Workgroup: RADEXT Working Group
Internet-Draft: draft-ietf-radext-radiusv11-02
Updates: 6614, 7360, 7930 (if approved)
Published: 11 October 2023
Intended Status: Experimental
Expires: 13 April 2024
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-ietf-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/radext-wg/draft-ietf-radext-radiusv11.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 13 April 2024.

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 that MD5 should no longer be used. These discussions are most notably in [RFC6151], and in Section 3 of [RFC6421], among others. 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.

While RADIUS originally used UDP transport, additional transport protocols were defined for TCP ([RFC6613]), TLS ([RFC6614]), and DTLS ([RFC7360]). However, those transport protocols 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 over (D)TLS which removes their 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 code changes are required to support RADIUS/1.1 on top of an existing RADIUS server.

While this document permits MD5 to be removed when using (D)TLS transports, it makes no changes to UDP or TCP transports. It is therefore RECOMMENDED that those transports only be used within secure networks, and only used in situations where FIPS compliance is not an issue.

In most cases, using ALPN requires only a few modifications to the RADIUS/TLS protocol implementation:

- A method to set the list of supported ALPN protocols before the TLS handshake starts

- After the TLS handshake has completed, a method to query if ALPN has chosen a protocol, and if yes, which protocol was chosen.

- Changes to the packet encoder and decoder, so that the individual packets are not signed, and no attribute is encoded with the historic obfuscation methods.

That is, the bulk of the ALPN protocol can be left to the underlying TLS implementation. This document discusses the ALPN exchange in detail in order to give simplified descriptions for the reader, and so that the reader does not have to read or understand all of [RFC7301].

The detailed list of changes from historic TLS-based transports to RADIUS/1.1 is 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,

*The default 4K packet size is unchanged, although [RFC7930] can still be leveraged to use larger packets,

*All attributes which have simple encodings (i.e. without using MD5 obfuscation), all have the same encoding and 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 who 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 be fully FIPS-140 compliant. 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. 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. It is therefore 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 in RADIUS/1.1.

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 document defines how to transform RADIUS into RADIUS/1.1, and no further discussion of that transformation is necessary.

We note that this document makes no changes to previous RADIUS specifications. Existing RADIUS implementations can continue to be used without modification. Where previous specifications are explicitly mentioned and updated, those updates or changes apply only when the RADIUS/1.1 transport profile is being used.

In short, when negotiated on a connection, the RADIUS/1.1 transport profile permits implementations to avoid MD5 when signing packets, or when 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].

*historic RADIUS/TLS

RADIUS/TLS as defined in [RFC6614] and [RFC7360].

*RADIUS

The Remote Authentication Dial-In User Service protocol, as defined in [RFC2865], [RFC2866], 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", but which does not signal 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 (ALPN or otherwise), 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 did not signal ALPN, or the server does not accept the ALPN proposal, the server does not reply with any ALPN name.

4) The client receives the ServerHello, validates the received 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 one which 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 "Version" 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 implementation signals use of this protocol via ALPN.

When set, this flag contains the list of permitted ALPN versions in humanly readable form. The implementation may allow multiple values in one variable, or allow multiple variables, or instead use two configuration for "minimum" and "maximum" allowed versions. We assume here that there is one variable, which can be configured as:

- *unset,
- *containing value "1.0" - forbid RADIUS/1.1.
- *containing values "1.0" and "1.1" - allow either historic RADIUS/TLS or RADIUS/1.1
- *containing value "1.1" - require RADIUS/1.1.

This configuration is also extensible to future ALPN names if that extension becomes necessary.

A more descriptive definition of the variable and the meaning of the values is given below.

Configuration Flag Name

   Version

Values

When the flag is unset, ALPN is not used.

Any connection MUST use historic RADIUS/TLS.

Client Behavior

The client MUST NOT send any protocol name via ALPN.

Server Behavior

The server MUST NOT signal any protocol name via ALPN.

If the server receives an ALPN name from the client, it MUST NOT close the connection. Instead, it simply does not reply with ALPN, and finishes the TLS connection setup as defined for historic RADIUS/TLS.

Note that if the client sends "radius/1.1", the client will see that the server failed to acknowledge this request, and will close the connection. For any other client configuration, the connection will use historic RADIUS/TLS.

"1.0" - send "radius/1.0", and use historical RADIUS/TLS.

