

RADIUS Extensions Working Group
Internet-Draft
Intended status: Experimental
Expires: August 17, 2012

S. Winter
RESTENA
M. McCauley
OSC
S. Venaas
K. Wierenga
Cisco
February 14, 2012

Transport Layer Security (TLS) encryption for RADIUS
draft-ietf-radext-radsec-12

Abstract

This document specifies a transport profile for RADIUS using Transport Layer Security (TLS) over TCP as the transport protocol. This enables dynamic trust relationships between RADIUS servers.

Status of This Memo

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

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

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

This Internet-Draft will expire on August 17, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
1.1. Requirements Language	3
1.2. Terminology	4
1.3. Document Status	4
2. Normative: Transport Layer Security for RADIUS/TCP	5
2.1. TCP port and packet types	5
2.2. TLS negotiation	5
2.3. Connection Setup	5
2.4. Connecting Client Identity	7
2.5. RADIUS Datagrams	8
3. Informative: Design Decisions	10
3.1. Implications of Dynamic Peer Discovery	10
3.2. X.509 Certificate Considerations	10
3.3. Ciphersuites and Compression Negotiation Considerations	11
3.4. RADIUS Datagram Considerations	11
4. Compatibility with other RADIUS transports	12
5. Diameter Compatibility	13
6. Security Considerations	13
7. IANA Considerations	14
8. Notes to the RFC Editor	15
9. Acknowledgements	15
10. References	15
10.1. Normative References	15
10.2. Informative References	16
Appendix A. Implementation Overview: Radiator	18
Appendix B. Implementation Overview: radsecproxy	19
Appendix C. Assessment of Crypto-Agility Requirements	20

1. Introduction

The RADIUS protocol [RFC2865] is a widely deployed authentication and authorisation protocol. The supplementary RADIUS Accounting specification [RFC2866] also provides accounting mechanisms, thus delivering a full Authentication, Authorization, and Accounting (AAA) solution. However, RADIUS is experiencing several shortcomings, such as its dependency on the unreliable transport protocol UDP and the lack of security for large parts of its packet payload. RADIUS security is based on the MD5 algorithm, which has been proven to be insecure.

The main focus of RADIUS over TLS is to provide a means to secure the communication between RADIUS/TCP peers using TLS. The most important use of this specification lies in roaming environments where RADIUS packets need to be transferred through different administrative domains and untrusted, potentially hostile networks. An example for a world-wide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

There are multiple known attacks on the MD5 algorithm which is used in RADIUS to provide integrity protection and a limited confidentiality protection (see [MD5-attacks]). RADIUS over TLS wraps the entire RADIUS packet payload into a TLS stream and thus mitigates the risk of attacks on MD5.

Because of the static trust establishment between RADIUS peers (IP address and shared secret) the only scalable way of creating a massive deployment of RADIUS-servers under control by different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RADIUS over TLS obsolete the use of IP addresses and shared MD5 secrets to identify other peers and thus allow the use of more contemporary trust models, e.g. checking a certificate by inspecting the issuer and other certificate properties.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119. [RFC2119]

1.2. Terminology

RADIUS/TLS node: a RADIUS over TLS client or server

RADIUS/TLS Client: a RADIUS over TLS instance which initiates a new connection.

RADIUS/TLS Server: a RADIUS over TLS instance which listens on a RADIUS over TLS port and accepts new connections

RADIUS/UDP: classic RADIUS transport over UDP as defined in [RFC2865]

1.3. Document Status

This document is an Experimental RFC.

It is one out of several approaches to address known cryptographic weaknesses of the RADIUS protocol (see also Section 4). The specification does not fulfill all recommendations on a AAA transport profile as per [RFC3539]; in particular, by being based on TCP as a transport layer, it does not prevent head-of-line blocking issues.

If this specification is indeed selected for advancement to standards track, certificate verification options (section 2.3.2) need to be refined.

Another experimental characteristic of this specification is the question of key management between RADIUS/TLS peers. RADIUS/UDP only allowed for manual key management, i.e. distribution of a shared secret between a client and a server. RADIUS/TLS allows manual distribution of long-term proofs of peer identity as well (by using TLS-PSK cipher suites, or identifying clients by a certificate fingerprint), but as a new feature enables use of X.509 certificates in a PKIX infrastructure. It remains to be seen if one of these methods prevail, or if both will find their place in real-life deployments. The authors can imagine pre-shared keys to be popular in small-scale deployments (SOHO or isolated enterprise deployments) where scalability is not an issue and the deployment of a CA is considered too much a hassle; but can also imagine large roaming consortia to make use of PKIX. Readers of this specification are encouraged to read the discussion of key management issues within [RFC6421] as well as [RFC4107].

It has yet to be decided whether this approach is to be chosen for standards track. One key aspect to judge whether the approach is usable at large scale is by observing the uptake, usability and operational behaviour of the protocol in large-scale, real-life deployments.

An example for a world-wide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

2. Normative: Transport Layer Security for RADIUS/TCP

2.1. TCP port and packet types

The default destination port number for RADIUS over TLS is TCP/2083. There are no separate ports for authentication, accounting and dynamic authorisation changes. The source port is arbitrary. See section Section 3.4 for considerations regarding separation of authentication, accounting and dynamic authorization traffic.

2.2. TLS negotiation

RADIUS/TLS has no notion of negotiating TLS in an established connection. Servers and clients need to be preconfigured to use RADIUS/TLS for a given endpoint.

2.3. Connection Setup

RADIUS/TLS nodes

1. establish TCP connections as per [I-D.ietf-radext-tcp-transport]. Failure to connect leads to continuous retries, with exponentially growing intervals between every try. If multiple servers are defined, the node MAY attempt to establish a connection to these other servers in parallel, in order to implement quick failover.
2. after completing the TCP handshake, immediately negotiate TLS sessions according to [RFC5246] or its predecessor TLS 1.1. The following restrictions apply:
 - * Support for TLS v1.1 [RFC4346] or later (e.g. TLS 1.2 [RFC5246]) is REQUIRED. To prevent known attacks on TLS versions prior to 1.1, implementations MUST NOT negotiate TLS versions prior to 1.1.
 - * Support for certificate-based mutual authentication is REQUIRED.
 - * Negotiation of mutual authentication is REQUIRED.
 - * Negotiation of a ciphersuite providing for confidentiality as well as integrity protection is REQUIRED. Failure to comply with this requirement can lead to severe security problems, like user passwords being recoverable by third parties. See

Section 6 for details.

- * Support for and negotiation of compression is OPTIONAL.
 - * Support for TLS-PSK mutual authentication [RFC4279] is OPTIONAL.
 - * RADIUS/TLS implementations MUST at a minimum support negotiation of the TLS_RSA_WITH_3DES_EDE_CBC_SHA), and SHOULD support TLS_RSA_WITH_RC4_128_SHA and TLS_RSA_WITH_AES_128_CBC_SHA as well (see Section 3.3).
 - * In addition, RADIUS/TLS implementations MUST support negotiation of the mandatory-to-implement ciphersuites required by the versions of TLS that they support.
3. Peer authentication can be performed in any of the following three operation models:
- * TLS with X.509 certificates using PKIX trust models (this model is mandatory to implement):
 - + Implementations MUST allow to configure a list of trusted Certification Authorities for incoming connections.
 - + Certificate validation MUST include the verification rules as per [RFC5280].
 - + Implementations SHOULD indicate their trusted Certification Authorities (CAs). For TLS 1.2, this is done using [RFC5246] section 7.4.4 "certificate authorities" (server side) and [RFC6066] Section 6 "Trusted CA Indication" (client side). See also Section 3.2.
 - + Peer validation always includes a check on whether the locally configured expected DNS name or IP address of the server that is contacted matches its presented certificate. DNS names and IP addresses can be contained in the Common Name (CN) or subjectAltName entries. For verification, only one of these entries is to be considered. The following precedence applies: for DNS name validation, subjectAltName:DNS has precedence over CN; for IP address validation, subjectAltName:iPAddr has precedence over CN. Implementors of this specification are advised to read [RFC6125] Section 6 for more details on DNS name validation.

