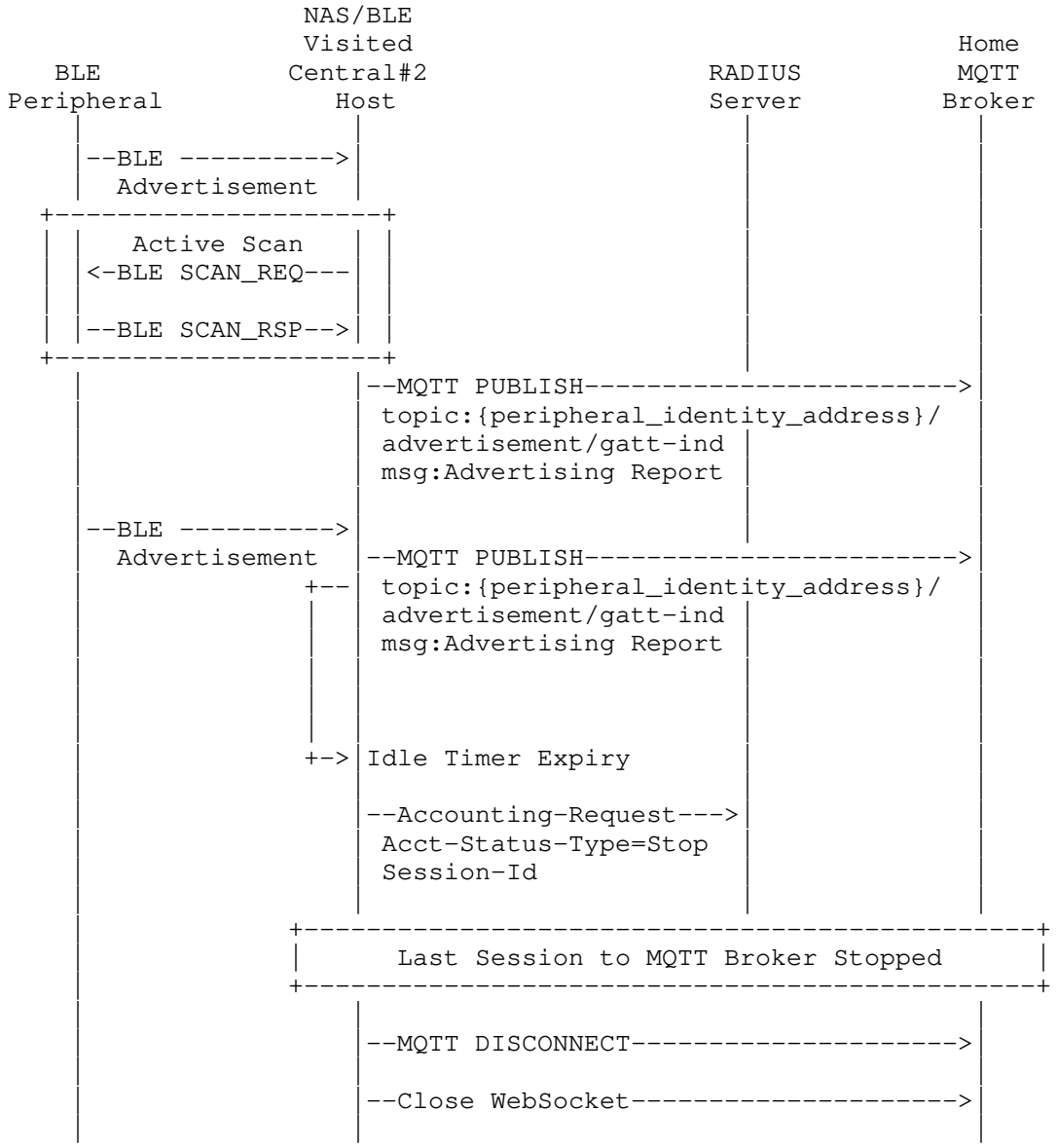
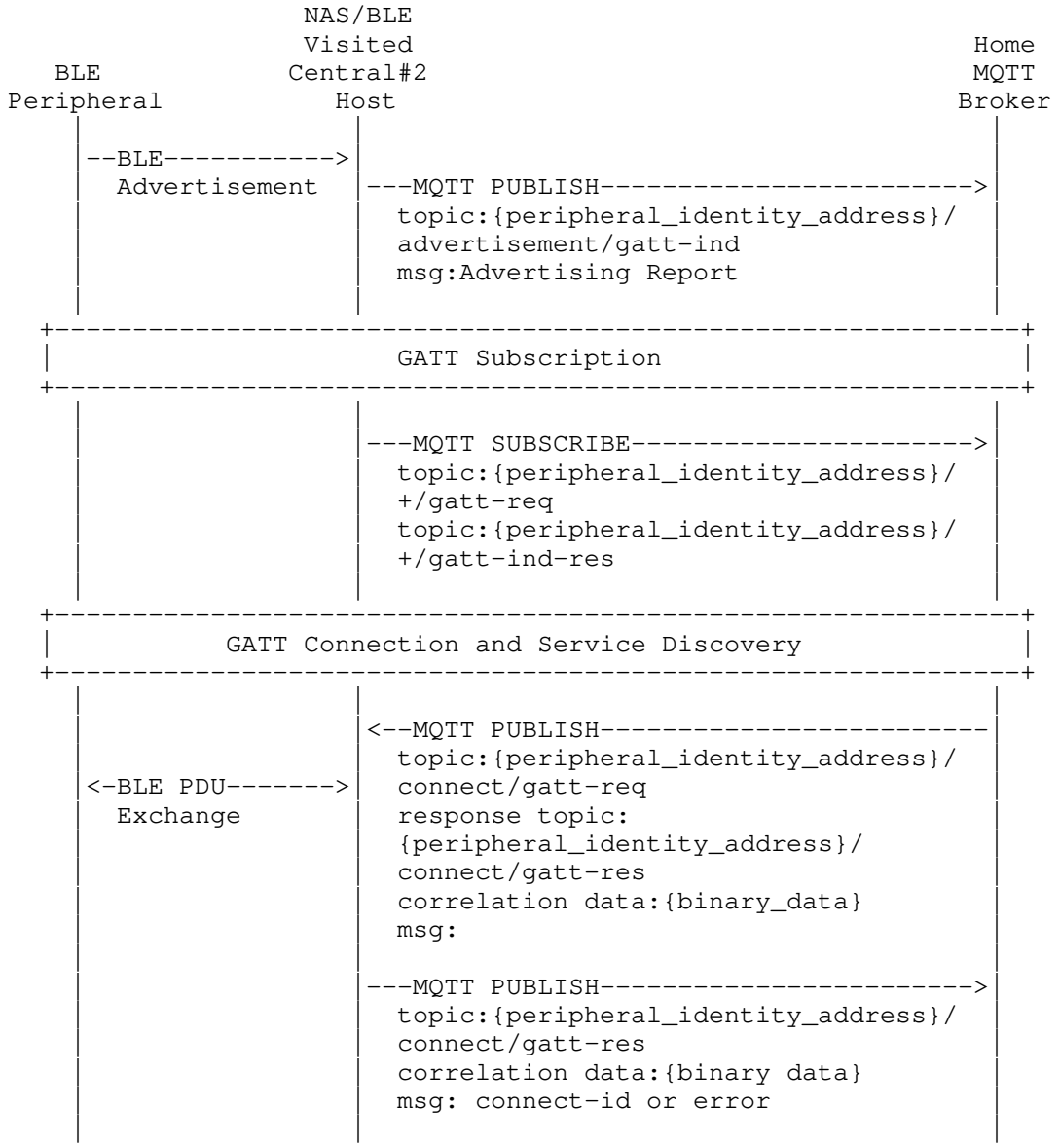


Figure 8: MQTT Exchange for Non-Connectable BLE Peripherals

A.4. Initial MQTT Exchange for Connectable BLE Peripherals

If the BLE Peripheral indicates in its scan that it is connectable, the NAS/BLE Visited Host is responsible for publishing the received advertisements received from the authenticated BLE Peripheral and to subscribing to the GATT requests published for the BLE Peripheral's Identity Address.