When the "Version" configuration flag is set to "1.0", the system will send the ALPN string "radius/1.0". However, everything else about the connection is identical to historic RADIUS/TLS.

This behavior is used to help administrators distinguish between systems which can use ALPN from ones which cannot use ALPN. The act of sending the name "radius/1.0" is an implicit statement that the system is likely to also support "radius/1.1".

Client Behavior

The client MUST send only the ALPN string "radius/1.0".

The client will receive either no ALPN response from the server, or an ALPN response of "radius/1.0", or a TLS alert of "no_application_protocol" (120).

If the connection remains open, the client MUST use historic RADIUS/TLS.

Server Behavior

If the server receives no ALPN name from the client, it MUST use historic RADIUS/TLS.

If the server receives an ALPN name "radius/1.0" from the client, it MUST reply with ALPN "radius/1.0", and then use historic RADIUS/TLS.

Note that the server may receive multiple ALPN names from the client. So long as the server receives "radius/1.0", it can reply with "radius/1.0".

If the server receives one or more ALPN names from the client, but none of the names match "radius/1.0", it MUST reply with a TLS alert of "no_application_protocol" (120), and then close the TLS connection.

"1.0, 1.1" - Negotiate historic RADIUS/TLS or RADIUS/1.1

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

Connections MAY use either historic RADIUS/TLS or RADIUS/1.1.

Client Behavior

The client MUST send both "radius/1.0" and "radius/1.1" via ALPN.

If the client receives no ALPN response from the server, or an ALPN response of "radius/1.0", it MUST use historic RADIUS/TLS.

If the client receives an ALPN response of "radius/1.1" from the server, it MUST use RADIUS/1.1.

Otherwise the client receives a TLS alert of "no_application_protocol" (120) from the server, and the connection is terminated.

Server Behavior

If the server receives no ALPN name from the client, it MUST use historic RADIUS/TLS.

If the server receives an ALPN name "radius/1.0" from the client, it MUST reply with ALPN "radius/1.0", and then use historic RADIUS/TLS.

Note that the server may receive multiple ALPN names from the client. So long as the server receives an ALPN name "radius/1.0" from the client, it is deemed to match, and the connection MUST use historic RADIUS/TLS.

If the server receives one or more ALPN names from the client, but none of the names match "radius/1.0", it MUST reply with a TLS alert of "no_application_protocol" (120), and then close the TLS connection.

"1.1" - Require the use of RADIUS/1.1

Any connection MUST use RADIUS/1.1

Client Behavior

The client MUST send only the ALPN string "radius/1.1".

If the client receives no ALPN response from the server, or an ALPN response of anything other than "radius/1.1", it MUST close the TLS connection.

If the client receives an ALPN response of "radius/1.1" from the server, it MUST use RADIUS/1.1.

Otherwise the client receives a TLS alert of "no_application_protocol" (120) from the server, and the connection is terminated.

Server Behavior

If the server receives no ALPN name from the client, it MUST reply with a TLS alert of "no_application_protocol" (120), and then close the TLS connection.

If the server receives an ALPN name "radius/1.0" from the client, it MUST reply with a TLS alert of "no_application_protocol" (120), and then close the TLS connection.

Note that the server may receive multiple ALPN names from the client. So long as the server receives an ALPN name "radius/1.1" from the client, it is deemed to match, and the connection MUST use RADIUS/1.1.

If the server receives one or more ALPN names from the client, but none of the names match "radius/1.1", it MUST reply with a TLS alert of "no_application_protocol" (120), and then close the TLS connection.

By requiring the "allow" setting to be the default, implementations will be compatible with both historic RADIUS/TLS connections, and with RADIUS/1.1 connections. It is therefore the only default setting which will not result in connection errors.

Once administrators verify that both ends of a connection support RADIUS/1.1, and that it has been negotiated successfully, the configurations SHOULD be updated to require RADIUS/1.1. The connections should be monitored after this change to ensure that the systems continue to remain connected. If there are connection issues, then the configuration should be reverted to using "allow", until such time as the connection problems have been resolved.

We reiterate that systems implementing this specification, but configured with setting which forbid RADIUS/1.1, will behave exactly the same as systems which do not implement this specification. Systems implementing RADIUS/1.1 SHOULD NOT be configured by default to forbid that protocol. That setting exists mainly for completeness, and to give administrators the flexibility to control their own deployments.

