Network Working Group
INTERNET-DRAFT
Category: Informational
<draft-ietf-radext-dtls-02.txt>
Expires: May 16, 2013
16 July 2012

Alan DeKok
FreeRADIUS

DTLS as a Transport Layer for RADIUS
draft-ietf-radext-dtls-02

Abstract

The RADIUS protocol [RFC2865] has limited support for authentication and encryption of RADIUS packets. The protocol transports data "in the clear", although some parts of the packets can have "obfuscated" content. Packets may be replayed verbatim by an attacker, and client-server authentication is based on fixed shared secrets. This document specifies how the Datagram Transport Layer Security (DTLS) protocol may be used as a fix for these problems. It also describes how implementations of this proposal can co-exist with current RADIUS systems.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

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

This Internet-Draft will expire on May 16, 2013

Copyright Notice

Table of Contents

1. Introduction

The RADIUS protocol as described in [RFC2865], [RFC2866], [RFC5176], and others has traditionally used methods based on MD5 [RFC1321] for per-packet authentication and integrity checks. However, the MD5 algorithm has known weaknesses such as [MD5Attack] and [MD5Break]. As a result, some specifications such as [RFC5176] have recommended using IPSec to secure RADIUS traffic.

While RADIUS over IPSec has been widely deployed, there are difficulties with this approach. The simplest point against IPSec is that there is no straightforward way for a RADIUS application to control or monitor the network security policies. That is, the requirement that the RADIUS traffic be encrypted and/or authenticated is implicit in the network configuration, and is not enforced by the RADIUS application.

This specification takes a different approach. We define a method for using DTLS [RFC6347] as a RADIUS transport protocol. This approach has the benefit that the RADIUS application can directly monitor and control the security policies associated with the traffic that it processes.

Another benefit is that RADIUS over DTLS continues to be a UDP-based protocol. This continuity ensures that existing network-layer infrastructure (firewall rules, etc.) does not need to be changed when RADIUS clients and servers are upgraded to support RADIUS over DTLS.

This specification does not, however, solve all of the problems associated with RADIUS. The DTLS protocol does not add reliable or in-order transport to RADIUS. DTLS also does not support fragmentation of application-layer messages, or of the DTLS messages themselves. This specification therefore shares with traditional RADIUS the issues of order, reliability, and fragmentation.

1.1. Terminology

This document uses the following terms:

RADIUS/DTLS
: This term is a short-hand for "RADIUS over DTLS".

RADIUS/DTLS client
: This term refers both to RADIUS clients as defined in [RFC2865], and to Dynamic Authorization clients as defined in [RFC5176], that implement RADIUS/DTLS.

RADIUS/DTLS server
: This term refers both to RADIUS servers as defined in [RFC2865], and to Dynamic Authorization servers as defined in [RFC5176], that implement RADIUS/DTLS.

RADIUS/UDP
: RADIUS over UDP, as defined in [RFC2865].

RADIUS/TLS
: RADIUS over TLS, as defined in [RFC6614].

silently discard
: This means that the implementation discards the packet without further processing. See Section X.Y for additional requirements on packets being silently discarded.

1.2. 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Building on Existing Foundations

Adding DTLS as a RADIUS transport protocol requires a number of changes to systems implementing standard RADIUS. This section outlines those changes, and defines new behaviors necessary to implement DTLS.

2.1. Changes to RADIUS

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/DTLS:

* 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 "obfuscated" attributes such as User-Password and Tunnel-Password.
* UDP port numbering and relationship between code and port

In short, the application creates a RADIUS packet as usual, and then instead of sending it over a UDP socket, sends the packet to a DTLS layer for encapsulation. DTLS then acts as a transport layer for RADIUS, hence the names "RADIUS/UDP" and "RADIUS/DTLS".

The requirement that RADIUS remain largely unchanged ensures the simplest possible implementation and widest interoperability of this specification.

We note that the DTLS encapsulation of RADIUS means that the minimum and maximum UDP packet sizes increase by the DTLS overhead. Implementations should be aware of this, and take it into account when allocating buffers to read and write RADIUS/DTLS packets.