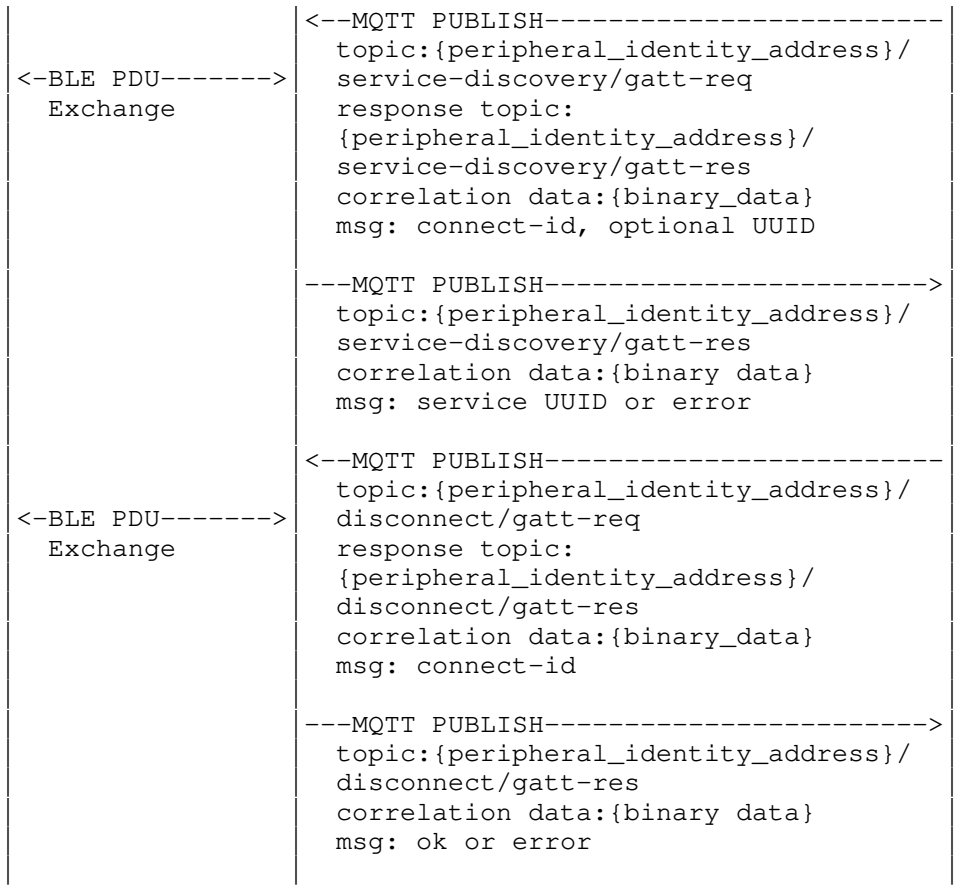


Figure 9: MQTT Exchange for GATT Service Discovery

A.5. MQTT Exchange for Reading a GATT Attribute

If the BLE Peripheral is connectable, a Bluetooth Application can read GATT attributes.

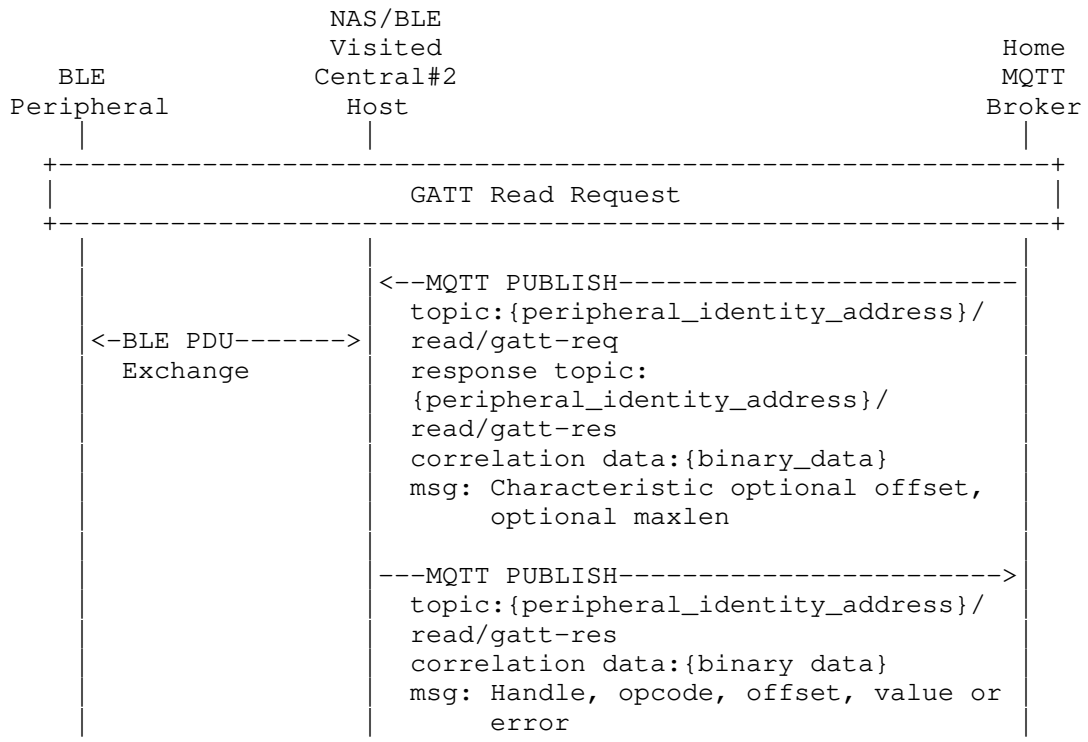


Figure 10: MQTT Exchange for GATT Read Attribute

A.6. MQTT Exchange for Writing a GATT Attribute

If the BLE Peripheral is connectable, a Bluetooth Application can write GATT attributes.