If a 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 to the other end that there is no compatible application protocol. It MUST then close the connection. This requirement applies to both new sessions, and to resumed sessions.

While [RFC7301] does not discuss the possibility of the server sending a TLS alert of "no_application_protocol" (120) when the client does not use ALPN, we believe that this behavior is useful. As such, servers MAY send a a TLS alert of "no_application_protocol" (120) when the client does not use ALPN. We recognize that this behavior may not always be possible or available in any underlying TLS implementation. The server MAY send this alert during the ClientHello, if it requires ALPN but does not receive it. That is, there may not always be a need to wait for the TLS connection to be fully established before realizing that no common ALPN protocol can be negotiated.

In contrast, there is no need for the client to signal that there are no compatible application protocol names. The client sends zero or more protocol names, and the server responds as above. From the point of view of the client, the list it sent results in either a connection failure, or a connection success.

It is RECOMMENDED that the server logs a descriptive error 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, when the client receives 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 only difference in behavior between the two values for the client is how it handles responses 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 only difference in behavior between the two values for the server is how it handles the situation when no ALPN is signaled from the client.

Unfortunately when ALPN negotiation fails, it is not always possible to send TLS alert of "no_application_protocol" (120). [RFC7301] Section 3.2 suggests that this alert can only be sent by the server which supports ALPN, in response to a client which requests ALPN. However, if either party does not support ALPN, then there are no provisions for this alert to be sent. In addition, the TLS implementations may not permit an application to send a TLS alert of its choice, at a time of its choice. So if one party supports ALPN while the other does not, it is not possible for the system supporting ALPN to send any kind of TLS alert which informs the other party that ALPN is required.

3.3.1. Using Protocol-Error for Application Signaling

When it is not possible to send a TLS alert of "no_application_protocol" (120), then the only remaining method for one party to signal the other is to send application data inside of the TLS tunnel. Therefore, for the situation when a one end of a connection determines that it requires ALPN while the other end does not support ALPN, the end requiring ALPN MAY send a Protocol-Error packet inside of the tunnel, and then close the connection. If this is done, the Response Authenticator field of the Protocol-Error packet MUST be all zeros, as this packet is not in response to any request. The Protocol-Error packet SHOULD contain a Reply-Message attribute with a textual string describing the cause of the error. The packet SHOULD also contain an Error-Cause attribute, with value Unsupported Extension (406).

An implementation sending this packet could bypass any RADIUS encoder, and simply write this packet as a predefined, fixed set of

data to the TLS connection. That process would likely be simpler than trying to call the normal RADIUS packet encoder to encode a reply packet without a request packet, and then trying to force the Response Authenticator to be all zeros.

As this packet is an unexpected response packet, existing implementations will ignore it. They may either log an error and close the connection, or they may discard the packet and leave the connection open. If the connection remains open, the end supporting ALPN will close the connection, so there will be no side effects from sending the packet. Therefore, while using a Protocol-Error packet in this way is unusual, it is both informative and safe.

The purpose of this packet is not to have the other end of the connection automatically determine what went wrong, and fix it. Instead, the packet is intended to be (eventually) seen by an administrator, who can then take remedial action.

3.3.2. 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 Version 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	1.0	1.0, 1.1	1.1
No ALPN	RadSec	RadSec	RadSec	Close-S
1.0	RadSec	1.0	1.0	Alert
1.0, 1.1	RadSec	1.0	1.1	1.1
1.1	Close-C	Alert	1.1	1.1

Figure 1: Possible outcomes for ALPN Negotiation

The table entries above have the following meaning:

Alert

The client sends either no ALPN, or the server does not agree to the clients ALPN proposal. The server replies with a TLS alert of "no_application_protocol" (120), and then closes the TLS connection.

As noted in the previous section, the server may send a Protocol-Error packet to the client before closing the connection.

Close-C

The client sends ALPN, but the server does not respond with ALPN. The client closes the connection.

As noted in the previous section, the client may send a Protocol-Error packet to the server before closing the connection.

Close-S

The client does not send ALPN string(s), but the server requires ALPN. The server closes the connection.

As noted in the previous section, the server may send a Protocol-Error packet to the client before closing the connection.

RadSec

Historic RADIUS/TLS is used. Either the client sends no ALPN proposal, or the client sends an ALPN proposal, and the server never replies with an ALPN string.

