Internet Engineering Task Force (IETF)
Request for Comments: 6613
Category: Experimental
ISSN: 2070-1721

A. DeKok
FreeRADIUS

May 2012

RADIUS over TCP

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 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/rfc6613.

Table of Contents

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 retransmissions, 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, is defined in [RFC6614]. 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 another secure upper layer.

However, "bare" TCP transport MAY 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 the Standards Track.

1.1. Applicability of Reliable Transport

The intent of this document is to address transport issues related to RADIUS/TLS [RFC6614] 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 piggybacking. 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 the Extensible Authentication Protocol (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 8 KB, 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-octet application-layer payload can be used alongside RADIUS/TLS. Since most TCP implementations support MTU discovery, the TCP Maximum Segment Size (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 that 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 8-KB 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 (see Section 1.2) 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 Network Access Server (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 Round-Trip Time (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. For example, 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. For example, Access-Accept, Access-Reject, Access-Challenge, Accounting-Response, or CoA-ACK.

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 [RFC6614].

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 in Section 2.6, 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 reuse 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 they 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 the 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 they 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 in Section 2.6.4.

RADIUS/TLS implementations MUST support receiving RADIUS packets over both UDP and TCP transports originating from the same endpoint. RADIUS packets received over UDP MUST be replied to over UDP; RADIUS packets received over TCP MUST be replied to over TCP. 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 failover 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 failover, 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 the 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 retransmitted 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 failover 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 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, at the start of this section (i.e., IP address, port, transport protocol).

2.6.4. Malformed Packets and Unknown Clients

The RADIUS specifications ([RFC2865], and many others) 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" upon receipt:

* 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 they 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 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 Media Access Control (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 the 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 limits on connection lifetime and idle timeouts. 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. 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 [RFC6614] for issues related to 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 another secure upper layer.

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

5. References

5.1. 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. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

[RFC5997] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", RFC 5997, August 2010.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, May 2012.

5.2. 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 K. Seo, "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] De Cnodder, S., Jonnala, N., and M. Chiba, "RADIUS Dynamic Authorization Client MIB", RFC 4672, September 2006.

[RFC4673] De Cnodder, S., Jonnala, N., and M. Chiba, "RADIUS Dynamic Authorization Server MIB", RFC 4673, September 2006.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, December 2007.

[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.

[RFC5216] Simon, D., Aboba, B., and R. Hurst, "The EAP-TLS Authentication Protocol", RFC 5216, March 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

Author’s Address

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

EMail: aland@freeradius.org