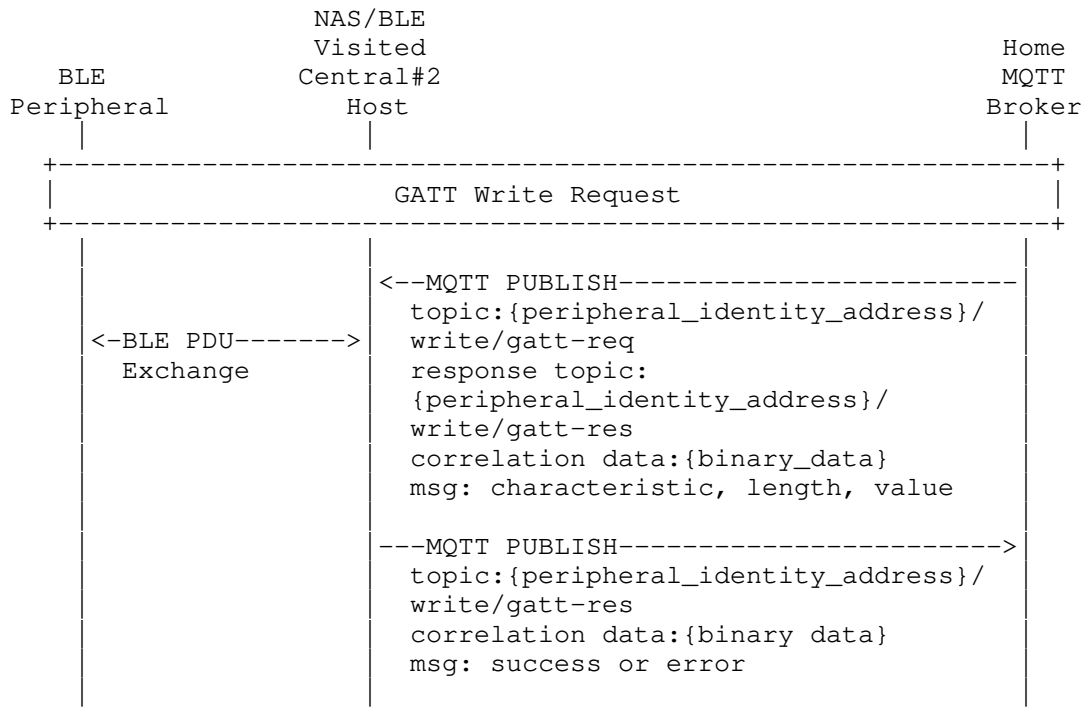


Figure 11: MQTT Exchange for GATT Write Attribute

A.7. MQTT Exchange for BLE Peripheral initiated Notifications

A Bluetooth Application can subscribe to receive Bluetooth notifications sent by the BLE Peripheral.

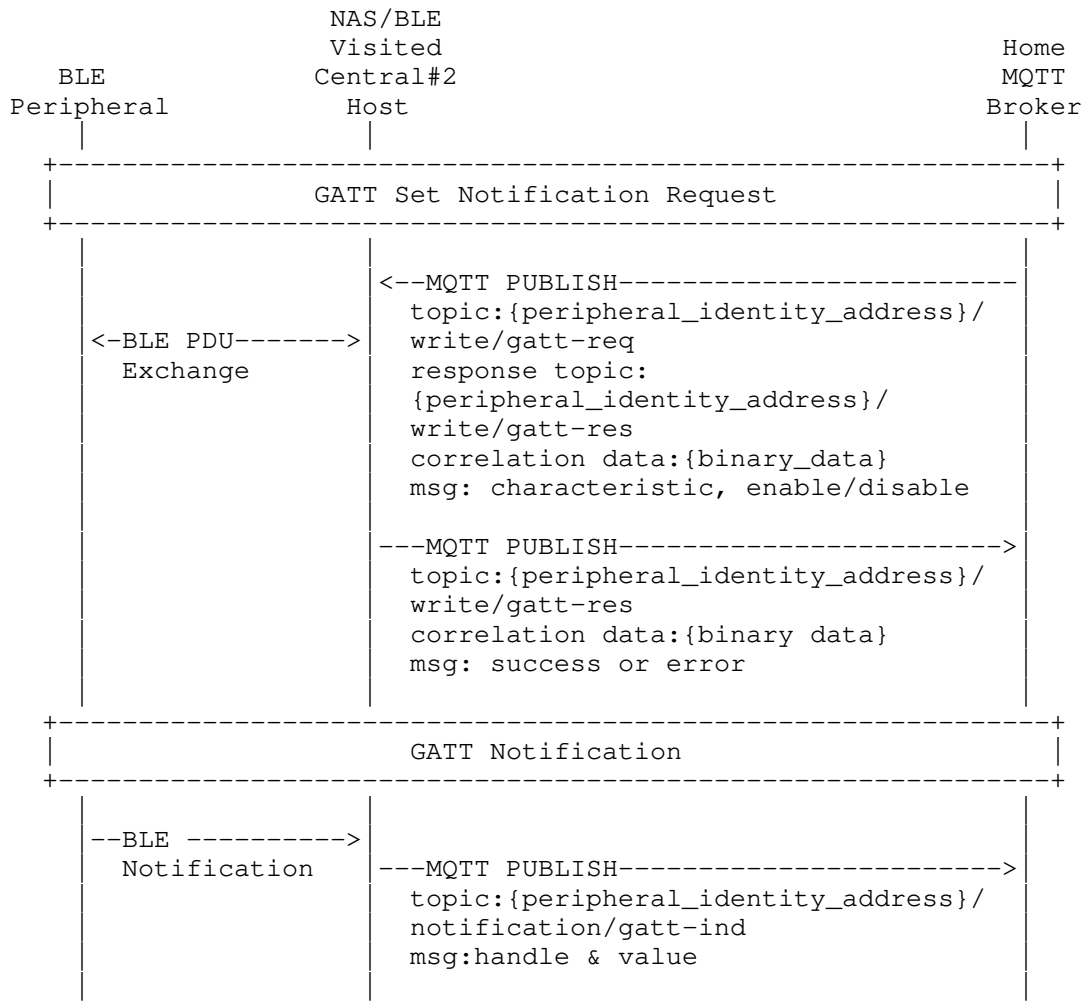


Figure 12: MQTT Exchange for BLE Peripheral Notifications

A.8. MQTT Exchange for BLE Peripheral initiated Indications

A Bluetooth Application can subscribe to receive Bluetooth indications sent by the BLE Peripheral.