The only changes made from RADIUS/UDP to RADIUS/DTLS are the following two items:

(1) The Length checks defined in [RFC2865] Section 3 MUST use the length of the decrypted DTLS data instead of the UDP packet length.

(2) The shared secret secret used to compute the MD5 integrity checks and the attribute encryption MUST be "radius/dtls".

All other aspects of RADIUS are unchanged.

2.2. Similarities with RADIUS/TLS

While this specification can be thought of as RADIUS/TLS over UDP instead of TCP, there are some differences between the two methods. The bulk of [RFC6614] applies to this specification, so we do not repeat it here.

This section explains the differences between RADIUS/TLS and RADIUS/DTLS, as semantic "patches" to [RFC6614]. The changes are as follows:

* We replace references to "TCP" with "UDP"

* We replace references to "RADIUS/TLS" with "RADIUS/DTLS"

* We replace references to "TLS" with "DTLS"

Those changes are sufficient to cover the majority of the differences between the two specifications. The next section reviews some more detailed changes from [RFC6614], giving additional commentary only where necessary.

2.2.1. Changes from RADIUS/TLS to RADIUS/DTLS

Section 2.1 does not apply to RADIUS/DTLS. The relationship between RADIUS packet codes and UDP ports in RADIUS/DTLS is unchanged from RADIUS/UDP.

Section 2.2 applies also to RADIUS/DTLS, except for the recommendation that implementations "SHOULD" support TLS_RSA_WITH_RC4_128_SHA. This recommendation is a historical artifact of RADIUS/TLS, and does not apply to RADIUS/DTLS.

Section 2.3 does not apply to RADIUS/DTLS.

Section 2.4 does not apply to RADIUS/DTLS. The relationship between RADIUS packet codes and UDP ports in RADIUS/DTLS is unchanged from RADIUS/UDP.

Section 3.3 item (1) does not apply to RADIUS/DTLS. Each RADIUS packet is encapsulated in one DTLS packet, and there is no "stream" of RADIUS packets inside of a TLS session. Implementors MUST enforce the requirements of [RFC2865] Section 3 for the RADIUS Length field,

using the length of the decrypted DTLS data for the checks. This check replaces the RADIUS method of using the length field from the UDP packet.

Section 3.3 item (3) does not apply to RADIUS/TDLS. The relationship between RADIUS packet codes and UDP ports in RADIUS/DTLS is unchanged from RADIUS.

Section 3.3 item (4) does not apply to RADIUS/DTLS. As RADIUS/DTLS still uses UDP for a transport, the use of negative ICMP responses is unchanged from RADIUS.

2.2.2. Reinforcement of RADIUS/TLS

We re-iterate that much of [RFC6614] applies to this document. Specifically, Section 4 and Section 6 of that document are applicable in their entirety to RADIUS/DTLS.

3. Reception of Packets

As this specification permits implementations to to accept both RADIUS/UDP and RADIUS/DTLS packets on the same port, we require a method to disambiguate packets between the two protocols. This method is applicable only to RADIUS/DTLS servers. RADIUS/DTLS clients SHOULD use connected sockets, as discussed in Section X.Y, below.

RADIUS/DTLS servers MUST maintain a boolean "DTLS Required" flag for each client that indicates if it requires a client to use RADIUS/DTLS. The interpretation of this flag is as follows. If the flag is "true" then the client supports RADIUS/DTLS, and all packets from that client MUST be processed as RADIUS/DTLS. If the flag is "false", then the client supports RADIUS/UDP, but may still support RADIUS/DTLS. Packets from the client need to be examined to see if they are RADIUS/UDP or RADIUS/DTLS.

The "DTLS Required" flag MUST be exposed to administrators of the server. As clients are upgraded, administrators can then manually mark them as using RADIUS/DTLS.