- + Implementations MAY allow to configure a set of additional properties of the certificate to check for a peer's authorisation to communicate (e.g. a set of allowed values in subjectAltName:URI or a set of allowed X509v3 Certificate Policies)
 - + When the configured trust base changes (e.g. removal of a CA from the list of trusted CAs; issuance of a new CRL for a given CA) implementations MAY re-negotiate the TLS session to re-assess the connecting peer's continued authorisation.
 - * TLS with X.509 certificates using certificate fingerprints (this model is optional to implement): Implementations SHOULD allow to configure a list of trusted certificates, identified via fingerprint of the DER encoded certificate octets. Implementations MUST support SHA-1 as the hash algorithm for the fingerprint. To prevent attacks based on hash collisions, support for a more contemporary hash function such as SHA-256 is RECOMMENDED.
 - * TLS using TLS-PSK (this model is optional to implement)
4. start exchanging RADIUS datagrams (note Section 3.4 (1)). The shared secret to compute the (obsolete) MD5 integrity checks and attribute encryption MUST be "radsec" (see Section 3.4 (2)).

2.4. Connecting Client Identity

In RADIUS/UDP, clients are uniquely identified by their IP address. Since the shared secret is associated with the origin IP address, if more than one RADIUS client is associated with the same IP address, then those clients also must utilize the same shared secret, a practice which is inherently insecure as noted in [RFC5247].

RADIUS/TLS supports multiple operation modes.

In TLS-PSK operation, a client is uniquely identified by its TLS identifier.

In TLS-X.509 mode using fingerprints, a client is uniquely identified by the fingerprint of the presented client certificate.

In TLS-X.509 mode using PKIX trust models, a client is uniquely identified by the tuple (serial number of presented client certificate;Issuer).

Note well: having identified a connecting entity does not mean the

server necessarily wants to communicate with that client. E.g. if the Issuer is not in a trusted set of Issuers, the server may decline to perform RADIUS transactions with this client.

There are numerous trust models in PKIX environments, and it is beyond the scope of this document to define how a particular deployment determines whether a client is trustworthy. Implementations which want to support a wide variety of trust models should expose as many details of the presented certificate to the administrator as possible so that the trust model can be implemented by the administrator. As a suggestion, at least the following parameters of the X.509 client certificate should be exposed:

- o Originating IP address
- o Certificate Fingerprint
- o Issuer
- o Subject
- o all X509v3 Extended Key Usage
- o all X509v3 Subject Alternative Name
- o all X509v3 Certificate Policies

In TLS-PSK operation, at least the following parameters of the TLS connection should be exposed:

- o Originating IP address
- o TLS Identifier

2.5. RADIUS Datagrams

Authentication, Accounting and Authorization packets are sent according to the following rules:

RADIUS/TLS clients transmit the same packet types on the connection they initiated as a RADIUS/UDP client would (see Section 3.4 (3) and (4)). E.g. they send

- o Access-Request
- o Accounting-Request

- o Status-Server
- o Disconnect-ACK
- o Disconnect-NAK
- o ...

and they receive

- o Access-Accept
- o Accounting-Response
- o Disconnect-Request
- o ...

RADIUS/TLS servers transmit the same packet types on connections they have accepted as a RADIUS/UDP server would. E.g. they send

- o Access-Challenge
- o Access-Accept
- o Access-Reject
- o Accounting-Response
- o Disconnect-Request
- o ...

and they receive

- o Access-Request
- o Accounting-Request
- o Status-Server
- o Disconnect-ACK
- o ...

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

Section 3.4 for a discussion of signaling.

- o When receiving an unwanted packet of type 'CoA-Request' or 'Disconnect-Request', it needs to be replied to with a 'CoA-NAK' or 'Disconnect-NAK' respectively. The NAK SHOULD contain an attribute Error-Cause with the value 406 ("Unsupported Extension"); see [RFC5176] for details.
- o When receiving an unwanted packet of type 'Accounting-Request', the RADIUS/TLS server SHOULD reply with an Accounting-Response containing an Error-Cause attribute with value 406 "Unsupported Extension" as defined in [RFC5176]. A RADIUS/TLS accounting client receiving such an Accounting-Response SHOULD log the error and stop sending Accounting-Request packets.

3. Informative: Design Decisions

This section explains the design decisions that led to the rules defined in the previous section.

3.1. Implications of Dynamic Peer Discovery

One mechanism to discover RADIUS over TLS peers dynamically via DNS is specified in [I-D.ietf-radext-dynamic-discovery]. While this mechanism is still under development and therefore is not a normative dependency of RADIUS/TLS, the use of dynamic discovery has potential future implications that are important to understand.

Readers of this document who are considering the deployment of DNS-based dynamic discovery are thus encouraged to read [I-D.ietf-radext-dynamic-discovery] and follow its future development.

3.2. X.509 Certificate Considerations

(1) If a RADIUS/TLS client is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia) and dynamic discovery is used, the discovery mechanism possibly does not yield sufficient information to identify the consortium uniquely (e.g. DNS discovery). Subsequently, the client may not know by itself which client certificate to use for the TLS handshake. Then it is necessary for the server to signal which consortium it belongs to, and which certificates it expects. If there is no risk of confusing multiple roaming consortia, providing this information in the handshake is not crucial.

(2) If a RADIUS/TLS server is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia), it

will need to select one of its certificates to present to the RADIUS/TLS client. If the client sends the Trusted CA Indication, this hint can make the server select the appropriate certificate and prevent a handshake failure. Omitting this indication makes it impossible to deterministically select the right certificate in this case. If there is no risk of confusing multiple roaming consortia, providing this indication in the handshake is not crucial.

3.3. Ciphersuites and Compression Negotiation Considerations

Not all TLS ciphersuites in [RFC5246] are supported by available TLS tool kits, and licenses may be required in some cases. The existing implementations of RADIUS/TLS use OpenSSL as cryptographic backend, which supports all of the ciphersuites listed in the rules in the normative section.

The TLS ciphersuite TLS_RSA_WITH_3DES_EDE_CBC_SHA is mandatory-to-implement according to [RFC4346] and thus has to be supported by RADIUS/TLS nodes.

The two other ciphersuites in the normative section are widely implemented in TLS toolkits and are considered good practice to implement.