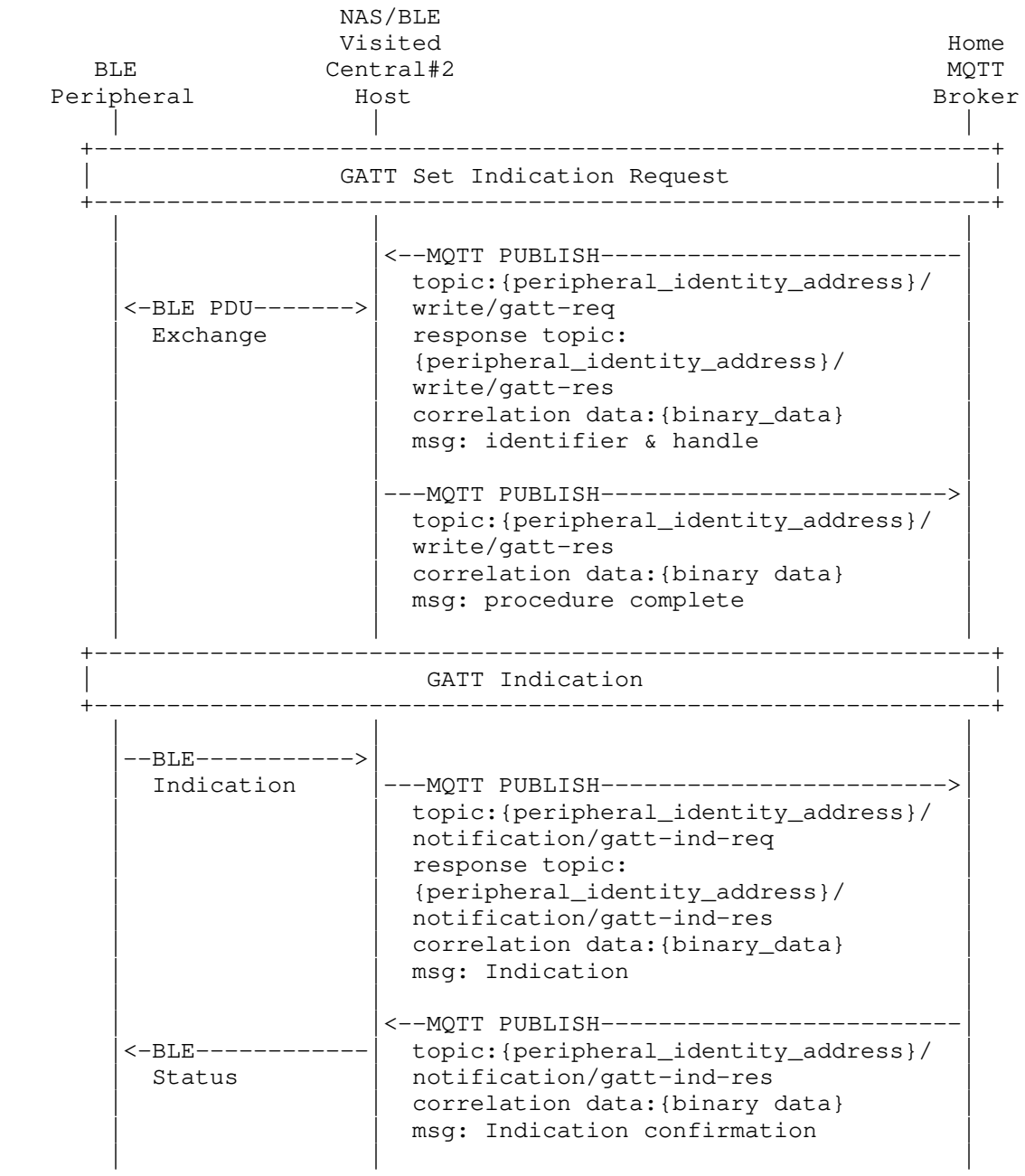
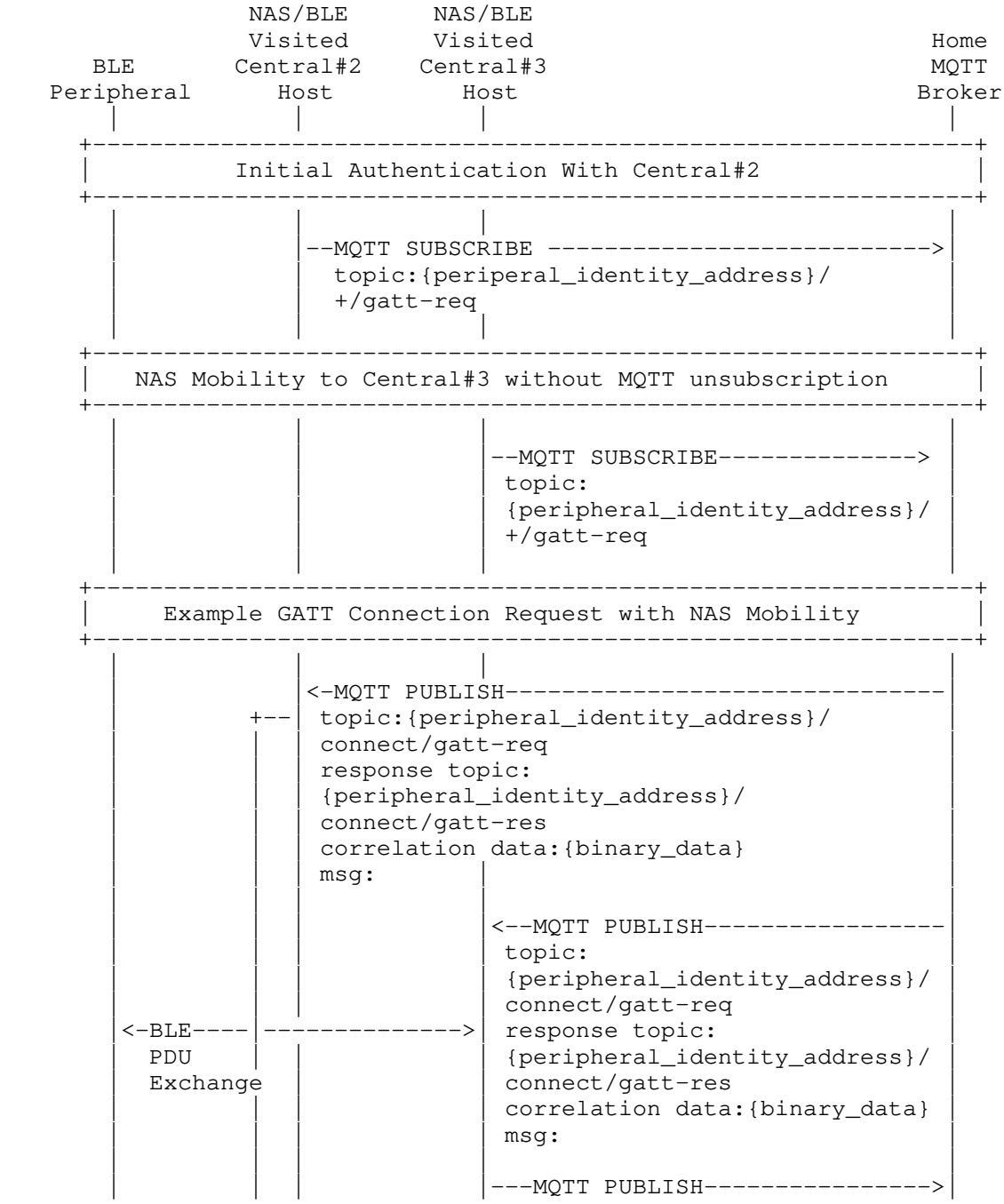


Figure 13: MQTT Exchange for BLE Peripheral Indications

A.9. MQTT Exchange for dealing with NAS Mobility



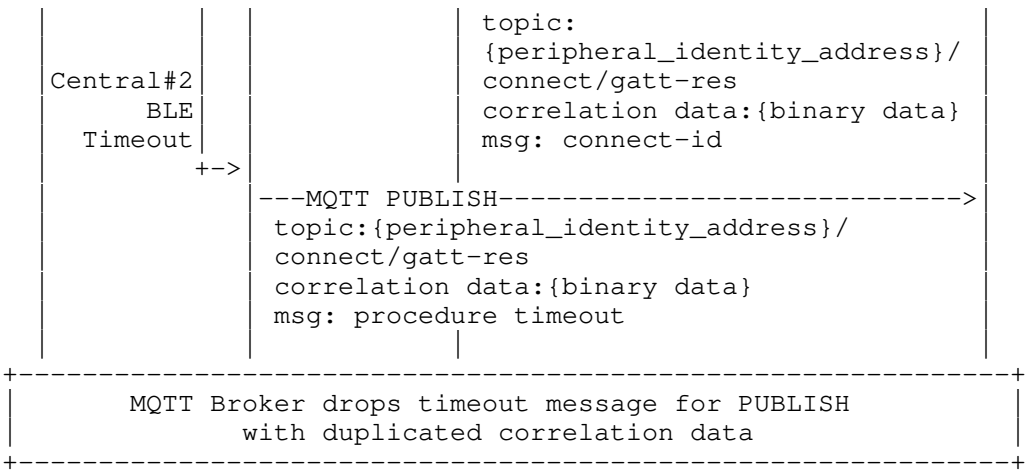


Figure 14: MQTT Exchange for Inter-NAS Mobility without MQTT Unsubscription

A.10. MQTT Exchange for ending a session for a connected BLE Peripheral

On idle-timeout the NAS/BLE Visited Host MUST un-subscribe from any subscribed to topics and send an Accounting-Request message with Acct-Status-Type set to STOP and Acct-Terminate-Cause set to Lost Carrier (2).