Once a RADIUS/DTLS server has established a DTLS session with a client that previously had the flag set to "false", the server MUST set the "DTLS Required" flag to "true". This change requires all subsequent traffic from that client to use DTLS, and prevents bidding-down attacks. The server SHOULD also notify the administrator that it has successfully established the first DTLS session with that client.

Note that this last requirement on servers can impose significant changes for clients. Clients can no longer have multiple independent RADIUS implementations or processes that originate RADIUS/UDP and RADIUS/DTLS packets. Instead, they need to use only one transport layer, either UDP or DTLS.

It is therefore RECOMMENDED that RADIUS/DTLS clients use a local proxy which arbitrates all traffic between the client and any servers. The proxy SHOULD accept traffic only from the authorized subsystems on the client machine, and SHOULD proxy that traffic to one or more known servers.

4. Connection Management

Where [RFC6614] can rely on the TCP state machine to perform connection tracking, this specification cannot. As a result, implementations of this specification will need to perform connection management of the DTLS session in the application layer.

4.1. Server Connection Management

A RADIUS/DTLS server MUST maintain a table that tracks ongoing client connections based on a key composed of the following 4-tuple:

* source IP address
* source port
* destination IP address
* destination port

Note that this table is independent of IP address version (IPv4 or IPv6).

Each table entry contains the following information:

Protocol Type
: A flag which is either "RADIUS/UDP" for old-style RADIUS traffic, or "RADIUS/DTLS" for RADIUS/DTLS connections.

DTLS Data
: An implementation-specific variable containing information about the active DTLS connection. For non-DTLS connections, this variable MUST be empty.

Last Packet
: A variable containing a timestamp which indicates when the last valid packet was received for this connection. Packets which are "silently discarded" MUST NOT update this variable.

Each entry may contain other information, such as idle timeouts, connection lifetimes, and other implementation-specific data.

RADIUS/DTLS servers SHOULD NOT use connected sockets to read DTLS packets from a client. This recommendation is because a connected UDP socket will accept packets only from one source IP address and port. This limitation would prevent the server from accepting packets from multiple clients on the same port.

4.1.1. Table Management

This tracking table is subject to Denial of Service (DoS) attacks due to the ability of an attacker to forge UDP traffic. RADIUS/DTLS servers SHOULD use the stateless cookie tracking technique described in [RFC6347] Section 4.2.1. DTLS sessions SHOULD NOT be added to the tracking table until a ClientHello packet has been received with an appropriate Cookie value. The requirement to accept RADIUS/UDP and RADIUS/DTLS on the same port makes this recommendation difficult to implement in practice. Server implementation SHOULD therefore have a way of tracking partially setup DTLS connections. Servers SHOULD limit both the number and impact on resources of partial connections.

Entries in the tracking table MUST deleted when a TLS Closure Alert ([RFC5246] Section 7.2.1) or a TLS Error Alert ([RFC5246] Section 7.2.2) is received. Where the specifications require that a packet received via a DTLS session be "silently discarded", the entry in the tracking table corresponding to that DTLS session MUST also be deleted, the DTLS session MUST be closed, and any TLS session resumption parameters for that session MUST be discarded. The implementation MAY provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.

As UDP does not guarantee delivery of messages, RADIUS/DTLS servers MUST also maintain a "Last Packet" timestamp per DTLS session. The timestamp SHOULD be updated on reception of a valid DTLS packet. The timestamp MUST NOT be updated in other situations. When a session has not received a packet for a period of time, it is labelled "idle". The server SHOULD delete idle DTLS session from the tracking table after an "idle timeout". The server MAY cache the TLS session parameters, in order to provide for fast session resumption.

This session "idle timeout" SHOULD be exposed to the administrator as a configurable setting. It SHOULD NOT be set to less than 60 seconds, and SHOULD NOT be set to more than 600 seconds (10 minutes). The minimum value useful value for this timer is determined by the application-layer watchdog mechanism defined in the following section.

RADIUS/DTLS servers SHOULD also keep track of the total number of sessions in the tracking table. They SHOULD stop the creating of new sessions when a large number are already being tracked. This "maximum sessions" number SHOULD be exposed to administrators as a configurable setting.