3.4. RADIUS Datagram Considerations

(1) After the TLS session is established, RADIUS packet payloads are exchanged over the encrypted TLS tunnel. In RADIUS/UDP, the packet size can be determined by evaluating the size of the datagram that arrived. Due to the stream nature of TCP and TLS, this does not hold true for RADIUS/TLS packet exchange. Instead, packet boundaries of RADIUS packets that arrive in the stream are calculated by evaluating the packet's Length field. Special care needs to be taken on the packet sender side that the value of the Length field is indeed correct before sending it over the TLS tunnel, because incorrect packet lengths can no longer be detected by a differing datagram boundary. See section 2.6.4 of [I-D.ietf-radext-tcp-transport] for more details.

(2) Within RADIUS/UDP [RFC2865], a shared secret is used for hiding of attributes such as User-Password, as well as in computation of the Response Authenticator. In RADIUS accounting [RFC2866], the shared secret is used in computation of both the Request Authenticator and the Response Authenticator. Since TLS provides integrity protection and encryption sufficient to substitute for RADIUS application-layer security, it is not necessary to configure a RADIUS shared secret. The use of a fixed string for the obsolete shared secret eliminates possible node misconfigurations.

(3) RADIUS/UDP [RFC2865] uses different UDP ports for authentication, accounting and dynamic authorisation changes. RADIUS/TLS allocates a single port for all RADIUS packet types. Nevertheless, in RADIUS/TLS the notion of a client which sends authentication requests and processes replies associated with it's users' sessions and the notion of a server which receives requests, processes them and sends the appropriate replies is to be preserved. The normative rules about acceptable packet types for clients and servers mirror the packet flow behaviour from RADIUS/UDP.

(4) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support reception and processing of the packet types in [RFC5176]. These packet types are listed as to be received in RADIUS/TLS implementations. Note well: it is not required for an implementation to actually process these packet types; it is only required to send the NAK as defined above.

(5) RADIUS/UDP [RFC2865] uses negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support reception and processing of RADIUS Accounting packets. There is no RADIUS datagram to signal an Accounting NAK. Clients may be misconfigured to send Accounting packets to a RADIUS/TLS server which does not wish to process their Accounting packet. To prevent a regression of detectability of this situation, the Accounting-Response + Error-Cause signaling was introduced.

4. Compatibility with other RADIUS transports

Ongoing work in the IETF defines multiple alternative transports to the classic UDP transport model as defined in [RFC2865], namely RADIUS over TCP [I-D.ietf-radext-tcp-transport], RADIUS over Datagram Transport Layer Security (DTLS) [I-D.ietf-radext-dtls] and this present document on RADIUS over TLS.

RADIUS/TLS does not specify any inherent backwards compatibility to RADIUS/UDP or cross compatibility to the other transports, i.e. an implementation which implements RADIUS/TLS only will not be able to receive or send RADIUS packet payloads over other transports. An implementation wishing to be backward or cross compatible (i.e. wishes to serve clients using other transports than RADIUS/TLS) will need to implement these other transports along with the RADIUS/TLS transport and be prepared to send and receive on all implemented transports, which is called a multi-stack implementation.

If a given IP device is able to receive RADIUS payloads on multiple transports, this may or may not be the same instance of software, and it may or may not serve the same purposes. It is not safe to assume

that both ports are interchangeable. In particular, it can not be assumed that state is maintained for the packet payloads between the transports. Two such instances MUST be considered separate RADIUS server entities.

5. Diameter Compatibility

Since RADIUS/TLS is only a new transport profile for RADIUS, compatibility of RADIUS/TLS - Diameter [RFC3588] vs. RADIUS/UDP [RFC2865] - Diameter [RFC3588] is identical. The considerations regarding payload size in [I-D.ietf-radext-tcp-transport] apply.

6. Security Considerations

The computational resources to establish a TLS tunnel are significantly higher than simply sending mostly unencrypted UDP datagrams. Therefore, clients connecting to a RADIUS/TLS node will more easily create high load conditions and a malicious client might create a Denial-of-Service attack more easily.

Some TLS ciphersuites only provide integrity validation of their payload, and provide no encryption. This specification forbids the use of such ciphersuites. Since the RADIUS payload's shared secret is fixed to the well-known term "radsec" (see Section 2.3 (4)), failure to comply with this requirement will expose the entire datagram payload in plain text, including User-Password, to intermediate IP nodes.

By virtue of being based on TCP, there are several generic attack vectors to slow down or prevent the TCP connection from being established; see [RFC4953] for details. If a TCP connection is not up when a packet is to be processed, it gets re-established, so such attacks in general lead only to a minor performance degradation (the time it takes to re-establish the connection). There is one notable exception where an attacker might create a bidding-down attack though: If peer communication between two devices is configured for both RADIUS/TLS (i.e. TLS security over TCP as a transport, shared secret fixed to "radsec") and RADIUS/UDP (i.e. shared secret security with a secret manually configured by the administrator), and where the RADIUS/UDP transport is the failover option if the TLS session cannot be established, a bidding-down attack can occur if an adversary can maliciously close the TCP connection, or prevent it from being established. Situations where clients are configured in such a way are likely to occur during a migration phase from RADIUS/UDP to RADIUS/TLS. By preventing the TLS session setup, the attacker can reduce the security of the packet payload from the selected TLS cipher suite packet encryption to the classic MD5 per-attribute encryption. The situation should be avoided by disabling the weaker

RADIUS/UDP transport as soon as the new RADIUS/TLS connection is established and tested. Disabling can happen at either the RADIUS client or server side:

- o Client side: de-configure the failover setup, leaving RADIUS/TLS as the only communication option
- o Server side: de-configure the RADIUS/UDP client from the list of valid RADIUS clients

RADIUS/TLS provides authentication and encryption between RADIUS peers. In the presence of proxies, the intermediate proxies can still inspect the individual RADIUS packets, i.e. "end-to-end" encryption is not provided. Where intermediate proxies are untrusted, it is desirable to use other RADIUS mechanisms to prevent RADIUS packet payload from inspection by such proxies. One common method to protect passwords is the use of the Extensible Authentication Protocol (EAP) and EAP methods which utilize TLS.

When using certificate fingerprints to identify RADIUS/TLS peers, any two certificates which produce the same hash value (i.e. which have a hash collision) will be considered the same client. It is therefore important to make sure that the hash function used is cryptographically uncompromised so that an attacker is very unlikely to be able to produce a hash collision with a certificate of his choice. While this specification mandates support for SHA-1, a later revision will likely demand support for more contemporary hash functions because as of issuance of this document there are already attacks on SHA-1.

7. IANA Considerations

No new RADIUS attributes or packet codes are defined. IANA is requested to update the already-assigned TCP port number 2083 in the following ways:

- o Reference: list the RFC number of this document as the reference
- o Assignment Notes: add the text "The TCP port 2083 was already previously assigned by IANA for "RadSec", an early implementation of RADIUS/TLS, prior to issuance of this RFC. This early implementation can be configured to be compatible to RADIUS/TLS as specified by the IETF. See RFC (RFC number of this document), Appendix A for details."

8. Notes to the RFC Editor

[I-D.ietf-radext-tcp-transport] is currently in the publication queue because it has a normative reference on this draft; it has no other blocking dependencies. The two drafts should be published as an RFC simultaneously, ideally with consecutive numbers. The references in this draft to [I-D.ietf-radext-tcp-transport] should be changed to references to the corresponding RFC prior to publication.

This section, "Notes to the RFC Editor" should be deleted from the draft prior to publication.

