Workgroup: RADEXT Working Group
Internet-Draft: draft-dekok-radext-radiusv11-05
Updates: 6613 7360 (if approved)
Published: 19 April 2023
Intended Status: Standards Track
Expires: 21 October 2023
Authors: A. DeKok
FreeRADIUS

RADIUS Version 1.1

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.

	Server			
Client	no ALPN	forbid	allow	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.

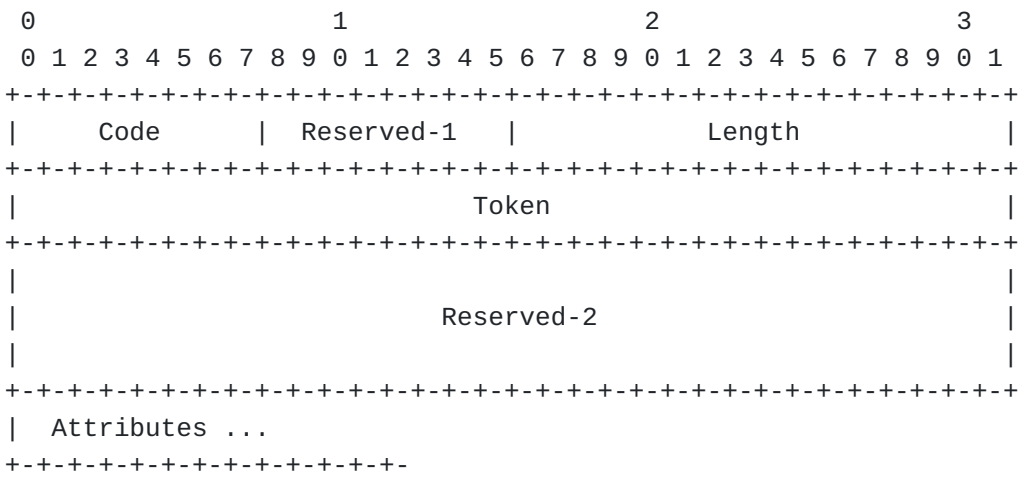

Figure 2: The RADIUS/1.1 Packet Format

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