4.1.2. Protocol Disambiguation

When the "DTLS Required" flag for a client is set to "false", the client may, or may not be sending DTLS packets. For existing connections, protocol disambiguation is simple, the "Protocol Type" field in the tracking table entry is examined. New connections must still be disambiguated.

In order to provide a robust upgrade path, the RADIUS/DTLS server MUST examine the packet to see if it is RADIUS/UDP or RADIUS/DTLS. This examination method is defined here.

We justify the examination methods by analysing the packet formats for the two protocols. We assume that the server has a buffer in which it has received a UDP packet matching no entry on the conneciton tracking table. It must then analyse this buffer to determine which protocol is used to process the packet.

The DTLS record format ([RFC6347] Section 4.1) is shown below, in pseudo-code:

```
struct {
        uint8 type;
        uint16 version;
        uint16 epoch;
        uint48 sequence_number;
        uint16 length;
        uint8 fragment[DTLSPlaintext.length];
} DTLSPlaintext;
```

The RADIUS record format ([RFC2865] Section 3) is shown below, in pseudo-code, with AuthVector.length=16.

```
struct {
        uint8 code;
        uint8 id;
        uint16 length;
        uint8 vector[AuthVector.length];
        uint8 data[RadiusPacket.length - 20];
} RadiusPacket;
```

We can see here that a number of fields overlap between the two

protocols. At first glance, it seems difficult for an application to accept both protocols on the same port. However, this is not the case.

The initial DTLS packet of a connection requires that the type field (first octet) has value 22 (handshake). The first octet of a RADIUS packet is the code field. The code value of 22 has been assigned as Resource-Free-Response. That code is intended to be a response from a server to a client, and will therefore never be sent by a client to a server.

As a result, protocol disambiguation for new connections to a server is straightforward. Only the first octet of the packet needs to be examined to disambiguate RADIUS/DTLS from RADIUS/UDP. If that octet has value 22, then the packet is likely to be RADIUS/DTLS. Otherwise, the packet is likely to be RADIUS/UDP.

4.1.3. Processing Algorithm

When a RADIUS/DTLS server recieves a packet, it uses the following algorithm to process that packet. As with RADIUS/UDP, packets from unknown clients MUST be silently discarded.

The "DTLS Required" flag for that client is examined. If it is set to "true", then the packet MUST be processed as RADIUS/DTLS.

If the "DTLS Required" flag is set to "false", the connection tracking table is examined. Packets matching an existing entry MUST be processed as defined by the "Protocol Type" field of that entry.

If the "DTLS Required" flag is set to "false" and no entry exists in the connection tracking table, then the first octet of the packet is examined. If it has value 22, then the packet MUST be processed as RADIUS/DTLS. Otherwise, the packet MUST be processed as RADIUS/UDP.

In all cases, the packet MUST be checked for correctness. For RADIUS/UDP, any packets which are silently discarded MUST NOT affect the state of any variable in the session tracking table. For RADIUS/DTLS, any packets which are discarded by the DTLS layer MUST NOT affect the state of any variable in the session tracking table. For RADIUS/DTLS, any RADIUS packets which are subsequently silently discarded MUST result in the removal of the associated entry from the connection tracking table.

When the packet matches an existing entry in the connection table, and is accepted for processing by the server, the "Last Packet" timestamp is updated. Where the packet does not match any entry in the connection table, a new connection is created using the 4-tuple

key defined above. The "Protocol Type" flag for that connection is set to "RADIUS/DTLS", or "RADIUS/UDP", as determined by examining the first octet of the packet.

When a server has the clients "DTLS Required" flag set to "false", it MUST set the flag to "true" after establishing a DTLS session with that client. It MUST NOT set the flag to "true" until a DTLS session has been fully established. Doing so would mean that attackers could perform a DoS attack by sending forged DTLS ClientHello packets to a server.