9. Acknowledgements

RADIUS/TLS was first implemented as "RADSec" by Open Systems Consultants, Currumbin Waters, Australia, for their "Radiator" RADIUS server product (see [radsec-whitepaper]).

Funding and input for the development of this Internet Draft was provided by the European Commission co-funded project "GEANT2" [geant2] and further feedback was provided by the TERENA Task Force Mobility [terena].

10. References

10.1. Normative References

- | | |
|-----------|--|
| [RFC2119] | Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997. |
| [RFC2865] | Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000. |
| [RFC2866] | Rigney, C., "RADIUS Accounting", RFC 2866, June 2000. |
| [RFC4279] | Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279, December 2005. |
| [RFC5280] | Cooper, D., Santesson, S., Farrell, S., Boeyen, S., |

Housley, R., and W. Polk,
"Internet X.509 Public Key
Infrastructure Certificate and
Certificate Revocation List
(CRL) Profile", RFC 5280,
May 2008.

[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,
January 2008.

[RFC5246]

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

[RFC5247]

Aboba, B., Simon, D., and P.
Eronen, "Extensible
Authentication Protocol (EAP)
Key Management Framework",
RFC 5247, August 2008.

[RFC6066]

Eastlake, D., "Transport Layer
Security (TLS) Extensions:
Extension Definitions",
RFC 6066, January 2011.

[I-D.ietf-radext-tcp-transport]

DeKok, A., "RADIUS Over TCP", dr
aft-ietf-radext-tcp-transport-09
(work in progress),
October 2010.

10.2. Informative References

[I-D.ietf-radext-dtls]

DeKok, A., "DTLS as a Transport
Layer for RADIUS",
draft-ietf-radext-dtls-01 (work
in progress), October 2010.

[I-D.ietf-radext-dynamic-discovery]

Winter, S. and M. McCauley,
"NAI-based Dynamic Peer
Discovery for RADIUS/TLS and
RADIUS/DTLS", draft-ietf-radext-
dynamic-discovery-03 (work in

progress), July 2011.

- [RFC3539] Aboba, B. and J. Wood,
"Authentication, Authorization
and Accounting (AAA) Transport
Profile", RFC 3539, June 2003.
- [RFC3588] Calhoun, P., Loughney, J.,
Guttman, E., Zorn, G., and J.
Arkko, "Diameter Base Protocol",
RFC 3588, September 2003.
- [RFC4107] Bellovin, S. and R. Housley,
"Guidelines for Cryptographic
Key Management", BCP 107,
RFC 4107, June 2005.
- [RFC4346] Dierks, T. and E. Rescorla, "The
Transport Layer Security (TLS)
Protocol Version 1.1", RFC 4346,
April 2006.
- [RFC4953] Touch, J., "Defending TCP
Against Spoofing Attacks",
RFC 4953, July 2007.
- [RFC6125] Saint-Andre, P. and J. Hodges,
"Representation and Verification
of Domain-Based Application
Service Identity within Internet
Public Key Infrastructure Using
X.509 (PKIX) Certificates in the
Context of Transport Layer
Security (TLS)", RFC 6125,
March 2011.
- [RFC6421] Nelson, D., "Crypto-Agility
Requirements for Remote
Authentication Dial-In User
Service (RADIUS)", RFC 6421,
November 2011.
- [radsec-whitepaper] Open System Consultants, "RadSec
- a secure, reliable RADIUS
Protocol", May 2005, <[http://
www.open.com.au/radiator/
radsec-whitepaper.pdf](http://www.open.com.au/radiator/radsec-whitepaper.pdf)>.

- [MD5-attacks] Black, J., Cochran, M., and T. Highland, "A Study of the MD5 Attacks: Insights and Improvements", October 2006, <<http://www.springerlink.com/content/40867185727r7084/>>.
- [radsecproxy-impl] Venaas, S., "radsecproxy Project Homepage", 2007, <<http://software.uninett.no/radsecproxy/>>.
- [eduroam] Trans-European Research and Education Networking Association, "eduroam Homepage", 2007, <<http://www.eduroam.org/>>.
- [geant2] Delivery of Advanced Network Technology to Europe, "European Commission Information Society and Media: GEANT2", 2008, <<http://www.geant2.net/>>.
- [terena] TERENA, "Trans-European Research and Education Networking Association", 2008, <<http://www.terena.org/>>.

Appendix A. Implementation Overview: Radiator

Radiator implements the RadSec protocol for proxying requests with the <Authby RADSEC> and <ServerRADSEC> clauses in the Radiator configuration file.

The <AuthBy RADSEC> clause defines a RadSec client, and causes Radiator to send RADIUS requests to the configured RadSec server using the RadSec protocol.

The <ServerRADSEC> clause defines a RadSec server, and causes Radiator to listen on the configured port and address(es) for connections from <Authby RADSEC> clients. When an <Authby RADSEC> client connects to a <ServerRADSEC> server, the client sends RADIUS requests through the stream to the server. The server then handles the request in the same way as if the request had been received from a conventional UDP RADIUS client.

Radiator is compliant to RADIUS/TLS if the following options are used:

<AuthBy RADSEC>

- * Protocol tcp
- * UseTLS
- * TLS_CertificateFile
- * Secret radsec

<ServerRADSEC>

- * Protocol tcp
- * UseTLS
- * TLS_RequireClientCert
- * Secret radsec

As of Radiator 3.15, the default shared secret for RadSec connections is configurable and defaults to "mysecret" (without quotes). For compliance with this document, this setting needs to be configured for the shared secret "radsec". The implementation uses TCP keepalive socket options, but does not send Status-Server packets. Once established, TLS connections are kept open throughout the server instance lifetime.

Appendix B. Implementation Overview: radsecproxy

The RADIUS proxy named radsecproxy was written in order to allow use of RadSec in current RADIUS deployments. This is a generic proxy that supports any number and combination of clients and servers, supporting RADIUS over UDP and RadSec. The main idea is that it can be used on the same host as a non-RadSec client or server to ensure RadSec is used on the wire, however as a generic proxy it can be used in other circumstances as well.

The configuration file consists of client and server clauses, where there is one such clause for each client or server. In such a clause one specifies either "type tls" or "type udp" for RadSec or UDP transport. For RadSec the default shared secret "mysecret" (without quotes), the same as Radiator, is used. For compliance with this document, this setting needs to be configured for the shared secret "radsec". A secret may be specified by putting say "secret somesharedsecret" inside a client or server clause.

In order to use TLS for clients and/or servers, one must also specify

where to locate CA certificates, as well as certificate and key for the client or server. This is done in a TLS clause. There may be one or several TLS clauses. A client or server clause may reference a particular TLS clause, or just use a default one. One use for multiple TLS clauses may be to present one certificate to clients and another to servers.

If any RadSec (TLS) clients are configured, the proxy will at startup listen on port 2083, as assigned by IANA for the OSC RadSec implementation. An alternative port may be specified. When a client connects, the client certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests coming from a RadSec client are treated exactly like requests from UDP clients.

The proxy will at startup try to establish a TLS connection to each (if any) of the configured RadSec (TLS) servers. If it fails to connect to a server, it will retry regularly. There is some back-off where it will retry quickly at first, and with longer intervals later. If a connection to a server goes down it will also start retrying regularly. When setting up the TLS connection, the server certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests are sent to a RadSec server just like they would to a UDP server.

