Internet Engineering Task Force (IETF)
Request for Comments: 6614
Category: Experimental
ISSN: 2070-1721

S. Winter
RESTENA
M. McCauley
OSC
S. Venaas
K. Wierenga
Cisco

May 2012

Transport Layer Security (TLS) Encryption for RADIUS

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 document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc6614.

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. Acknowledgements 15
9. References 15
9.1. Normative References 15
9.2. 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 authorization protocol. The supplementary RADIUS Accounting specification [RFC2866] 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 worldwide roaming environment that uses RADIUS over TLS to secure communication is "eduroam", see [eduroam].

There are multiple known attacks on the MD5 algorithm that 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 the control of different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times, as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RADIUS over TLS 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 that initiates a new connection.

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

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

1.3. 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, point 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 ciphersuites, 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 will prevail or if both will find their place in real-life deployments. The authors can imagine pre-shared keys (PSK) to be popular in small-scale deployments (Small Office, Home Office (SOHO) or isolated enterprise deployments) where scalability is not an issue and the deployment of a Certification Authority (CA) is considered too much of a hassle; however, the authors 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 on a large scale is by observing the uptake, usability, and operational behavior of the protocol in large-scale, real-life deployments.

An example for a worldwide 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 authorization changes. The source port is arbitrary. See Section 3.4 for considerations regarding the 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 [RFC6613]. 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 the configuration of 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 the configuration of a set of additional properties of the certificate to check for a peer's authorization to communicate (e.g., a set of allowed values in subjectAltName:URI or a set of allowed 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 renegotiate the TLS session to reassess the connecting peer's continued authorization.

* TLS with X.509 certificates using certificate fingerprints (this model is optional to implement): Implementations SHOULD allow the configuration of 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 that 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. For example, if the Issuer is not in a trusted set of Issuers, the server may decline to perform RADIUS transactions with this client.

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

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

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

- Originating IP address
- TLS Identifier

2.5. RADIUS Datagrams

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

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

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

and they receive

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

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

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

and they receive

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

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

- When an unwanted packet of type 'CoA-Request' or 'Disconnect-Request' is received, a RADIUS/TLS server needs to respond with a 'CoA-NAK' or 'Disconnect-NAK', respectively. The NAK SHOULD contain an attribute Error-Cause with the value 406 ("Unsupported Extension"); see [RFC5176] for details.

- When an unwanted packet of type 'Accounting-Request' is received, the RADIUS/TLS server SHOULD reply with an Accounting-Response containing an Error-Cause attribute with value 406 "Unsupported Extension" as defined in [RFC5176]. A RADIUS/TLS accounting client receiving such an Accounting-Response SHOULD log the error and stop sending Accounting-Request packets.

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 [DYNAMIC]. 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 [DYNAMIC] and follow its future development.

3.2. X.509 Certificate Considerations

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

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

3.3. 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 a 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]; thus, it has to be supported by RADIUS/TLS nodes.

The two other ciphersuites in the normative section are widely implemented in TLS tool kits 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 [RFC6613] for more details.

(2) Within RADIUS/UDP [RFC2865], a shared secret is used for hiding attributes such as User-Password, as well as in computation of the Response Authenticator. In RADIUS accounting [RFC2866], the shared secret is used in computation of both the Request Authenticator and the Response Authenticator. Since TLS provides integrity protection and encryption sufficient to

substitute for RADIUS application-layer security, it is not necessary to configure a RADIUS shared secret. The use of a fixed string for the obsolete shared secret eliminates possible node misconfigurations.

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

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

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

4. Compatibility with Other RADIUS Transports

The IETF defines multiple alternative transports to the classic UDP transport model as defined in [RFC2865], namely RADIUS over TCP [RFC6613] and the present document on RADIUS over TLS. The IETF also proposed RADIUS over Datagram Transport Layer Security (DTLS) [RADEXT-DTLS].