1.0

The client sends the ALPN string "radius/1.0". The server responds with the ALPN string "radius/1.0", and then historic RADIUS/TLS is used

1.1

The client sends the ALPN string "radius/1.1. The server ACKs with "radius/1.1", and then RADIUS/1.1 is used.

Implementations should note that this table may be extended in future specifications. The above text is informative, and does not mandate that only the above ALPN strings are used. The actual ALPN negotiation takes place as defined in the preceding sections of this document, and in [RFC7301].

3.4. Miscellaneous Items

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

The use of the ALPN string "radius/1.0" is technically unnecessary, as it is largely equivalent to not sending any ALPN string. However, that value is useful for RADIUS administrators. A system which sends the ALPN string "radius/1.0" is explicitly signaling that it supports ALPN negotiation, but that it is not currently configured to support RADIUS/1.1. That information can be used by administrators to determine which devices are capable of ALPN.

The use of the ALPN string "radius/1.0" also permits server implementations to send a TLS alert of "no_application_protocol" (120) when it cannot find a matching ALPN string. Experiments with TLS library implementations suggest that in some cases it is possible to send that TLS alert when ALPN is not used. However, such a scenario is not discussed on [RFC7301], and is likely not universal. As a result, ALPN as defined in [RFC7301] permits servers to send that TLS alert in situations where it would be otherwise forbidden, or perhaps unsupported.

Finally, defining ALPN strings for all known RADIUS versions will make it easier to support additional ALPN strings if that functionality is ever needed.

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 systems which negotiate the "radius/1.1" protocol MUST associate that information with the session ticket, and enforce the use of "radius/1.1" on session resumption. That is, if "radius/1.1" was negotiated for a session, both clients and servers MUST behave as if the RADIUS/1.1 flag was set to "require" for that session.

A client which is resuming a "radius/1.1" connection MUST advertise only the capability to do "radius/1.1" for the resumed session. That is, even if the client configuration is "allow" for new connections, it MUST signal "radius/1.1" when resuming a session which had previously negotiated "radius/1.1".

Similarly, when a server does resumption for a session which had previously negotiated "radius/1.1", If the client attempts to resume the sessions without signaling the use of RADIUS/1.1, the server MUST close the connection. The server MUST send an appropriate TLS error, and also SHOULD log a descriptive message as described above.

In contrast, there is no requirement for a client or server to force the use of [RFC6614] RADIUS/TLS on session resumption. Clients are free to signal support for "radius/1.1" on resumed sessions, even if the original session did not negotiate "radius/1.1". Servers are free to accept this request, and to negotiate the use of "radius/1.1" for such sessions.

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 / Response Authenticator fields are no longer used in RADIUS/1.1. 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 in RADIUS/1.1.

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

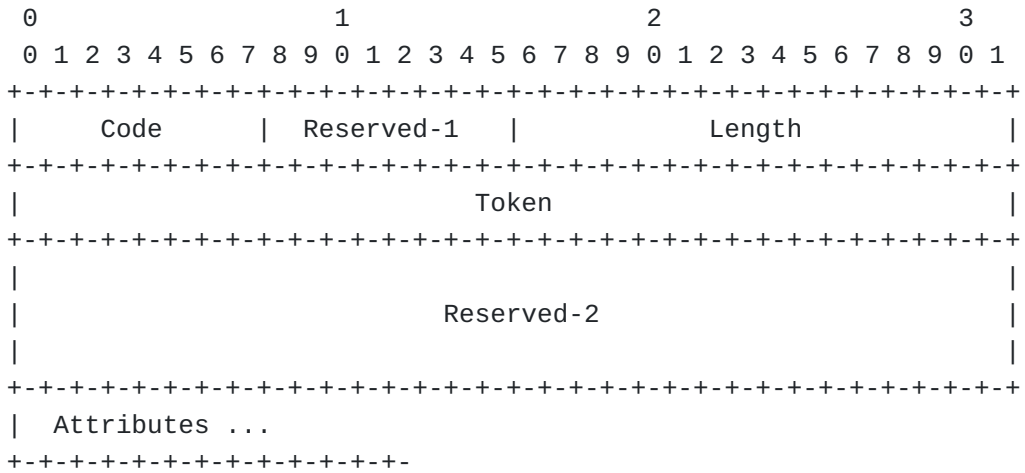

Figure 2: The RADIUS/1.1 Packet Format

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 replaces it as the way to identify and requests, and to associate responses with requests. The Reserved-1 field MUST be ignored when receiving a packet.