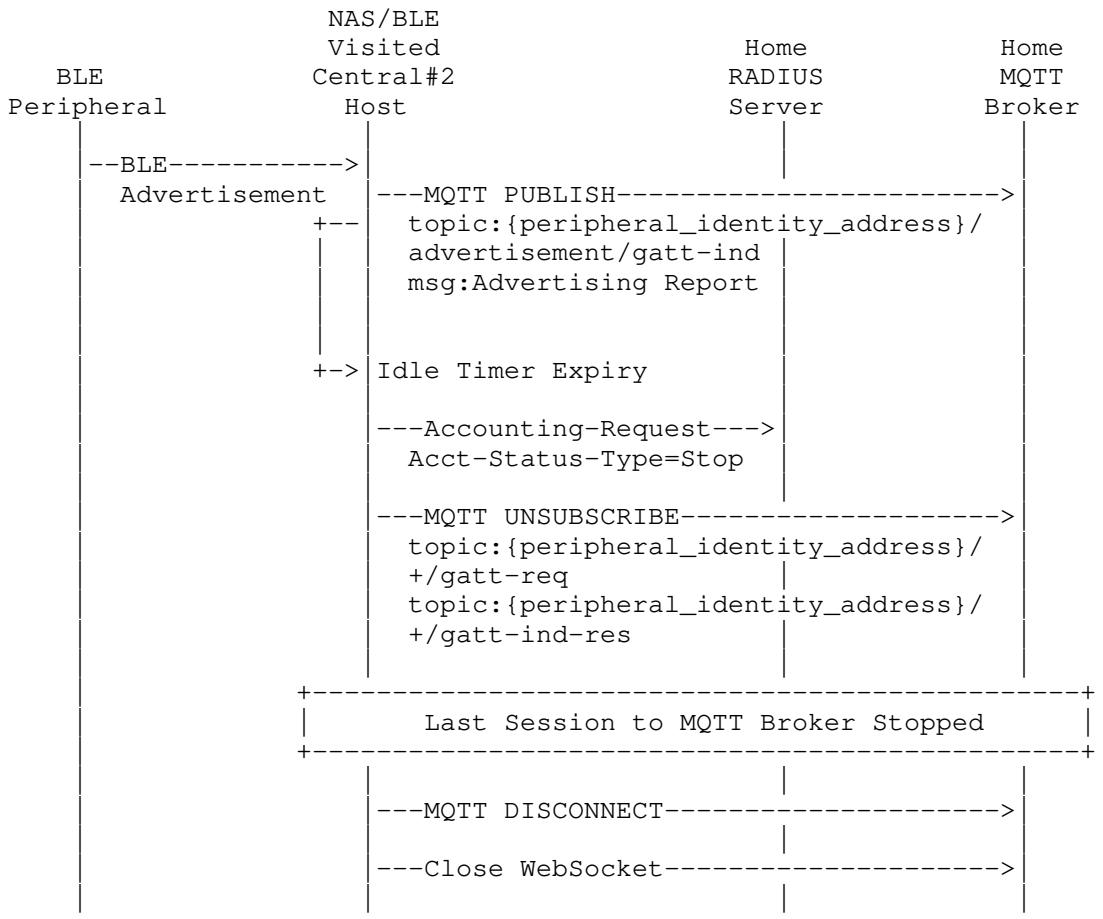


Figure 15: MQTT Exchange when disconnecting from a connected BLE Peripheral

Appendix B. History of Changes

Note: This appendix will be deleted in the final version of the document.

From version 00 -> 01:

* switched from User-Password to new Hashed-Password attribute using SHA256

* switched to TLV-encoding of BLE-Keying-Material

* re-ordered MQTT topic definitions

* removed redundant attribute sections

Acknowledgements

Thanks to Oleg Pekar and Eric Vyncke for their review comments.

Authors' Addresses

Mark Grayson
Cisco Systems
10 New Square Park
Feltham
TW14 8HA
United Kingdom
Email: mgrayson@cisco.com

Eliot Lear
Cisco Systems
Glatt-com
CH- CH-8301 Glattzentrum, Zurich
Switzerland
Email: elear@cisco.com

RADIUS EXTensions
Internet-Draft
Obsoletes: 6614 (if approved)
Intended status: Standards Track
Expires: 11 September 2023

J.-F. Rieckers
DFN
S. Winter
RESTENA

10 March 2023

Transport Layer Security (TLS) Encryption for RADIUS
draft-rieckers-radext-rfc6614bis-02

Abstract

This document specifies a transport profile for RADIUS using Transport Layer Security (TLS) over TCP as the transport protocol. This enables dynamic trust relationships between RADIUS servers as well as encrypting RADIUS traffic between servers using a shared secret.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rieckers-radext-rfc6614bis/.

Discussion of this document takes place on the RADIUS EXTensions Working Group mailing list (mailto:radext@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/radext/.

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 11 September 2023.

Table of Contents

1. Introduction

The RADIUS protocol [RFC2865] is a widely deployed authentication and authorization protocol. The supplementary RADIUS Accounting specification [RFC2866] provides accounting mechanisms, thus delivering a full Authentication, Authorization, and Accounting (AAA) solution. However, RADIUS has shown several shortcomings, especially the lack of security for large parts of its packet payload. RADIUS security is based on the MD5 algorithm, which has been proven to be insecure.

The main focus of RADIUS over TLS is to provide a means to secure the communication between RADIUS/TCP peers using TLS. The most important use of this specification lies in roaming environments where RADIUS packets need to be transferred through different administrative domains and untrusted, potentially hostile network.

There are multiple known attacks on the MD5 algorithm that is used in RADIUS to provide integrity protection and a limited confidentiality protection. RADIUS over TLS wraps the entire RADIUS packet payload into a TLS stream and thus mitigates the risk of attacks on MD5.

Because of the static trust establishment between RADIUS peers (IP address and shared secret), the only scalable way of creating a massive deployment of RADIUS servers under the control of different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times, as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RADIUS over TLS add a new way to identify other peers, e.g., by checking a certificate for the issuer or other certificate properties, but also provides a simple upgrade path for existing RADIUS connection by simply using the shared secret to authenticate the TLS session.

1.1. Conventions and Definitions

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.

Within this document we will use the following terms:

RADIUS/TLS node: a RADIUS-over-TLS client or server

RADIUS/TLS Client: a RADIUS-over-TLS instance that initiates a new connection

RADIUS/TLS Server: a RADIUS-over-TLS instance that listens on a RADIUS-over-TLS port and accepts new connections

RADIUS/UDP: a classic RADIUS transport over UDP as defined in [RFC2865]

1.2. Changes from RFC6614

Currently, there are no big changes, since this is just a restructured spec from [RFC6614].

The following things have changed:

Required TLS versions: TLS 1.2 is now the minimum TLS version, TLS 1.3 is included as recommended.

TLS compression: [RFC6614] allowed usage of TLS compression, this document forbids it.

TLS-PSK support: [RFC6614] lists support for TLS-PSK as OPTIONAL, this document changes this to RECOMMENDED.

Mandatory-to-implement(MTI) cipher suites: Following the recommendation from [RFC9325], the RC4 cipher suite is no longer included as SHOULD, and the AES cipher suite is the new MTI cipher suite, since it is the MTI cipher suite from TLS 1.2. Additionally, this document references [RFC9325] for further recommendations for cipher suites.

The following things will change in future versions of this draft:

* Usage of Server Name Indication

* More text for TLS-PSK

2. Transport layer security for RADIUS/TCP

This section specifies the way TLS is used to secure the traffic and the changes in the handling of RADIUS packets.

2.1. TCP port and Packet Types

The default destination port number for RADIUS over TLS is TCP/2083. There are no separate ports for authentication, accounting, and dynamic authorization changes. The source port is arbitrary.

2.2. TLS Connection setup

The RADIUS/TLS nodes first try to establish a TCP connection as per [RFC6613]. Failure to connect leads to continuous retries. It is RECOMMENDED to use exponentially growing intervals between every try.

