Network Working Group
INTERNET-DRAFT
Category: Proposed Standard
<draft-ietf-radext-tcp-transport-02.txt>
Expires: June 16, 2009
16 December 2008

A. DeKok
FreeRADIUS

RADIUS Over TCP
draft-ietf-radext-tcp-transport-02

Copyright Notice

Abstract

The Remote Authentication Dial In User Server (RADIUS) Protocol has traditionally used the User Datagram Protocol (UDP) as it's underlying transport layer. This document defines RADIUS over the Transmission Control Protocol (TCP).

Table of Contents

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

1. Introduction

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

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

* Packet fragmentation. [RFC2865] Section 3 permits RADIUS packets up to 4096 octets in length. These packets are larger than the default Internet MTU (576), resulting in fragmentation of the packets at the IP layer. 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 can reliably detect when the other is down. This information has to be deduced instead from the absence of a reply to a request.

As RADIUS is widely deployed, and has been widely deployed for well over a decade, these issues are relatively minor. However, new systems may be interested in choosing a different set of trade-offs than those outlined in [RFC2865] Section 2.4. For those systems, we define RADIUS over TCP.

1.1. Applicability of Reliable Transport

The intent of this document is to address transport issues related to RadSec [RADSEC]. The use of "bare" TCP transport is NOT RECOMMENDED, as there has been little implementational or operational experience with it. Additionally, [RFC2865] Section 2.4 contains a list of reasons why UDP was originally chosen as the transport protocol for RADIUS. UDP SHOULD be used as transport protocol in all cases where the rationale given in [RFC2865] Section 2.4 applies.

There are a number of benefits to using a reliable transport. For example, when RADIUS is used to carry EAP conversions [RFC3579], the EAP exchanges may involve 5 round trips at the RADIUS application layer. We may assume a probability P of packet loss in each direction (with P having a value of 1% or less). Any one authentication attempt will then have at least one lost packet, with a probability of approximately (10 * P).

These lost packets require the supplicant and/or the NAS to re-transmit packets at the application layer. The difficulty with this approach is that retransmission implementations have historically been poor. Some implementations retransmit packets, others do not, and others send new packets rather then performing retransmission. Some implementations are incapable of detecting EAP retransmissions, and will instead treat the retransmitted packet as an error.

These retransmissions have a high likelihood of causing the entire authentication session to fail. For a system with a million logins a day, and having a packet loss probability of P=0.01%, we expect that 0.1% of connections will experience a lost packet. That is, 1,000 user sessions each day will experience authentication failure.

In addition, transport of fragmented UDP packets is a poorly tested code path on network devices. Some devices appear to be incapable of transporting fragmented UDP packets, meaning that the packet loss rate for fragmented packets approaches 100 percent. The net effect can be to prevent the deployment of authentication methods such as EAP-TLS that require large RADIUS packets.

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 reliable delivery. In addition, most TCP implementations discover Path MTU better than RADIUS application implementations, resulting in significantly fewer fragmented packets. Modern TCP implementations also implement anti-spoofing provisions, which is more difficult to do in UDP applications.

Transporting RADIUS over TCP means that the RADIUS applications can leverage these additional protections offered by TCP.

However, there are also some drawbacks to using TCP. RADIUS over TCP has some drawbacks, as noted in [RFC2865] Section 2.4. [RFC3539] Section 2 discusses further issues with using TCP as a transport for Authentication, Authorization, and/or Accounting (AAA) protocols such as RADIUS.

Specifically, as noted in [RFC3539] Section 2.1, for systems originating low numbers of RADIUS request packets, inter-packet spacing is often larger than the packet RTT. In those situations, RADIUS over TCP SHOULD NOT be used.

In general, RADIUS clients generating small amounts of RADIUS traffic SHOULD NOT use TCP. This suggestion will usually apply to most NASes, and to most clients that originate CoA-Request and Disconnect-Request packets.

RADIUS over TCP is most applicable to RADIUS proxies that exchange a large volume of packets with RADIUS clients and servers (10's to 1000's of packets per second). In those situations, RADIUS over TCP may be a good fit, and may result in increased network stability and performance.

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 RADIUS authentication, authorization, and/or accounting (AAA) server is an entity that provides one or more AAA services to a NAS.

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

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

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

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

Adding TCP as a RADIUS transport has a number of impacts on the protocol, on applications using the protocol, and on networks that deploy the protocol. In short, RADIUS over TCP is little more than sending RADIUS formatted messages over a TCP connection.

As always, there are additional details that need to be discussed. This section outlines the various impacts of using RADIUS over TCP,

and the discusses the proposal in more detail.

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 over 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 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 Over TCP

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

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

These ports are unused by existing RADIUS applications. Implementations SHOULD use the assigned values 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 over TCP MUST NOT use TCP ports 1645 or 1646 as the default ports for this specification.

2.3. Management Information Base (MIB)

The MIB Module definitions in [RFC4668], [RFC4669], [RFC4670], [RFC4671], [RFC4672], and [RFC4673] each contain only one reference to UDP. These references are in the DESCRIPTION field of the MIB

Module definition, and are in the form of "The UDP port" or "the UDP destination port".

Implementations of RADIUS over TCP SHOULD re-use these MIB Modules to perform statistics counting for RADIUS over TCP connections. However, implementors are warned that there is no way for these MIB Modules to distinguish between packets sent over UDP or over TCP transport. Similarly, there is no requirement in RADIUS that the RADIUS services offered over UDP on a particular IP address and port are identical to the RADIUS services offered over TCP on a particular IP address and the same (numerical) port.