RADIUS/TLS does not specify any inherent backward compatibility to RADIUS/UDP or cross compatibility to the other transports, i.e., an implementation that utilizes RADIUS/TLS only will not be able to receive or send RADIUS packet payloads over other transports. An implementation wishing to be backward or cross compatible (i.e., wishes to serve clients using other transports than RADIUS/TLS) will

need to implement these other transports along with the RADIUS/TLS transport and be prepared to send and receive on all implemented transports, which is called a "multi-stack implementation".

If a given IP device is able to receive RADIUS payloads on multiple transports, this may or may not be the same instance of software, and it may or may not serve the same purposes. It is not safe to assume that both ports are interchangeable. In particular, it cannot be assumed that state is maintained for the packet payloads between the transports. Two such instances MUST be considered separate RADIUS server entities.

5. Diameter Compatibility

Since RADIUS/TLS is only a new transport profile for RADIUS, the compatibility of RADIUS/TLS - Diameter [RFC3588] and RADIUS/UDP [RFC2865] - Diameter [RFC3588] is identical. The considerations regarding payload size in [RFC6613] 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 plaintext, including User-Password, to intermediate IP nodes.

By virtue of being based on TCP, there are several generic attack vectors to slow down or prevent the TCP connection from being established; see [RFC4953] for details. If a TCP connection is not up when a packet is to be processed, it gets re-established, so such attacks in general lead only to a minor performance degradation (the time it takes to re-establish the connection). There is one notable exception where an attacker might create a bidding-down attack though. If peer communication between two devices is configured for both RADIUS/TLS (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 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 ciphersuite 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:

- Client side: de-configure the failover setup, leaving RADIUS/TLS as the only communication option

- 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 that utilize TLS.

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

7. IANA Considerations

No new RADIUS attributes or packet codes are defined. IANA has updated the already assigned TCP port number 2083 to reflect the following:

- Reference: [RFC6614]

- Assignment Notes: 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 6614, Appendix A for details.

8. 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 document was provided by the European Commission co-funded project "GEANT2" [geant2] and further feedback was provided by the TERENA Task Force on Mobility and Network Middleware [terena].

9. References

9.1. 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.

[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.

[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.

[RFC6066] Eastlake, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, January 2011.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, May 2012.

9.2. Informative References

[DYNAMIC] Winter, S. and M. McCauley, "NAI-based Dynamic Peer Discovery for RADIUS/TLS and RADIUS/DTLS", Work in Progress, July 2011.

[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/40867l85727r7084/>.

[RADEXT-DTLS] DeKok, A., "DTLS as a Transport Layer for RADIUS", Work in Progress, October 2010.

[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.

[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/>.

[radsec-whitepaper] Open System Consultants, "RadSec - a secure, reliable RADIUS Protocol", May 2005, <http://www.open.com.au/radiator/radsec-whitepaper.pdf>.

[radsecproxy-impl] Venaas, S., "radsecproxy Project Homepage", 2007, <http://software.uninett.no/radsecproxy/>.

[terena] Trans-European Research and Education Networking Association (TERENA), "Task Force on Mobility and Network Middleware", 2008, <http://www.terena.org/activities/tf-mobility/>.

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 TLS or UDP transport. Versions prior to 1.6 used "mysecret" as a default shared secret for RADIUS/TLS; version 1.6 and onwards uses "radsec". For backwards compatibility with older versions, the secret can be changed (which makes the configuration not compliant with this specification).

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 Fully Qualified Domain Name (FQDN) or IP address matches what is in the certificate. Requests coming from a RadSec client are treated exactly like requests from UDP clients.

At startup, the proxy will 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 be 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 document [RFC6421] defines numerous classification criteria for protocols that strive to enhance the security of RADIUS. It contains mandatory (M) and recommended (R) criteria that 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 utilized by RADIUS implementations are shown to be weak enough to provide little or no security" [RFC6421]. 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 that would be yet-unarticulated in the document.

(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 a 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
EMail: klaas@cisco.com
URI: http://www.cisco.com