After completing the TCP handshake, the RADIUS/TLS nodes immediately negotiate a TLS session. The following restrictions apply:

* Support for TLS 1.2 [RFC5246] is REQUIRED, support for TLS 1.3 [RFC8446] is RECOMMENDED. RADIUS/TLS nodes MUST NOT negotiate TLS versions prior to TLS 1.2.

* The RADIUS/TLS nodes MUST NOT offer or negotiate cipher suites which do not provide confidentiality and integrity protection.

* The RADIUS/TLS nodes MUST NOT negotiate compression.

* When using TLS 1.3, RADIUS/TLS nodes MUST NOT use early data ([RFC8446], Section 2.3)

* RADIUS/TLS implementations MUST, at minimum, support negotiation of the TLS_RSA_WITH_AES_128_CBC_SHA cipher suite and SHOULD follow the recommendations for supported cipher suites in [RFC9325], Section 4.

* In addition, RADIUS/TLS implementations MUST support negotiation of the mandatory-to-implement cipher suites required by the versions of TLS they support.

Details for peer authentication are described in Section 2.3.

After successful negotiation of a TLS session, the RADIUS/TLS peers can start exchanging RADIUS datagrams. The shared secret to compute the (obsolete) MD5 integrity checks and attribute obfuscation MUST be "radsec".

2.3. TLS Peer Authentication

Peers MUST mutually authenticate each other at the TLS layer. The authentication of peers can be done using different models, that will be described here. Peers can also perform additional authorization checks based on non-TLS information. For example, verifying that the client IP address (source IP address of the TLS connection) falls within a particular network range.

2.3.1. Authentication using X.509 certificates with PKIX trust model

All RADIUS/TLS implementations MUST implement this model, following the following rules:

* Implementations MUST allow the configuration of a list of trusted Certificate Authorities for incoming connections.

* Certificate validation MUST include the verification rules as per [RFC5280].

* Implementations SHOULD indicate their trusted Certification Authorities (CAs). See [RFC5246], Section 7.4.4 and [RFC6066], Section 6 for TLS 1.2 and [RFC8446], Section 4.2.4 for TLS 1.3.

* RADIUS/TLS clients validate the server identity to match their local configuration:

  - If the expected RADIUS/TLS server was configured as a hostname, the configured name is matched against the presented names from the subjectAltName:DNS extension; if no such exist, against the presented CN component of the certificate subject.

  - If the expected RADIUS/TLS server was configured as an IP address, the configured IP address is matched against the presented addresses in the subjectAltName:iPAddr extension; if no such exist, against the presented CN component of the certificate subject.

  - If the expected RADIUS/TLS server was not configured but discovered as per [RFC7585], the peer executes the following checks in this order, accepting the certificate on the first match:

    o The realm which was used as input to the discovery is matched against the presented realm names from the subjectAltName:naiRealm extension.

  - If the discovery process yielded a hostname, this hostname is matched against the presented names from the subjectAltName:DNS extension; if no such exist, against the presented CN component of the certificate subject. Implementations MAY require the use of DNSSEC [RFC4033] to ensure the authenticity of the DNS result before relying on this for trust checks.

  - If the previous checks fail, the certificate MAY be accepted without further name checks immediately after the [RFC5280] trust chain checks.

* RADIUS/TLS server validate the incoming certificate against a local database of acceptable clients. The database may enumerate acceptable clients either by IP address or by a name component in the certificate.

  - For clients configured by name, the configured name is matched against the presented names from the subjectAltName:DNS extension; if no such exists, against the presented CN component in the certificate subject.

  - For clients configured by their source IP address, the configured IP address is matched against the presented addresses in the subjectAltName:iPAddr extension; if no such exist, against the presented CN component of the certificate subject.

  - It is possible for a RADIUS/TLS server to not require additional name checks for incoming RADIUS/TLS clients. In this case, the certificate is accepted immediately after the [RFC5280] trust chain checks. This MUST NOT be used outside of trusted network environments or without additional certificate attribute checks in place.

* Implementations MAY allow the configuration of a set of additional properties of the certificate to check for a peer's authorization to communicate (e.g., a set of allowed values in subjectAltName:URI or a set of allowed X.509v3 Certificate Policies).

* When the configured trust base changes (e.g., removal of a CA from the list of trusted CAs; issuance of a new CRL for a given CA), implementations MAY renegotiate the TLS session to reassess the connecting peer's continued authorization.
  // Replace may with should here?
  //
  // -- Janfred

2.3.2. Authentication using certificate fingerprints

RADIUS/TLS implementations SHOULD allow the configuration of a list of trusted certificates, identified via fingerprint of the DER encoded certificate octets. When implementing this model, support for SHA-1 as hash algorithm for the fingerprint is REQUIRED, and support for the more contemporary has function SHA-256 is RECOMMENDED.

2.3.3. Authentication using TLS-PSK

RADIUS/TLS implementations SHOULD support the use of TLS-PSK.

2.3.4. Authentication using Raw Public Keys

RADIUS/TLS implementations SHOULD support using Raw Public Keys [RFC7250] for mutual authentication.
// TODO: More text here.
//
// -- Janfred

2.4. Connecting Client Identity

In RADIUS/UDP, clients are uniquely identified by their IP address. Since the shared secret is associated with the origin IP address, if more than one RADIUS client is associated with the same IP address, then those clients also must utilize the same shared secret. This practice is inherently insecure, as noted in [RFC5247], Section 5.3.2.

Following the different authentication modes presented in Section 2.3, the identification of clients can be done by different means:

In TLS-PSK operation, a client is uniquely identified by its PSK Identity.

When using certificate fingerprints, a client is uniquely identified by the fingerprint of the presented client certificate.

When using X.509 certificates with a PKIX trust model, a client is uniquely identified by the tuple of the serial number of the presended client certificate and the issuer of the client certificate.

// TODO: Client identity when using Raw Public Key needs to be
// described here.
//
// -- Janfred

Note well: having identified a connecting entity does not mean the server necessarily wants to communicate with that client. For example, if the issuer is not in a trusted set of issuers, the server may decline to perform RADIUS transactions with this client.

There are numerous trust models in PKIX environments, and it is beyond the scope of this document to define how a particular deployment determines whether a client is trustworthy. Implementations that want to support a wide variety of trust models should expose as many details of the presented certificate to the administrator as possible so that the trust model can be implemented by the administrator. As a suggestion, at least the following parameters of the X.509 client certificate should be exposed:

* Originating IP address

* Certificate Fingerprint

* Issuer

* Subject

* all X.509v3 Extended Key Usage

* all X.509v3 Subject Alternative Name

* all X.509v3 Certificate Policies

For TLS-PSK operation, at least the following parameters of the TLS connection should be exposed:

* Originating IP address

* PSK Identity

2.5. RADIUS Datagrams

Authentication, Authorization, and Accounting packets are sent according to the following rules:

RADIUS/TLS clients transmit the same packet types on the connection they initiated as a RADIUS/UDP client would. For example, they send

* Access-Request
* Accounting-Request
* Status-Server
* Disconnect-ACK
* Disconnect-NAK
* ...

RADIUS/TLS servers transmit the same packets on connections they have accepted as a RADIUS/UDP server would. For example, they send

* Access-Challenge
* Access-Accept
* Access-Reject
* Accounting-Response
* Disconnect-Request
* ...

Due to the use of one single TCP port for all packet types, it is required that a RADIUS/TLS server signal which types of packets are supported on a server to a connecting peer.

* When an unwanted packet of type 'CoA-Request' or 'Disconnect-Request' is received, a RADIUS/TLS server needs to respond with a 'CoA-NAK' or 'Disconnect-NAK', respectively. The NAK SHOULD contain an attribute Error-Cause with the value 406 ("Unsupported Extension"); see [RFC5176] for details.