Implementations of RADIUS over TCP SHOULD include the protocol (UDP) or (TCP) in the radiusAuthServIdent, radiusAuthClientID, radiusAuthClientIdentifier, radiusAccServIdent, radiusAccClientID, or radiusAccClientIdentifier fields of the MIB Module. This information can help the administrator distinguish capabilities of systems in the network.

2.4. Interaction with RadSec

IANA has already assigned TCP ports for RadSec (i.e. RADIUS over TLS over TCP), as outlined below:

* radsec 2083/tcp

This value SHOULD be used as the default port for RADIUS over TLS (i.e. RadSec). The "radius" port (1812/tcp) SHOULD NOT be used for RadSec.

2.5. RADIUS Proxies

As RADIUS is a "hop by hop" protocol, a RADIUS proxy effectively 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.

If a request is proxied through intermediate proxies, it is not possible to detect which of the later hops is responsible for the absence of a reply. An intermediate proxy also cannot signal that the outage lies in a later hop because RADIUS does not have the ability to carry such signalling information. This issue is further exacerbated by some proxy implementations that do not reply to a client if they do not receive a reply to a proxied request.

When UDP was 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 over TCP, the continued existence of the TCP connection SHOULD be used to deduce that the service on the other end of the connection is still responsive. Further, the application layer watchdog defined in [RFC3539] Section 3.4 enables clients to determine that the server is "live", even though it may not have responded recently to other, non-watchdog requests.

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

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

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

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

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

2.6. TCP Specific Issues

The guidelines defined in [RFC3539] for implementing an AAA protocol operating over a reliable transport MUST be followed by implementors of this specification.

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

Implementations MUST NOT confuse UDP and TCP transport. 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 be configurable to have different shared secrets for UDP and TCP to the same destination IP address and numerical port.

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

Whenever the underlying operating system permits the use of TCP keepalive socket options, their use is RECOMMENDED.

2.6.1. Duplicates and Retransmissions

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

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

If a TCP connection is broken or closed, any cached RADIUS response

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

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

As noted previously, RADIUS packets SHOULD NOT be re-transmitted to the same destination IP and numerical port, but over a different transport layer. There is no guarantee in RADIUS that the two ports are in any way related. This requirement does not forbid the practice of putting multiple servers into a fail-over or load-balance pool.

Much of the discussion in this section can be summarized by the following requirement. RADIUS requests MAY be re-transmitted verbatim only if the following 5-tuple (Client IP address, Client port, Transport Protocol, Server IP address, Server port) remains the same. If any field of that 5-tuple changes, the packet MUST NOT be considered to be a re-transmission. Instead, the packet MUST be considered to be a new request, and be treated accordingly. This involves updating header calculations, packet signatures, associated timers and counters, etc.

The above requirement is necessary, but not sufficient in all cases. Other specifications give additional situations where the packet is to be considered as a new request. Those recommendations MUST also be followed.

2.6.2. Shared Secrets

The use of shared secrets in calculating the Response Authenticator, and other attributes such as User-Password or Message-Authenticator [RFC3579] MUST be unchanged from previous specifications.

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

2.6.3. Malformed Packets and Unknown Clients

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

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

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

* Packet 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 applicable).
* Packet where the Response Authenticator fails validation (where applicable).
* Packet where the Message-Authenticator attribute fails validation (where applicable).

TCP connections MAY be closed if any of the following circumstances are seen. Alternatively, the TCP connection MAY remain open if any of the following circumstances are seen, but the invalid packet MUST BE silently discarded.

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

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

2.6.4. 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 [STATUS], 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 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 will likely require fundamental changes to the RADIUS protocol, and as such, is outside of the scope of this specification.

2.6.5. EAP Sessions

When RADIUS clients send EAP requests using RADIUS over TCP, they SHOULD choose the same TCP connection for all packets related to one EAP conversation. A simple method that may work in many situations is to hash the contents of the Calling-Station-Id attribute, which normally contains the MAC address. The output of that hash can be used to select a particular TCP connection.

If this practice is used, then the client SHOULD also reserve one RADIUS Id per TCP connection for a particular EAP session.

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 as per the algorithm in [RFC3539] Appendix A.

2.6.6. 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. We RECOMMEND that implementors of this specification familiarize themselves with TCP application programming concepts. We RECOMMEND also that existing TCP applications be examined with an eye

to robustness, performance, scalability, etc.

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

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

3. Diameter Considerations

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

4. IANA Considerations

This document requires no action by IANA.

5. Security Considerations

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

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

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

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

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

6.2. Informative References

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

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, September 2003.

[RFC4668] Nelson, D, "RADIUS Authentication Client MIB for IPv6", RFC 4668, August 2006.

[RFC4669] Nelson, D, "RADIUS Authentication Server MIB for IPv6", RFC 4669, August 2006.

[RFC4670] Nelson, D, "RADIUS Accounting Client MIB for IPv6", RFC 4670, August 2006.

[RFC4671] Nelson, D, "RADIUS Accounting Server MIB for IPv6", RFC 4671, August 2006.

[RFC4672] Nelson, D, "RADIUS Dynamic Authorization Client MIB", RFC 4672, August 2006.

[RFC4673] Nelson, D, "RADIUS Dynamic Authorization Server MIB", RFC 4673, August 2006.

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

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

[STATUS] DeKok, A., "Use of Status-Server Packets in the Remote Authentication Dial In User Service (RADIUS) Protocol", draft-ietf-radext-status-server-02.txt, November 2008 (work in progress).

[RADSEC] Winter, S. et. al., "TLS encryption for RADIUS over TCP (RadSec)", draft-ietf-radext-radsec-02.txt, October 2008 (work in progress).

Acknowledgments

None at this time.

Authors' Addresses

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

Email: aland@freeradius.org

Open issues

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

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