RADIUS Extensions Working Group S. Winter
Internet-Draft RESTENA
Intended status: Experimental M. McCauley
Expires: January 9, 2012 OSC
S. Venaas
UNINETT
K. Wierenga
Cisco
July 8, 2011

TLS encryption for RADIUS
draft-ietf-radext-radsec-09

Abstract

This document specifies security on the transport layer (TLS) for the RADIUS protocol when transmitted over TCP. This enables dynamic trust relationships between RADIUS servers.

Table of Contents

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 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 on the transport layer. 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 dynamic establishment of connections to peers that are not previously configured, and thus makes it possible to avoid aggregation-only RADIUS proxies and reduce the number of middleboxes which can eavesdrop on traffic. One mechanism to discover RADIUS over TLS peers with DNS is specified in [I-D.winter-dynamic-discovery].

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]

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 3.3 (4) and (5) for considerations regarding separation of authentication, accounting and dynauth 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. The following restrictions apply: 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.

* 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.

* 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 3.2 (1)).

* In addition, RADIUS/TLS implementations MUST support negotiation of the mandatory-to-implement ciphersuites required by the versions of TLS that they support.

* RADIUS/TLS nodes MUST NOT negotiate ciphersuites with NULL encryption (e.g. [RFC4785]).

3. If TLS is used in an X.509 certificate based operation mode, the following list of certificate validation options applies:

   * Implementations MUST allow to configure a list of acceptable Certification Authorities for incoming connections.

   * Certificate validation MUST include the verification rules as per [RFC5280].

   * Implementations SHOULD indicate their acceptable Certification Authorities as per section 7.4.4 (server side) and x.y.z ["Trusted CA Indication"] (client side) of [RFC5246] (see Section 3.1)

   * Implementations SHOULD allow to configure a list of acceptable certificates, identified via certificate fingerprint. When a fingerprint configured, the fingerprint is prepended with an ASCII label identifying the hash function followed by a colon. Implementations MUST support SHA-1 as the hash algorithm and use the ASCII label "sha-1" to identify the SHA-1 algorithm. The length of a SHA-1 hash is 20 bytes and the length of the corresponding fingerprint string is 65 characters. An example certificate fingerprint is: sha-

1:E1:2D:53:2B:7C:6B:8A:29:A2:76:C8:64:36:0B:08:4B:7A:F1:9E:9D

* 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.

* Implementations SHOULD allow to configure a set of acceptable values for subjectAltName:URI.

4. start exchanging RADIUS datagrams. Note Section 3.3 (1)). The shared secret to compute the (obsolete) MD5 integrity checks and attribute encryption MUST be "radsec" (see Section 3.3 (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 with PKI infrastructure, a client is uniquely identified by the serial number of the tuple (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 PKI 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:

- 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, 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 3.3 (3) and (4)). E.g. they send

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

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 ...

3. Informative: Design Decisions

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

3.1. 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) If dynamic peer discovery as per [I-D.winter-dynamic-discovery] is used, peer authentication alone is not sufficient; the peer must also be authorised to perform user authentications. In these cases, the trust fabric cannot depend on peer authentication methods like DNSSEC to identify RADIUS/TLS nodes. The nodes also need to be properly authorised. Typically, this can be achieved by adding appropriate authorisation fields into a X.509 certificate. Such fields include SRV authority [RFC4985], subjectAltNames, or a defined list of certificate fingerprints. Operators of a RADIUS/TLS infrastructure should define their own authorisation trust model and apply this model to the certificates. The checks enumerated in Section 2.3 provide sufficient flexibility for the implementation of authorisation trust models.

3.2. 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 [RFC5246] 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.

3.3. 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 sufficient that upon receiving such a packet, an unconditional NAK is sent back to indicate that the action is not supported.

(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. The server will need to silently drop the packet. The client will need to deduce from the absence of replies that it is misconfigured; no negative ICMP response will reveal this.

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 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.

As a consequence, the selection of transports to communicate from a client to a server is a manual administrative action. An automatic fallback to RADIUS/UDP is NOT RECOMMENDED, as it may lead to down-bidding attacks on the peer communication.

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.

In the case of dynamic peer discovery as per [I-D.winter-dynamic-discovery], a RADIUS/TLS node needs to be able to accept connections from a large, not previously known, group of hosts, possibly the whole internet. In this case, the server's RADIUS/TLS port can not be protected from unauthorised connection attempts with measures on the network layer, i.e. access lists and firewalls. This opens more attack vectors for Distributed Denial of Service attacks, just like any other service that is supposed to serve arbitrary clients (like for example web servers).

In the case of dynamic peer discovery as per [I-D.winter-dynamic-discovery], X.509 certificates are the only proof of authorisation for a connecting RADIUS/TLS nodes. Special care needs to be taken that certificates get verified properly according to the chosen trust model (particularly: consulting CRLs, checking critical extensions, checking subjectAltNames etc.) to prevent unauthorised connections.

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 and well-known, failure to comply with this requirement will expose the entire datagram payload in plain text, including User-Password, to intermediate IP nodes.

If peer communication between two devices is configured for both RADIUS/TLS and RADIUS/UDP, a failover from TLS security to classic RADIUS security opens the way for a down-bidding attack if an adversary can maliciously close the TCP connection, or prevent it from being established. In this case, security of the packet payload is reduced from the selected TLS cipher suite packet encryption to the classic MD5 per-attribute encryption. Such an attack can be mitigated by delisting the RADIUS/UDP client from the server configuration after successfully migrating that client to RADIUS/TLS.

The RADIUS/TLS transport 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 EAP methods which utilize TLS.

7. IANA Considerations

This document has no actions for IANA. The TCP port 2083 was already previously assigned by IANA for RadSec, an early implementation of RADIUS/TLS. No new RADIUS attributes or packet codes are defined.

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 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].

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.

[RFC4785] Blumenthal, U. and P. Goel, "Pre-Shared Key (PSK) Ciphersuites with NULL Encryption for Transport Layer Security (TLS)", RFC 4785, January 2007.

[RFC4985] Santesson, S., "Internet X.509 Public Key Infrastructure Subject Alternative Name for Expression of

Service Name", RFC 4985, August 2007.

[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.

[I-D.ietf-radext-tcp-transport] DeKok, A., "RADIUS Over TCP", draft-ietf-radext-tcp-transport-09 (work in progress), October 2010.

9.2. 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.winter-dynamic-discovery] Winter, S. and M. McCauley, "NAI-based Dynamic Peer Discovery for RADIUS over TLS and DTLS", draft-winter-dynamic-discovery-00 (work in progress), February 2009.

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[RFC4346] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[radsec-whitepaper] Open System Consultants, "RadSec - a secure, reliable RADIUS Protocol", May 2005, <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/40867l85727r7084/>.

[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 version 2 of RadSec 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 (link to RFC once issued here) 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 operating on the transport layer with TLS, 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 alogrithms 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 alogrithms 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
UNINETT
Abels gate 5 - Teknobyen
Trondheim 7465
NORWAY

Phone: +47 73 55 79 00
Fax: +47 73 55 79 01
EMail: stig.venaas@uninett.no
URI: http://www.uninett.no.

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.