4.2. Client Connection Management

Clients SHOULD use "connected" UDP sockets for RADIUS/DTLS traffic. A connected socket will then rely on the operating system to perform connection tracking. Clients SHOULD NOT use "unconnected" sockets for RADIUS/DTLS traffic. Using unconnected sockets would require the client to implement a connection tracking table, which is complex and unnecessary.

Once a DTLS session is established, a RADIUS/DTLS client SHOULD use the application-layer watchdog algorithm defined in [RFC3539] to determine server responsiveness. The Status-Server packet defined in [RFC5997] MUST be used as the "watchdog packet" in the watchdog algorithm.

RADIUS/DTLS clients SHOULD pro-actively close sessions when they have been idle for a period of time. Clients SHOULD close a session when no traffic other than watchdog packets and (possibly) watchdog responses have been sent for three watchdog timeouts. This behavior ensures that clients do not waste resources on the server by causing it to track idle sessions.

RADIUS/DTLS clients MUST NOT send both RADIUS/UDP and RADIUS/DTLS packets over the same key of (source IP, source port, destination IP, destination port) as defined in Section 4.1, above . Doing so would make it impossible to correctly process either kind of packet.

RADIUS/DTLS clients SHOULD NOT send both RADIUS/UDP and RADIUS/DTLS packets to different servers from the same source socket. This practice causes increased complexity in the client application, and increases the potential for security breaches due to implementation issues.

RADIUS/DTLS clients SHOULD use TLS session resumption. This practice lowers the time and effort required to start a DTLS session with a server, and increases network responsiveness.

5. Diameter Considerations

This specification defines a transport layer for RADIUS. It makes no other changes to the RADIUS protocol. As a result, there are no Diameter considerations.

6. IANA Considerations

This specification does not create any new registries, nor does it require assignment of any protocol parameters.

7. Security Considerations

This entire specification is devoted to discussing security considerations related to RADIUS. However, we discuss a few additional issues here.

This specification relies on the existing DTLS, RADIUS/UDP, and RADIUS/TLS specifications. As a result, all security considerations for DTLS apply to the DTLS portion of RADIUS/DTLS. Similarly, the TLS and RADIUS security issues discussed in [RFC6614] also apply to this specification. All of the security considerations for RADIUS apply to the RADIUS portion of the specification.

However, many security considerations raised in the RADIUS documents are related to RADIUS encryption and authorization. Those issues are largely mitigated when DTLS is used as a transport method. The issues that are not mitigated by this specification are related to the RADIUS packet format and handling, which is unchanged in this specification.

The only new portion of the specification that could have security implications is a servers ability to accept both RADIUS and DTLS packets on the same port. The filter that disambiguates the two protocols is simple, and is just a check for the value of one octet. We do not expect this check to have any security issues.

We also note that nothing prevents malicious clients from sending DTLS packets to existing RADIUS implementations, or RADIUS packets to existing DTLS implementations. There should therefore be no issue with clients sending RADIUS/DTLS packets to legacy servers that do not support the protocol. These packets will be silently ignored, and will not change the security profile of the server.

7.1. Legacy RADIUS Security

We reiterate here the poor security of the legacy RADIUS protocol. It is RECOMMENDED that all RADIUS clients and servers implement this

specification. New attacks on MD5 have appeared over the past few years, and there is a distinct possibility that MD5 may be completely broken in the near future.

The existence of fast and cheap attacks on MD5 could result in a loss of all network security which depends on RADIUS. Attackers could obtain user passwords, and possibly gain complete network access. We cannot overstate the disastrous consequences of a successful attack on RADIUS.

We also caution implementors (especially client implementors) about using RADIUS/DTLS. It may be tempting to use the shared secret as the basis for a TLS pre-shared key (PSK) method, and to leave the user interface otherwise unchanged. This practice MUST NOT be used. The administrator MUST be given the option to use DTLS. Any shared secret used for RADIUS/UDP MUST NOT be used for DTLS. Re-using a shared secret between RADIUS/UDP and RADIUS/DTLS would negate all of the benefits found by using DTLS.