The proxy supports Status-Server messages. They are only sent to a server if enabled for that particular server. Status-Server requests are always responded to.

This RadSec implementation has been successfully tested together with Radiator. It is a freely available open-source implementation. For source code and documentation, see [radsecproxy-impl].

Appendix C. Assessment of Crypto-Agility Requirements

The RADIUS Crypto-Agility Requirements [RFC6421] defines numerous classification criteria for protocols that strive to enhance the security of RADIUS. It contains mandatory (M) and recommended (R) criteria which crypto-agile protocols have to fulfill. The authors believe that the following assessment about the crypto-agility properties of RADIUS/TLS are true.

By virtue of being a transport profile using TLS over TCP as a transport protocol, the cryptographically agile properties of TLS are inherited, and RADIUS/TLS subsequently meets the following points:

(M) negotiation of cryptographic algorithms for integrity and auth

- (M) negotiation of cryptographic algorithms for encryption
- (M) replay protection
- (M) define mandatory-to-implement cryptographic algorithms
- (M) generate fresh session keys for use between client and server
- (R) support for Perfect Forward Secrecy in session keys
- (R) support X.509 certificate based operation
- (R) support Pre-Shared keys
- (R) support for confidentiality of the entire packet
- (M/R) support Automated Key Management

The remainder of the requirements is discussed individually below in more detail:

- (M) "avoid security compromise, even in situations where the existing cryptographic algorithms used by RADIUS implementations are shown to be weak enough to provide little or no security" - The existing algorithm, based on MD5, is not of any significance in RADIUS/TLS; its compromise does not compromise the outer transport security.
- (R) mandatory-to-implement algorithms are to be NIST-Acceptable with no deprecation date - The mandatory-to-implement algorithm is TLS_RSA_WITH_3DES_EDE_CBC_SHA. This ciphersuite supports three-key 3DES operation, which is classified as Acceptable with no known deprecation date by NIST.
- (M) demonstrate backward compatibility with RADIUS - There are multiple implementations supporting both RADIUS and RADIUS/TLS, and the translation between them.
- (M) After legacy mechanisms have been compromised, secure algorithms MUST be used, so that backward compatibility is no longer possible - In RADIUS, communication between client and server is always a manual configuration; after a compromise, the legacy client in question can be de-configured by the same manual configuration.
- (M) indicate a willingness to cede change control to the IETF - Change control of this protocol is with the IETF.

(M) be interoperable between implementations based purely on the information in the specification - At least one implementation was created exclusively based on this specification and is interoperable with other RADIUS/TLS implementations.

(M) apply to all packet types - RADIUS/TLS operates on the transport layer, and can carry all packet types.

(R) message data exchanged with Diameter SHOULD NOT be affected - The solution is Diameter-agnostic.

(M) discuss any inherent assumptions - The authors are not aware of any implicit assumptions which would be yet-unarticulated in the draft

(R) provide recommendations for transition - The Security Considerations section contains a transition path.

(R) discuss legacy interoperability and potential for bidding-down attacks - The Security Considerations section contains an corresponding discussion.

Summarizing, it is believed that this specification fulfills all the mandatory and all the recommended requirements for a crypto-agile solution and should thus be considered UNCONDITIONALLY COMPLIANT.

Authors' Addresses

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

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

Mike McCauley
Open Systems Consultants
9 Bulbul Place
Currumbin Waters QLD 4223
AUSTRALIA

Phone: +61 7 5598 7474
Fax: +61 7 5598 7070
EMail: mikem@open.com.au
URI: <http://www.open.com.au>.

Stig Venaas
cisco Systems
Tasman Drive
San Jose, CA 95134
USA

EMail: stig@cisco.com

Klaas Wierenga
Cisco Systems International BV
Haarlerbergweg 13-19
Amsterdam 1101 CH
The Netherlands

Phone: +31 (0)20 3571752
Fax:
EMail: kwiereng@cisco.com
URI: <http://www.cisco.com>.

Network Working Group
INTERNET-DRAFT
Category: Experimental
<draft-ietf-radext-tcp-transport-09.txt>
Expires: April 12, 2011
12 October 2010

A. DeKok
FreeRADIUS

RADIUS Over TCP
draft-ietf-radext-tcp-transport-09

Abstract

The Remote Authentication Dial In User Server (RADIUS) Protocol has until now required the User Datagram Protocol (UDP) as the underlying transport layer. This document defines RADIUS over the Transmission Control Protocol (RADIUS/TCP), in order to address handling issues related to RADIUS over Transport Layer Security (RADIUS/TLS). It permits TCP to be used as a transport protocol for RADIUS only when a transport layer such as TLS or IPsec provides confidentiality and security.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 12, 2011

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info/>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Applicability of Reliable Transport	5
1.2. Terminology	6
1.3. Requirements Language	7
2. Changes to RADIUS	7
2.1. Packet Format	8
2.2. Assigned Ports for RADIUS/TCP	8
2.3. Management Information Base (MIB)	9
2.4. Detecting Live Servers	9
2.5. Congestion Control Issues	10
2.6. TCP Specific Issues	10
2.6.1. Duplicates and Retransmissions	11
2.6.2. Head of Line Blocking	12
2.6.3. Shared Secrets	12
2.6.4. Malformed Packets and Unknown Clients	12
2.6.5. Limitations of the ID Field	13
2.6.6. EAP Sessions	14
2.6.7. TCP Applications are not UDP Applications	15
3. Diameter Considerations	15
4. IANA Considerations	15
5. Security Considerations	15
6. References	16
6.1. Normative References	16
6.2. Informative References	16

1. Introduction

The RADIUS Protocol is defined in [RFC2865] as using the User Datagram Protocol (UDP) for the underlying transport layer. While there are a number of benefits to using UDP as outlined in [RFC2865] Section 2.4, there are also some limitations:

- * Unreliable transport. As a result, systems using RADIUS have to implement application-layer timers and re-transmissions, as described in [RFC5080] Section 2.2.1.

- * Packet fragmentation. [RFC2865] Section 3 permits RADIUS packets up to 4096 octets in length. These packets are larger than the common Internet MTU (576), resulting in fragmentation of the packets at the IP layer when they are proxied over the Internet. Transport of fragmented UDP packets appears to be a poorly tested code path on network devices. Some devices appear to be incapable of transporting fragmented UDP packets, making it difficult to deploy RADIUS in a network where those devices are deployed.

- * Connectionless transport. Neither clients nor servers receive positive statements that a "connection" is down. This information has to be deduced instead from the absence of a reply to a request.

- * Lack of congestion control. Clients can send arbitrary amounts of traffic with little or no feedback. This lack of feedback can result in congestive collapse of the network.

RADIUS has been widely deployed for well over a decade, and continues to be widely deployed. Experience shows that these issues have been minor in some use-cases, and problematic in others. For use-cases such as inter-server proxying, an alternative transport and security model -- RADIUS/TLS, as defined in [RADIUS/TLS]. That document describes the transport implications of running RADIUS/TLS.

The choice of TCP as a transport protocol is largely driven by the desire to improve the security of RADIUS by using RADIUS/TLS. For practical reasons, the transport protocol (TCP) is defined separately from the security mechanism (TLS).