* When an unwanted packet of type 'Accounting-Request' is received, the RADIUS/TLS server SHOULD reply with an Accounting-Response containing an Error-Cause attribute with value 406 "Unsupported Extension" as defined in [RFC5176]. A RADIUS/TLS accounting client receiving such an Accounting-Response SHOULD log the error and stop sending Accounting-Request packets to this server.

3. Design Decisions

This section explains the design decisions that led to the rules defined in the previous section, as well as a reasoning behind the differences to [RFC6614].

3.1. Implications of Dynamic Peer Discovery

One mechanism to discover RADIUS-over-TLS peers dynamically via DNS is specified in [RFC7585]. While this mechanism is still under development and therefore is not a normative dependency of RADIUS/TLS, the use of dynamic discovery has potential future implications that are important to understand.

Readers of this document who are considering the deployment of DNS-based dynamic discovery are thus encouraged to read [RFC7585] and follow its future development.

3.2. X.509 Certificate Considerations

(1) If a RADIUS/TLS client is in possession of multiple certificates from different CAs (i.e., is part of multiple roaming consortia) and dynamic discovery is used, the discovery mechanism possibly does not yield sufficient information to identify the consortium uniquely (e.g., DNS discovery). Subsequently, the client may not know by itself which client certificate to use for the TLS handshake. Then, it is necessary for the server to signal to which consortium it belongs and which certificates it expects. If there is no risk of confusing multiple roaming consortia, providing this information in the handshake is not crucial.

(2) If a RADIUS/TLS server is in possession of multiple certificates from different CAs (i.e., is part of multiple roaming consortia), it will need to select one of its certificates to present to the RADIUS/TLS client. If the client sends the Trusted CA Indication, this hint can make the server select the appropriate certificate and prevent a handshake failure. Omitting this indication makes it impossible to deterministically select the right certificate in this case. If there is no risk of confusing multiple roaming consortia, providing this indication in the handshake is not crucial.

3.3. Cipher Suites and Compression Negotiation Considerations

See [RFC9325] for considerations regarding the cipher suites and negotiation.

3.4. RADIUS Datagram Considerations

(1) After the TLS session is established, RADIUS packet payloads are exchanged over the encrypted TLS tunnel. In RADIUS/UDP, the packet size can be determined by evaluating the size of the datagram that arrived. Due to the stream nature of TCP and TLS, this does not hold true for RADIUS/TLS packet exchange. Instead, packet boundaries of RADIUS packets that arrive in the stream are calculated by evaluating the packet's Length field. Special care needs to be taken on the packet sender side that the value of the Length field is indeed correct before sending it over the TLS tunnel, because incorrect packet lengths can no longer be detected by a differing datagram boundary. See Section 2.6.4 of [RFC6613] for more details.

(2) Within RADIUS/UDP [RFC2865], a shared secret is used for hiding attributes such as User-Password, as well as in computation of the Response Authenticator. In RADIUS accounting [RFC2866], the shared secret is used in computation of both the Request Authenticator and the Response Authenticator. Since TLS provides integrity protection and encryption sufficient to substitute for RADIUS application-layer security, it is not necessary to configure a RADIUS shared secret. The use of a fixed string for the obsolete shared secret eliminates possible node misconfigurations.

(3) RADIUS/UDP [RFC2865] uses different UDP ports for authentication, accounting, and dynamic authorization changes. RADIUS/TLS allocates a single port for all RADIUS packet types. Nevertheless, in RADIUS/TLS, the notion of a client that sends authentication requests and processes replies associated with its users' sessions and the notion of a server that receives requests, processes them, and sends the appropriate replies is to be preserved. The normative rules about acceptable packet types for clients and servers mirror the packet flow behavior from RADIUS/UDP.

(4) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support the reception and processing of the packet types in [RFC5176]. These packet types are listed as to be received in RADIUS/TLS implementations. Note well: it is not required for an implementation to actually process these packet types; it is only required that the NAK be sent as defined above.

(5) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly

allocated UDP port to signal that a peer RADIUS server does not support the reception and processing of RADIUS Accounting packets. There is no RADIUS datagram to signal an Accounting NAK. Clients may be misconfigured for sending Accounting packets to a RADIUS/TLS server that does not wish to process their Accounting packet. To prevent a regression of detectability of this situation, the Accounting-Response + Error-Cause signaling was introduced.

4. Compatibility with Other RADIUS Transports

The IETF defines multiple alternative transports to the classic UDP transport model as defined in [RFC2865], namely RADIUS over TCP [RFC6613], the present document on RADIUS over TLS and RADIUS over Datagram Transport Layer Security (DTLS) [RFC7360].

RADIUS/TLS does not specify any inherent backward compatibility to RADIUS/UDP or cross compatibility to the other transports, i.e., an implementation that utilizes RADIUS/TLS only will not be able to receive or send RADIUS packet payloads over other transports. An implementation wishing to be backward or cross compatible (i.e., wishes to serve clients using other transports than RADIUS/TLS) will need to implement these other transports along with the RADIUS/TLS transport and be prepared to send and receive on all implemented transports, which is called a "multi-stack implementation".

If a given IP device is able to receive RADIUS payloads on multiple transports, this may or may not be the same instance of software, and it may or may not serve the same purposes. It is not safe to assume that both ports are interchangeable. In particular, it cannot be assumed that state is maintained for the packet payloads between the transports. Two such instances MUST be considered separate RADIUS server entities.

5. Security Considerations

The computational resources to establish a TLS tunnel are significantly higher than simply sending mostly unencrypted UDP datagrams. Therefore, clients connecting to a RADIUS/TLS node will more easily create high load conditions and a malicious client might create a Denial-of-Service attack more easily.

Some TLS cipher suites only provide integrity validation of their payload and provide no encryption. This specification forbids the use of such cipher suites. Since the RADIUS payload's shared secret is fixed to the well-known term "radsec", failure to comply with this requirement will expose the entire datagram payload in plaintext, including User-Password, to intermediate IP nodes.

By virtue of being based on TCP, there are several generic attack vectors to slow down or prevent the TCP connection from being established; see [RFC4953] for details. If a TCP connection is not up when a packet is to be processed, it gets re-established, so such attacks in general lead only to a minor performance degradation (the time it takes to re-establish the connection). There is one notable exception where an attacker might create a bidding-down attack though. If peer communication between two devices is configured for both RADIUS/TLS and RADIUS/UDP, and the RADIUS/UDP transport is the failover option if the TLS session cannot be established, a bidding-down attack can occur if an adversary can maliciously close the TCP connection or prevent it from being established. Situtations where clients are configured in such a way are likely to occur during a migration phase from RADIUS/UDP to RADIUS/TLS. By preventing the TLS session setup, the attacker can reduce the security of the packet payload from the selected TLS cipher suite packet encryption to the classic MD5 per-attribute encryption. The situation should be avoided by disabling the weaker RADIUS/UDP transport as soon as the new RADIUS/TLS connection is established and tested.

RADIUS/TLS provides authentication and encryption between RADIUS peers. In the presence of proxies, the intermediate proxies can still inspect the individual RADIUS packets, i.e., "end-to-end" encryption is not provided. Where intermediate proxies are untrusted, it is desirable to use other RADIUS mechanisms to prevent RADIUS packet payload from inspection by such proxies. One common method to protect passwords is the use of the Extensible Authentication Protocol (EAP) and EAP methods that utilize TLS.