RADIUS/DTLS client implementors MUST expose a configuration that allows the administrator to choose the cipher suite. Where certificates are used, RADIUS/DTLS client implementors MUST expose a configuration which allows an administrator to configure all certificates necessary for certificate-based authentication. These certificates include client, server, and root certificates.

When using PSK methods, RADIUS/DTLS servers MUST support keys (i.e. shared secrets) that are at least 32 characters in length. These keys SHOULD be able to contain arbitrary binary data. RADIUS/DTLS server administrators MUST use strong shared secrets for those PSK methods. We RECOMMEND using keys derived from a cryptographically secure pseudo-random number generator (CSPRNG). For example, a reasonable key may be 32 characters of a SHA-256 hash of at least 64 octetss of data taken from a CSPRNG. If this method seems too complicated, a certificate-based TLS method SHOULD be used instead.

The previous RADIUS practice of using shared secrets that are minor variations of words is NOT RECOMMENDED, as it would negate nearly all of the security of DTLS.

7.2. Resource Exhaustion

The use of DTLS allows DoS attacks, and resource exhaustion attacks which were not possible in RADIUS/UDP. These attacks are the same as described in [RFC6614] Section X.Y.

Use of the connection tracking table defined in Section X.Y can result in resource exhaustion. Servers MUST therefore limit the

absolute number of entries in the table. Servers MUST limit the number of partially open DTLS sessions. These limits SHOULD be exposed to the administrator as configurable settings.

7.3. Network Address Translation

Network Address Translation (NAT) is fundamentally incompatible with RADIUS/UDP. RADIUS/UDP uses the source IP address to determine the shared secret for the client, and NAT hides many clients behind one source IP address.

The migration flag described above in Section 3 is also tracked per source IP address. Using a NAT in front of many RADIUS clients negates the function of the flag, making it impossible to migrate multiple clients in a secure fashion.

In addition, port re-use on a NAT gateway means that packets from different clients may appear to come from the same source port on the NAT. That is, a RADIUS server may receive a RADIUS/DTLS packet from a client IP/port combination, followed by the reception of a RADIUS/UDP packet from that same client IP/port combination. If this behavior is allowed, it would permit a downgrade attack to occur, and would negate all of the security added by RADIUS/DTLS.

As a result, RADIUS clients SHOULD NOT be located behind a NAT gateway. If clients are located behind a NAT gateway, then a secure transport such as DTLS MUST be used. As discussed below, a method for uniquely identifying each client MUST be used.

7.4. Wildcard Clients

Some RADIUS server implementations allow for "wildcard" clients. That is, clients with an IPv4 netmask of other than 32, or an IPv6 netmask of other than 128. That practice is NOT RECOMMENDED for RADIUS/UDP, as it means multiple clients use the same shared secret.

When a client is a "wildcard", then RADIUS/DTLS MUST be used. Clients MUST be uniquely identified, and any certificate or PSK used MUST be unique to each client.

8. References

8.1. Normative references

[RFC2865]
Rigney, C., Willens, S., Rubens, A. and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3539]
Aboba, B. et al., "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, June 2003.

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

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

[RFC6347]
Rescorla E., and Modadugu, N., "Datagram Transport Layer Security", RFC 6347, April 2006.

[RFC6614]
Winter. S, et. al., "TLS encryption for RADIUS over TCP", RFFC 6614, May 2012

8.2. Informative references

[RFC1321]
Rivest, R. and S. Dusse, "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2119]
Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March, 1997.

[RFC2866]
Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC5176]
Chiba, M. et al., "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

[MD5Attack]
Dobbertin, H., "The Status of MD5 After a Recent Attack", CryptoBytes Vol.2 No.2, Summer 1996.

[MD5Break]
Wang, Xiaoyun and Yu, Hongbo, "How to Break MD5 and Other Hash Functions", EUROCRYPT. ISBN 3-540-25910-4, 2005.

Acknowledgments

Parts of the text in Section 3 defining the Request and Response Authenticators were taken with minor edits from [RFC2865] Section 3.

Authors' Addresses

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

Email: aland@freeradius.org