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

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 that the client sends.

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. The only cost for tracking Tokens is 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 that 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 for RADIUS/1.1. 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 the clear" across the network. This allegation was never true for RADIUS, which obfuscated passwords on the wire. This allegation is definitely untrue when (D)TLS transport is used. While passwords are encoded in packets as strings, the entire RADIUS

exchange including packets, attributes (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 categorically false.

There are risks from sending passwords over the network, even when they are protected by TLS. One such risk comes 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.

There are 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, user 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.

We note that as the RADIUS shared secret is no longer used in this specification, 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.

5.1.1. 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 be 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 5.3 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, it is no longer possible to use the Request Authenticator field as the CHAP-Challenge when this transport profile is used.

Clients which send CHAP-Password attribute ([RFC2865] Section 5.3) in an Access-Request packet over a RADIUS/1.1 connection MUST also include a CHAP-Challenge attribute ([RFC2865] Section 5.40).

Proxies may need to Access-Request packets over a non-RADIUS/1.1 transport and then forward those packets over a RADIUS/1.1 connection. In that case, if the received Access-Request packet contains a CHAP-Password attribute but no CHAP-Challenge attribute, the proxy MUST create a CHAP-Challenge attribute in the proxied packet using the contents from the incoming Request Authenticator of the received packet.

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 'string' which contains a tag, such as Tunnel-Client-Endpoint ([RFC2868] Section 3.3). Since the attribute is no longer obfuscated in RADIUS/1.1, 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 be 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 'string' ([RFC8044] Section 3.4).

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 not used or needed in RADIUS/1.1.

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 for the RADIUS/1.1 transport. 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.

For proxies, the Message-Authenticator attribute was always defined as being created and consumed on a "hop by hop" basis. That is, a proxy which received a Message-Authenticator attribute from a client would never forward that attribute as-is to another server. Instead, the proxy would either suppress, or re-create, the Message-Authenticator attribute in the outgoing request. This existing behavior is leveraged in RADIUS/1.1 to suppress the use of Message-Authenticator over a RADIUS/1.1 connection.

A proxy may receive an Access-Request packet over a RADIUS/1.1 connection, and then forward that packet over a RADIUS/UDP or a RADIUS/TCP connection. In that situation, the proxy SHOULD add a Message-Authenticator attribute to every Access-Request packet which is sent over an insecure transport protocol.

The original text in [RFC3579] Section 3.3, "Note 1" paragraph required that a server suggested that the Message-Authenticator

attribute be present for certain Access-Request packets. It also required the use of Message-Authenticator when the Access-Request packet contained an EAP-Message attribute. Experience has shown that some RADIUS clients never use the Message-Authenticator, even for the situations where its use is suggested.

When the Message-Authenticator attribute is missing from Access-Request packets, it is often possible to trivially forge or replay those packets. As such, this document RECOMMENDS that RADIUS clients always include Message-Authenticator in Access-Request packets when using UDP or TCP transport. As the scope of this document is limited to defining RADIUS/1.1, we cannot mandate that behavior here. Instead, we can note that there are no known negatives to this behavior, and there are definite positives, such as increased security.

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 in RADIUS/1.1, the related MAC-Randomizer attribute [RFC6218] Section 3.2 is also no longer used. It MUST also be treated the same way 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

[RFC7930] Section 4 defines an Original-Packet-Code attribute. This attribute is needed because otherwise it is impossible to correlate the Protocol-Error response packet with a particular request packet.

The definition in [RFC7930] Section 4 describes the reasoning behind this need:

> The Original-Packet-Code contains the code from the request that generated the protocol error so that clients can disambiguate requests with different codes and the same ID.

This attribute is no longer needed in RADIUS/1.1. The Identifier field is unused, so it impossible for two requests to have the "same" ID. Instead, the Token field permits clients and servers to correlate requests and responses, independent of the Code being used.

Therefore, the Original-Packet-Code attribute ([RFC7930] Section 4) MUST NOT be sent over a RADIUS/1.1 connection. That attribute is not used or needed over RADIUS/1.1 connections.

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 not needed in RADIUS/1.1 to correlate Protocol-Error replies with outstanding requests.

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 not used in this transport profile.

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 packets. 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 outstanding. The safest way forward in that case 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 [RFC7593] and [EDUROAM] and in OpenRoaming [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, and once implementations support it, 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

down-bidding 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 following section, this specification also bans the development of all new cryptographic or crypto-agility methods in the RADIUS protocol.