For dynamic discovery, this document allows the acceptance of a certificate only after doing PKIX checks. When using publicly trusted CAs as trust anchor, this may lead to security issues, since an advisary may easily get a valid certificate from this CAs. In current practice of [RFC6614], this problem is circumvented by using a private CA as a trust anchor. This private CA only issues certificate to members of the roaming consortium. This may still enable a malicious member to intercept traffic not intended for them, however, depending on the size of the consortium, this attack vector may be negligible. If the private CA also issues certificates for other purposes than RADIUS/TLS, the RADIUS/TLS certificates SHOULD include RADIUS/TLS-specific attributes against the implementation can check such as a X.509v3 Certificate Policy specific for RADIUS/TLS.

When using certificate fingerprints to identify RADIUS/TLS peers, any two certificates that produce the same hash value (i.e., that have a hash collision) will be considered the same client. Therefore, it is important to make sure that the hash function used is cryptographically uncompromised so that an attacker is very unlikely

to be able to produce a hash collision with a certificate of his choice. While this specification mandates support for SHA-1, a later revision will likely demand support for more contemporary hash functions because as of issuance of this document, there are already attacks on SHA-1.

6. IANA Considerations

Upon approval, IANA should update the Reference to radsec in the Service Name and Transport Protocol Port Number Registry:

* Service Name: radsec
* Port Number: 2083
* Transport Protocol: tcp
* Description: Secure RADIUS Service
* Assignment notes: The TCP port 2083 was already previously assigned by IANA for "RadSec", an early implementation of RADIUS/TLS, prior to issuance of the experimental RFC 6614. [This document] updates RFC 6614, while maintaining backward compatibility, if configured. For further details see RFC 6614, Appendix A or [This document], Appendix C.

7. References

7.1. 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>.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/info/rfc6613>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <https://www.rfc-editor.org/info/rfc7250>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

7.2. Informative References

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC4953] Touch, J., "Defending TCP Against Spoofing Attacks", RFC 4953, DOI 10.17487/RFC4953, July 2007, <https://www.rfc-editor.org/info/rfc4953>.

[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>.

[RFC5247] Aboba, B., Simon, D., and P. Eronen, "Extensible Authentication Protocol (EAP) Key Management Framework", RFC 5247, DOI 10.17487/RFC5247, August 2008, <https://www.rfc-editor.org/info/rfc5247>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC7360] DeKok, A., "Datagram Transport Layer Security (DTLS) as a Transport Layer for RADIUS", RFC 7360, DOI 10.17487/RFC7360, September 2014, <https://www.rfc-editor.org/info/rfc7360>.

[RFC7585] Winter, S. and M. McCauley, "Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS Based on the Network Access Identifier (NAI)", RFC 7585, DOI 10.17487/RFC7585, October 2015, <https://www.rfc-editor.org/info/rfc7585>.

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

Appendix A. Lessons learned from deployments of the Experimental [RFC6614]

There are at least two major (world-scale) deployments of [RFC6614].

A.1. eduroam

eduroam is a globally operating Wi-Fi roaming consortium exclusively for persons in Research and Education. For an extensive background on eduroam and its authentication fabric architecture, refer to [RFC7593].

Over time, more than a dozen out of 100+ national branches of eduroam used RADIUS/TLS in production to secure their country-to-country RADIUS proxy connections. This number is big enough to attest that the protocol does work, and scales. The number is also low enough to wonder why RADIUS/UDP continued to be used by a majority of country deployments despite its significant security issues.

Operational experience reveals that the main reason is related to the choice of PKIX certificates for securing the proxy interconnections. Compared to shared secrets, certificates are more complex to handle in multiple dimensions:

* Lifetime: PKIX certificates have an expiry date, and need administrator attention and expertise for their renewal

* Validation: The validation of a certificate (both client and server) requires contacting a third party to verify the recovaction status. This either takes time during session setup (OCSP checks) or requires the presence of a fresh CRL on the server - this in turn requires regular update of that CRL.

* Issuance: PKIX certificates carry properties in the Subject and extensions that need to be vetted. Depending on the CA policy, a certificate request may need significant human intervention to be verified. In particular, the authorisation of a requester to operate a server for a particular NAI realm needs to be verified. This rules out public "browser-trusted" CAs; eduroam is operating a special-purpose CA for eduroam RADIUS/TLS purposes.

* Automatic failure over time: CRL refresh and certificate renewal must be attended to regularly. Failure to do so leads to failure of the authentication service. Among other reasons, employee churn with incorrectly transferred or forgotten responsibilities is a risk factor.

It appears that these complexities often outweigh the argument of improved security; and a fallback to RADIUS/UDP is seen as the more appealing option.

It can be considered an important result of the experiment in [RFC6614] that providing less complex ways of operating RADIUS/TLS are required. The more thoroughly specified provisions in the current document towards TLS-PSK and raw public keys are a response to this insight.

On the other hand, using RADIUS/TLS in combination with Dynamic Discovery as per [RFC7585] necessitates the use of PKIX certificates. So, the continued ability to operate with PKIX certificates is also important and cannot be discontinued without sacrificing vital funcionality of large roaming consortia.

A.2. Wireless Broadband Alliance's OpenRoaming

OpenRoaming is a globally operating Wi-Fi roaming consortium for the general public, operated by the Wireless Broadband Alliance (WBA). With its (optional) settled usage of hotspots, the consortium requires both RADIUS authentication as well as RADIUS accounting.

The consortium operational procedures were defined in the late 2010s when [RFC6614] and [RFC7585] were long available. The consortium decided to fully base itself on these two RFCs.

In this architecture, using PSKs or raw public keys is not an option. The complexities around PKIX certificates as discussed in the previous section are believed to be controllable: the consortium operates its own special-purpose CA and can rely on a reliable source of truth for operator authorisation (becoming an operator requires a paid membership in WBA); expiry and revocation topics can be expected to be dealt with as high-priority because of the monetary implications in case of infrastructure failure during settled operation.

A.3. Participating in more than one roaming consortium

It is possible for a RADIUS/TLS (home) server to participate in more than one roaming consortium, i.e. to authenticate its users to multiple clients from distinct consortia, which present client certificates from their respective consortium's CA; and which expect the server to present a certificate from the matching CA.

The eduroam consortium has chosen to cooperate with (the settlement-free parts of) OpenRoaming to allow eduroam users to log in to (settlement-free) OpenRoaming hotspots.

eduroam RADIUS/TLS servers thus may be contacted by OpenRoaming clients expecting an OpenRoaming server certificate, and by eduroam clients expecting an eduroam server certificate.

It is therefore necessary to decide on the certificate to present during TLS session establishment. To make that decision, the availability of Trusted CA Indication in the client TLS message is important.

It can be considered an important result of the experiment in [RFC6614] that Trusted CA Indication is an important asset for inter-connectivity of multiple roaming consortia.

Appendix B. Interoperable Implementations

[RFC6614] is implemented and interoperates between at least three server implementations: FreeRADIUS, radsecproxy, Radiator. It is also implemented among a number of Wireless Access Points / Controllers from numerous vendors, including but not limited to: Aruba Networks, LANCOM Systems.

Appendix C. Backward compatibility

TODO describe necessary steps to configure common servers for compatibility with this version. Hopefully the differences to [RFC6614] are small enough that almost no config change is necessary.

Acknowledgments

Thanks to the original authors of RFC 6614: Stefan Winter, Mike McCauley, Stig Venaas and Klaas Vierenga.

TODO more acknowledgements

Authors' Addresses

Jan-Frederik Rieckers
Deutsches Forschungsnetz | German National Research and Education Network
Alexanderplatz 1
10178 Berlin
Germany
Email: rieckers@dfn.de
URI: www.dfn.de

Stefan Winter
Fondation Restena | Restena Foundation
2, avenue de l'Université
L-4365 Esch-sur-Alzette
Luxembourg
Email: stefan.winter@restena.lu
URI: www.restena.lu