Since "bare" TCP does not provide for confidentiality or enable negotiation of credible ciphersuites, its use is not appropriate for inter-server communications where strong security is required. As a result "bare" TCP transport MUST NOT be used without TLS, IPsec, or other secure upper layer.

"Bare" TCP transport MAY, however, be used when another method such as IPSec [RFC4301] is used to provide additional confidentiality and security. Should experience show that such deployments are useful, this specification could be moved to standards track.

1.1. Applicability of Reliable Transport

The intent of this document is to address transport issues related to RADIUS/TLS [RADIUS/TLS] in inter-server communications scenarios, such as inter-domain communication between proxies. These situations benefit from the confidentiality and ciphersuite negotiation that can be provided by TLS. Since TLS is already widely available within the operating systems used by proxies, implementation barriers are low.

In scenarios where RADIUS proxies exchange a large volume of packets, it is likely that there will be sufficient traffic to enable the congestion window to be widened beyond the minimum value on a long-term basis, enabling ACK piggy-backing. Through use of an application-layer watchdog as described in [RFC3539], it is possible to address the objections to reliable transport described in [RFC2865] Section 2.4 without substantial watchdog traffic, since regular traffic is expected in both directions.

In addition, use of RADIUS/TLS has been found to improve operational performance when used with multi-round trip authentication mechanisms such as EAP over RADIUS [RFC3579]. In such exchanges, it is typical for EAP fragmentation to increase the number of round-trips required. For example, where EAP-TLS authentication [RFC5216] is attempted and both the EAP peer and server utilize certificate chains of 8KB, as many as 15 round-trips can be required if RADIUS packets are restricted to the common Ethernet MTU (1500 octets) for EAP over LAN (EAPoL) use-cases. Fragmentation of RADIUS/UDP packets is generally inadvisable due to lack of fragmentation support within intermediate devices such as filtering routers, firewalls and NATs. However, since RADIUS/UDP implementations typically do not support MTU discovery, fragmentation can occur even when the maximum RADIUS/UDP packet size is restricted to 1500 octets.

These problems disappear if a 4096 application-layer payload can be used alongside RADIUS/TLS. Since most TCP implementations support MTU discovery, the TCP MSS is automatically adjusted to account for the MTU, and the larger congestion window supported by TCP may allow multiple TCP segments to be sent within a single window. Even those few TCP stacks which do not perform path MTU discovery can already support arbitrary payloads.

Where the MTU for EAP packets is large, RADIUS/EAP traffic required for an EAP-TLS authentication with 8KB certificate chains may be

reduced to 7 round-trips or less, resulting in substantially reduced authentication times.

In addition, experience indicates that EAP sessions transported over RADIUS/TLS are less likely to abort unsuccessfully. Historically, RADIUS over UDP implementations have exhibited poor retransmission behavior. Some implementations retransmit packets, others do not, and others send new packets rather than performing retransmission. Some implementations are incapable of detecting EAP retransmissions, and will instead treat the retransmitted packet as an error. As a result, within RADIUS/UDP implementations, retransmissions have a high likelihood of causing an EAP authentication session to fail. For a system with a million logins a day running EAP-TLS mutual authentication with 15 round-trips, and having a packet loss probability of $P=0.01\%$, we expect that 0.3% of connections will experience at least one lost packet. That is, 3,000 user sessions each day will experience authentication failure. This is an unacceptable failure rate for a mass-market network service.

Using a reliable transport method such as TCP means that RADIUS implementations can remove all application-layer retransmissions, and instead rely on the Operating System (OS) kernel's well-tested TCP transport to ensure Path MTU discovery and reliable delivery. Modern TCP implementations also implement anti-spoofing provisions, which is more difficult to do in a UDP application.

In contrast, use of TCP as a transport between a NAS and a RADIUS server is usually a poor fit. As noted in [RFC3539] Section 2.1, for systems originating low numbers of RADIUS request packets, inter-packet spacing is often larger than the packet RTT, meaning that, the congestion window will typically stay below the minimum value on a long-term basis. The result is an increase in packets due to ACKs as compared to UDP, without a corresponding set of benefits. In addition, the lack of substantial traffic implies the need for additional watchdog traffic to confirm reachability.

As a result, the objections to reliable transport indicated in [RFC2865] Section 2.4 continue to apply to NAS-RADIUS server communications and UDP SHOULD continue to be used as the transport protocol in this scenario. In addition, it is recommended that implementations of "RADIUS Dynamic Authorization Extensions" [RFC5176] SHOULD continue to utilize UDP transport, since the volume of dynamic authorization traffic is usually expected to be small.

1.2. Terminology

This document uses the following terms:

RADIUS client

A device that provides an access service for a user to a network.
Also referred to as a Network Access Server, or NAS.

RADIUS server

A device that provides one or more of authentication,
authorization, and/or accounting (AAA) services to a NAS.

RADIUS proxy

A RADIUS proxy acts as a RADIUS server to the NAS, and a RADIUS
client to the RADIUS server.

RADIUS request packet

A packet originated by a RADIUS client to a RADIUS server. e.g.
Access-Request, Accounting-Request, CoA-Request, or Disconnect-
Request.

RADIUS response packet

A packet sent by a RADIUS server to a RADIUS client, in response to
a RADIUS request packet. e.g. Access-Accept, Access-Reject,
Access-Challenge, Accounting-Response, CoA-ACK, etc.

RADIUS/UDP

RADIUS over UDP, as defined in [RFC2865].

RADIUS/TCP

RADIUS over TCP, as defined in this document.

RADIUS/TLS

RADIUS over TLS, as defined in [RADIUS/TLS].

1.3. Requirements Language

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

2. Changes to RADIUS

RADIUS/TCP involves sending RADIUS application messages over a TCP
connection. In the sections that follow, we discuss the implications
for the RADIUS packet format (Section 2.1), port usage (Section 2.2),
RADIUS MIBs (Section 2.3) and RADIUS proxies (Section 2.5). TCP-
specific issues are discussed in Section 2.6.

2.1. Packet Format

The RADIUS packet format is unchanged from [RFC2865], [RFC2866], and [RFC5176]. Specifically, all of the following portions of RADIUS MUST be unchanged when using RADIUS/TCP:

- * Packet format
- * Permitted codes
- * Request Authenticator calculation
- * Response Authenticator calculation
- * Minimum packet length
- * Maximum packet length
- * Attribute format
- * Vendor-Specific Attribute (VSA) format
- * Permitted data types
- * Calculations of dynamic attributes such as CHAP-Challenge, or Message-Authenticator.
- * Calculation of "encrypted" attributes such as Tunnel-Password.

The use of TLS transport does not change the calculation of security-related fields (such as the Response-Authenticator) in RADIUS [RFC2865] or RADIUS Dynamic Authorization [RFC5176]. Calculation of attributes such as User-Password [RFC2865] or Message-Authenticator [RFC3579] also does not change.

Clients and servers MUST be able to store and manage shared secrets based on the key described above, of (IP address, port, transport protocol).

The changes to RADIUS implementations required to implement this specification are largely limited to the portions that send and receive packets on the network.

2.2. Assigned Ports for RADIUS/TCP

IANA has already assigned TCP ports for RADIUS transport, as outlined below:

- * radius 1812/tcp
- * radius-acct 1813/tcp
- * radius-dynauth 3799/tcp

Since these ports are unused by existing RADIUS implementations, the assigned values MUST be used as the default ports for RADIUS over TCP.