6.4. Error-Cause Attribute

The Error-Cause attribute is defined in [RFC5176]. The "Table of Attributes" section given in [RFC5176] Section 3.6 permits that attribute to appear in CoA-NAK and Disconnect-NAK packets. As no other packet type is listed, the implication is that the Error-Cause attribute cannot appear in any other packet. [RFC7930] also permits Error-Cause to appear in Protocol-Error packets.

However, [RFC5080] Section 2.6.1 suggests that Error-Cause may appear in Access-Reject packets. No reference is given for this change from [RFC5176]. There is not even an acknowledgment that this suggestion is a change from any previous specification. We correct that issue here.

This specification updates [RFC5176] to allow the Error-Cause attribute to appear in Access-Reject packets. It is RECOMMENDED that implementations include the Error-Cause attribute in Access-Reject packets where appropriate.

That is, the reason for sending the Access-Reject packet (or Protocol-Error packet) may match a defined Error-Cause value. In that case, it is useful for implementations to send an Error-Cause attribute with that value. This behavior can help RADIUS system administrators debug issues in complex proxy chains.

For example, a proxy may normally forward Access-Request packets which contain EAP-Message attributes. The proxy can determine if the contents of the EAP-Message are invalid, for example if the first octet has value larger than 4. In that case, there may be no benefit to forwarding the packet, as the home server will reject it. It may then then possible for the proxy (with the knowledge and consent of involved parties) to immediately reply with an Access-Reject containing an Error-Cause attribute with value 202 for "Invalid EAP Packet (Ignored)".

Another possibility is that if a proxy is configured to forward packets for a particular realm, but it has determined that there are no available connections to the next hop for that realm. In that case, it is may be possible for the proxy (again with the knowledge

and consent of involved parties) to reply with an Access-Reject containing an Error-Cause attribute with value 502 for "Request Not Routable (Proxy)"

These examples are given only for illustrative and informational purposes. While it is useful to return an informative value for the Error-Cause attribute, proxies can only modify the traffic they forward with the explicit knowledge and consent of all involved parties.

6.5. Future Standards

Future work may define new attributes, packet types, etc. It is important to be able to do such work without requiring that every new standard mention RADIUS/1.1 explicitly. Instead, this document defines a mapping from RADIUS to RADIUS/1.1 which covers all RADIUS practices and cryptographic primitives in current use. As a result, any new standard which uses the existing RADIUS practices can simply inherit that mapping, and they do not need to mention RADIUS/1.1 explicitly.

We reiterate that this specification defines a new transport profile for RADIUS. It does not define a completely new protocol. Any future specification which defines a new attribute MUST define it for RADIUS/UDP first, 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).

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 signaling 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.

The result is that 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.

We note that it is theoretically possible for future standards to define new cryptographic primitives for use with RADIUS/UDP. In that case, those documents would likely have to describe how to transport that data in RADIUS/1.1. We believe that such standards are unlikely to be published, as other efforts in the RADEXT working group are forbidding such updates to RADIUS.

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. Acknowledgments

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 Richardson, Hannes Tschofenig, Matthew Newton, and Josh Howlett 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.

draft-ietf-radext-radiusv11-00

No changes from previous draft.

draft-ietf-radext-radiusv11-01

Move to "experimental" based on WG feedback.

Many cleanups based on review from Matthew Newton

Removed requirement for supporting TLS-PSK.

This document does not deprecate new cryptographic work in RADIUS. The "deprecating insecure transports" document does that.

draft-ietf-radext-radiusv11-02

Note that we also update RFC 7930

Minor updates to text.

Add text explaining why "allow" is the default, and how to upgrade to "require"

Discuss the use of the TLS alert "no_application_protocol" (120), and its limitations.

Suggest the use of Protocol-Error as an application signal when it is not possible to send a "no_application_protocol" TLS alert.

Update discussion of Message-Authenticator, and suggest that RADIUS/1.1 proxies always add Message-Authenticator to Access-Request packets being sent over UDP or TCP.

Add term "historic RADIUS/TLS" as it is simpler than more awkward "6614 or 7360".

Re-add ALPN "radius/1.0" based on comments from Heikki. It signals that the system is ALPN-capable, among other.

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

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

[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>.

[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>.

[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>.

[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>.

[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>.

[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