The early deployment of RADIUS was done using UDP port number 1645, which conflicts with the "datametrics" service. Implementations

using RADIUS/TCP MUST NOT use TCP ports 1645 or 1646 as the default ports for this specification.

The "radsec" port (2083/tcp) SHOULD be used as the default port for RADIUS/TLS. The "radius" port (1812/tcp) SHOULD NOT be used for RADIUS/TLS.

2.3. Management Information Base (MIB)

The MIB Module definitions in [RFC4668], [RFC4669], [RFC4670], [RFC4671], [RFC4672], and [RFC4673] are intended to be used for RADIUS over UDP. As such, they do not support RADIUS/TCP, and will need to be updated in the future. Implementations of RADIUS/TCP SHOULD NOT re-use these MIB Modules to perform statistics counting for RADIUS/TCP connections.

2.4. Detecting Live Servers

As RADIUS is a "hop by hop" protocol, a RADIUS proxy shields the client from any information about downstream servers. While the client may be able to deduce the operational state of the local server (i.e. proxy), it cannot make any determination about the operational state of the downstream servers.

Within RADIUS as defined in [RFC2865], proxies typically only forward traffic between the NAS and RADIUS server, and do not generate their own responses. As a result, when a NAS does not receive a response to a request, this could be the result of packet loss between the NAS and proxy, a problem on the proxy, loss between the RADIUS proxy and server, or a problem with the server.

When UDP is used as a transport protocol, the absence of a reply can cause a client to deduce (incorrectly) that the proxy is unavailable. The client could then fail over to another server, or conclude that no "live" servers are available (OKAY state in [RFC3539] Appendix A). This situation is made even worse when requests are sent through a proxy to multiple destinations. Failures in one destination may result in service outages for other destinations, if the client erroneously believes that the proxy is unresponsive.

For RADIUS/TLS, it is RECOMMENDED that implementations utilize the existence of a TCP connection along with the application layer watchdog defined in [RFC3539] Section 3.4 to determine that the server is "live".

RADIUS clients using RADIUS/TCP MUST mark a connection DOWN if the network stack indicates that the connection is no longer active. If the network stack indicates that connection is still active, Clients

MUST NOT decide that it is down until the application layer watchdog algorithm has marked it DOWN ([RFC3539] Appendix A). RADIUS clients using RADIUS/TCP MUST NOT decide that a RADIUS server is unresponsive until all TCP connections to it have been marked DOWN.

The above requirements do not forbid the practice of a client proactively closing connections, or marking a server as DOWN due to an administrative decision.

2.5. Congestion Control Issues

Additional issues with RADIUS proxies involve transport protocol changes where the proxy receives packets on one transport protocol, and forwards them on a different transport protocol. There are several situations in which the law of "conservation of packets" could be violated on an end-to-end basis (e.g. where more packets could enter the system than could leave it on a short-term basis):

- * Where TCP is used between proxies, it is possible that the bandwidth consumed by incoming UDP packets destined to a given upstream server could exceed the sending rate of a single TCP connection to that server, based on the window size/RTT estimate.

- * It is possible for the incoming rate of TCP packets destined to a given realm to exceed the UDP throughput achievable using the transport guidelines established in [RFC5080]. This could happen, for example, where the TCP window between proxies has opened, but packet loss is being experienced on the UDP leg, so that the effective congestion window on the UDP side is 1.

Intrinsically, proxy systems operate with multiple control loops instead of one end-to-end loop, and so are less stable. This is true even for TCP-TCP proxies. As discussed in [RFC3539], the only way to achieve stability equivalent to a single TCP connection is to mimic the end-to-end behavior of a single TCP connection. This typically is not achievable with an application-layer RADIUS implementation, regardless of transport.

2.6. TCP Specific Issues

The guidelines defined in [RFC3539] for implementing a AAA protocol over reliable transport are applicable to RADIUS/TLS.

The Application Layer Watchdog defined in [RFC3539] Section 3.4 MUST be used. The Status-Server packet [RFC5997] MUST be used as the application layer watchdog message. Implementations MUST reserve one RADIUS ID per connection for the application layer watchdog message. This restriction is described further below in Section 2.6.4.

RADIUS/TLS Implementations MUST support receiving RADIUS packets over both UDP and TLS transports originating from the same endpoint. RADIUS packets received over UDP MUST be replied to over UDP; RADIUS packets received over TLS MUST be replied to over TLS. That is, RADIUS clients and servers MUST be treated as unique based on a key of the three-tuple (IP address, port, transport protocol). Implementations MUST permit different shared secrets to be used for UDP and TCP connections to the same destination IP address and numerical port.

This requirement does not forbid the traditional practice of using primary and secondary servers in a fail-over relationship. Instead, it requires that two services sharing an IP address and numerical port, but differing in transport protocol, MUST be treated as independent services for the purpose of fail-over, load-balancing, etc.

Whenever the underlying network stack permits the use of TCP keepalive socket options, their use is RECOMMENDED.

2.6.1. Duplicates and Retransmissions

As TCP is a reliable transport, implementations MUST NOT retransmit RADIUS request packets over a given TCP connection. Similarly, if there is no response to a RADIUS packet over one TCP connection, implementations MUST NOT retransmit that packet over a different TCP connection to the same destination IP address and port, while the first connection is in the OKAY state ([RFC3539] Appendix A).

However, if the TCP connection is broken or closed, retransmissions over new connections are permissible. RADIUS request packets that have not yet received a response MAY be transmitted by a RADIUS client over a new TCP connection. As this procedure involves using a new source port, the ID of the packet MAY change. If the ID changes, any security attributes such as Message-Authenticator MUST be recalculated.

If a TCP connection is broken or closed, any cached RADIUS response packets ([RFC5080] Section 2.2.2) associated with that connection MUST be discarded. A RADIUS server SHOULD stop processing of any requests associated with that TCP connection. No response to these requests can be sent over the TCP connection, so any further processing is pointless. This requirement applies not only to RADIUS servers, but also to proxies. When a client's connection to a proxy server is closed, there may be responses from a home server that were supposed to be sent by the proxy back over that connection to the client. Since the client connection is closed, those responses from the home server to the proxy server SHOULD be silently discarded by

the proxy.

Despite the above discussion, RADIUS servers SHOULD still perform duplicate detection on received packets, as described in [RFC5080] Section 2.2.2. This detection can prevent duplicate processing of packets from non-conformant clients.

RADIUS packets SHOULD NOT be re-transmitted to the same destination IP and numerical port, but over a different transport protocol. There is no guarantee in RADIUS that the two ports are in any way related. This requirement does not, however, forbid the practice of putting multiple servers into a fail-over or load-balancing pool. In that situation, RADIUS request MAY be retransmitted to another server that is known to be part of the same pool.

2.6.2. Head of Line Blocking

When using UDP as a transport for RADIUS, there is no ordering of packets. If a packet sent by a client is lost, that loss has no effect on subsequent packets sent by that client.

Unlike UDP, TCP is subject to issues related to Head of Line (HoL) blocking. This occurs when when a TCP segment is lost and a subsequent TCP segment arrives out of order. While the RADIUS server can process RADIUS packets out of order, the semantics of TCP makes this impossible. This limitation can lower the maximum packet processing rate of RADIUS/TCP.

2.6.3. Shared Secrets

The use of TLS transport does not change the calculation of security-related fields (such as the Response-Authenticator) in RADIUS [RFC2865] or RADIUS Dynamic Authorization [RFC5176]. Calculation of attributes such as User-Password [RFC2865] or Message-Authenticator [RFC3579] also does not change.

Clients and servers MUST be able to store and manage shared secrets based on the key described above, of (IP address, port, transport protocol).

2.6.4. Malformed Packets and Unknown Clients

The RADIUS specifications ([RFC2865], etc.) say that an implementation should "silently discard" a packet in a number of circumstances. This action has no further consequences for UDP transport, as the "next" packet is completely independent of the previous one.

When TCP is used as a transport, decoding the "next" packet on a connection depends on the proper decoding of the previous packet. As a result, the behavior with respect to discarded packets has to change.

Implementations of this specification SHOULD treat the "silently discard" texts referenced above as "silently discard and close the connection." That is, the TCP connection MUST be closed if any of the following circumstances are seen:

- * Connection from an unknown client
- * Packet where the RADIUS "length" field is less than the minimum RADIUS packet length
- * Packet where the RADIUS "length" field is more than the maximum RADIUS packet length
- * Packet that has an Attribute "length" field has value of zero or one (0 or 1).
- * Packet where the attributes do not exactly fill the packet
- * Packet where the Request Authenticator fails validation (where validation is required).
- * Packet where the Response Authenticator fails validation (where validation is required).
- * Packet where the Message-Authenticator attribute fails validation (when it occurs in a packet).

After applying the above rules, there are still two situations where the previous specifications allow a packet to be "silently discarded" on reception:

- * Packets with an invalid code field
- * Response packets that do not match any outstanding request

In these situations, the TCP connections MAY remain open, or MAY be closed, as an implementation choice. However, the invalid packet MUST be silently discarded.

These requirements reduce the possibility for a misbehaving client or server to wreak havoc on the network.

2.6.5. Limitations of the ID Field

The RADIUS ID field is one octet in size. As a result, any one TCP connection can have only 256 "in flight" RADIUS packets at a time. If more than 256 simultaneous "in flight" packets are required, additional TCP connections will need to be opened. This limitation is also noted in [RFC3539] Section 2.4.

An additional limit is the requirement to send a Status-Server packet

over the same TCP connection as is used for normal requests. As noted in [RFC5997], the response to a Status-Server packet is either an Access-Accept or an Accounting-Response. If all IDs were allocated to normal requests, then there would be no free ID to use for the Status-Server packet, and it could not be sent over the connection.

Implementations SHOULD reserve ID zero (0) on each TCP connection for Status-Server packets. This value was picked arbitrarily, as there is no reason to choose any one value over another for this use.

Implementors may be tempted to extend RADIUS to permit more than 256 outstanding packets on one connection. However, doing so is a violation of a fundamental part of the protocol and MUST NOT be done. Making that extension here is outside of the scope of this specification.

2.6.6. EAP Sessions

When RADIUS clients send EAP requests using RADIUS/TCP, they SHOULD choose the same TCP connection for all packets related to one EAP session. This practice ensures that EAP packets are transmitted in order, and that problems with any one TCP connection do not affect the minimum number of EAP sessions.

A simple method that may work in many situations is to hash the contents of the Calling-Station-Id attribute, which normally contains the MAC address. The output of that hash can be used to select a particular TCP connection.

However, EAP packets for one EAP session can still be transported from client to server over multiple paths. Therefore, when a server receives a RADIUS request containing an EAP request, it MUST be processed without considering the transport protocol. For TCP transport, it MUST be processed without considering the source port. The algorithm suggested in [RFC5080] Section 2.1.1 SHOULD be used to track EAP sessions, as it is independent of source port and transport protocol.

The retransmission requirements of Section 2.6.1, above, MUST be applied to RADIUS encapsulated EAP packets. That is, EAP retransmissions MUST NOT result in retransmissions of RADIUS packets over a particular TCP connection. EAP retransmissions MAY result in retransmission of RADIUS packets over a different TCP connection, but only when the previous TCP connection is marked DOWN.

2.6.7. TCP Applications are not UDP Applications

Implementors should be aware that programming a robust TCP application can be very different from programming a robust UDP application. It is RECOMMENDED that implementors of this specification familiarize themselves with TCP application programming concepts.

Clients and servers SHOULD implement configurable connection limits. Clients and servers SHOULD implement configurable rate limiting on new connections. Allowing an unbounded number or rate of TCP connections may result in resource exhaustion.

Further discussion of implementation issues is outside of the scope of this document.

3. Diameter Considerations

This document defines TCP as a transport layer for RADIUS. It defines no new RADIUS attributes or codes. The only interaction with Diameter is in a RADIUS to Diameter, or in a Diameter to RADIUS gateway. The RADIUS side of such a gateway MAY implement RADIUS/TCP, but this change has no effect on Diameter.

4. IANA Considerations

This document requires no action by IANA.

5. Security Considerations

As the RADIUS packet format, signing, and client verification are unchanged from prior specifications, all of the security issues outlined in previous specifications for RADIUS/UDP are also applicable here.

As noted above, clients and servers SHOULD support configurable connection limits. Allowing an unlimited number of connections may result in resource exhaustion.

Implementors should consult [RADIUS/TLS] for issues related the security of RADIUS/TLS, and [RFC5246] for issues related to the security of the TLS protocol.

Since "bare" TCP does not provide for confidentiality or enable negotiation of credible ciphersuites, its use is not appropriate for inter-server communications where strong security is required. As a result "bare" TCP transport MUST NOT be used without TLS, IPsec, or other secure upper layer.

There are no (at this time) other known security issues for RADIUS over TCP transport.

6. References

6.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.
- [RFC3539] Aboba, B. et al., "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.
- [RADIUS/TLS]
Winter, S. et. al., "TLS encryption for RADIUS over TCP (RadSec)", draft-ietf-radext-radsec-07.txt, July 2010 (work in progress).
- [RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August, 2010.

6.2. Informative References

- [RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.
- [RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.
- [RFC4301] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 4301, December, 2005.
- [RFC4668] Nelson, D, "RADIUS Authentication Client MIB for IPv6", RFC 4668, August 2006.
- [RFC4669] Nelson, D, "RADIUS Authentication Server MIB for IPv6", RFC 4669, August 2006.
- [RFC4670] Nelson, D, "RADIUS Accounting Client MIB for IPv6", RFC 4670, August 2006.
- [RFC4671] Nelson, D, "RADIUS Accounting Server MIB for IPv6", RFC 4671, August 2006.

- [RFC4672] Nelson, D, "RADIUS Dynamic Authorization Client MIB", RFC 4672, August 2006.
- [RFC4673] Nelson, D, "RADIUS Dynamic Authorization Server MIB", RFC 4673, August 2006.
- [RFC5080] Nelson, D. and DeKok, A, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.
- [RFC5176] Chiba, M. et al., "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.
- [RFC5216] Simon, D., etc al., "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.
- [RFC5246] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Acknowledgments

None at this time.

Authors' Addresses

Alan DeKok
The FreeRADIUS Server Project
<http://freeradius.org/>

Email: aland@freeradius.org

Open issues

Open issues relating to this document are tracked on the following web site:

<http://www.drizzle.com/~aboba/RADEXT/>

