RADEXT
Internet-Draft
Intended status: Standards Track
Expires: 13 November 2026

S. Yue
China Mobile
C. Lin
New H3C Technologies
F. Yang
China Mobile

12 May 2026

RADIUS over QUIC
draft-yl-radext-quic-transport-04

Abstract

The Remote Authentication Dial-In User Server (RADIUS) Protocol can use the User Datagram Protocol (UDP) and the Transport Control Protocol (TCP) as its underlying transport layer. But 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. QUIC inherently supports encryption (using TLS 1.3), which could provide a higher level of security. And QUIC supports multiple streams over a single connection, enhancing throughput and efficiency. This document defines RADIUS over the QUIC transport protocol, named RADIUSoQUIC.

Status of This Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 13 November 2026.

Copyright Notice

Table of Contents

1. Introduction

RADIUS (Remote Authentication Dial-In User Service) is a networking protocol that provides centralized authentication, authorization, and accounting for users who connect and use a network service. It is typically used in environments such as Internet service providers (ISPs), corporate networks, and other places where secure access management is required.

The RADIUS protocol was originally defined in [RFC2865] as using the User Datagram Protocol (UDP) for the underlying transport layer. Since UDP has some limitations (such as unreliable transport, packet fragmentation, connectionless transport, and lack of congestion control), the TCP transport is recommended to be used [RFC6613] when another method such as IPsec [RFC4301] or RADIUS/TLS [RFC6614] is used to provide additional confidentiality and security of RADIUS in inter-server communications scenarios, such as inter-domain communication between proxies.

QUIC is developed based on the UDP protocol and aims to address certain limitations of TCP. Compared with the traditional TCP/UDP protocol used by RADIUS, QUIC provides the following enhancements:

* Improved Security. QUIC inherently supports encryption (using TLS 1.3), which could provide a higher level of security for RADIUS communications compared to the standard RADIUS protocol that relies heavily on IPsec or TLS over TCP for encryption. This can also greatly simplify security-related configurations.

* Enhanced Performance. QUIC reduces connection establishment time with its zero-round-trip setup for repeat connections, potentially speeding up the authentication process significantly.

* Reliability and Robustness. QUIC’s congestion control, loss recovery, and reduced latency features could make RADIUS communications more reliable, especially over less-stable network connections.

* Connection Migration. The ability of QUIC to handle connection migrations seamlessly could enhance RADIUS network mobility between client (running on Network Access Server) and server, allowing client to change networks without disrupting authentication sessions.

* Multiplexing. QUIC’s support for multiple streams over a single connection could enable multiple RADIUS transactions to be processed simultaneously, enhancing throughput and efficiency. This can alleviate the congestion problem of large-scale authentication messages and provide users with faster authentication services and higher authentication success rate via fewer connections. Independent streams can also prevent Head-of-Line (HoL) blocking between different RADIUS transaction messages.

Therefore, for inter-server communication scenarios that require secure connections and reliable data transmission, QUIC is a competitive transport protocol for the message transmission mechanism of RADIUS Protocol. This document specifies how to use QUIC as the transport protocol for RADIUS Protocol, named RADIUSoQUIC.

2. Terminology and Definitions

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

In this document, the terms "client" and "server" are used to refer to the two ends of the QUIC connection. The client actively initiates the QUIC connection. The terms "RADIUS client" and "RADIUS server" are used to refer to the two ends of the RADIUS Protocol session. Generally, an "RADIUS client" provides an access service for a user to a network, meanwhile a "RADIUS server" provides one or more of authentication, authorization, and/or accounting (AAA) services to a Network Access Server (NAS). In addition, A RADIUS Proxy acts as a RADIUS server to the NAS, and a RADIUS client to the RADIUS server.

* Client: The endpoint that initiates a QUIC connection, the RADIUS client or RADIUS proxy.

* Server: The endpoint that accepts a QUIC connection, the RADIUS server or RADIUS proxy.

3. Connection Management

3.1. Connection Establishment

QUIC connection establishment is described in [RFC9000]. During establishing connection, RADIUSoQUIC support is indicated by selecting the Application-Layer Protocol Negotiation (ALPN) [RFC7301] token as listed in the IANA Section 7 in the TLS handshake.

The RADIUS Client or Proxy as client should be the initiator of the QUIC connection to the RADIUS Server meanwhile the RADIUS Server or Proxy as server acts as a connection acceptor.

3.2. Connection Termination

3.2.1. QUIC Connection Termination Process

The typical QUIC connection termination process is described in [RFC9000].

3.2.2. RADIUSoQUIC Considerations for Connection Termination

When an RADIUS session is implemented based on a QUIC connection, the idle timeout should be disabled or the QUIC max_idle_timeout should be set appropriately in order to keep the QUIC connection persistent even if the RADIUS session is idle.

If the QUIC 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 QUIC connection. Since 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 QUIC 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 QUIC connection. No response to these requests can be sent over the QUIC 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.

RADIUS clients using QUIC 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 QUIC MUST NOT decide that a RADIUS server is unresponsive until all QUIC connections to it have been marked DOWN.

A client should proactively close connections or mark a server as DOWN due to an administrative decision.

4. Stream Mapping and Usage

QUIC [RFC9000] uses multiple simultaneous streams to carry data in one direction. QUIC Streams provide a lightweight, ordered byte-stream abstraction to an application. Streams can be unidirectional or bidirectional meanwhile streams can be initiated by either the client or the server. Unidirectional streams carry data in one direction: from the initiator of the stream to its peer. Bidirectional streams allow for data to be sent in both directions.

QUIC uses Stream ID to identify the stream. The least significant bit (0x1) of the stream ID identifies the initiator of the stream (client with the bit set to 0). The second least significant bit (0x2) of the stream ID distinguishes between bidirectional streams (with the bit set to 0) and unidirectional streams [RFC9000].

RADIUS packets include request packets and response packets. RADIUS request packet is 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 is 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 [RFC6613].

4.1. Bidirectional Stream between Client and Server

If RADIUS request packets are carried via transport protocol from RADIUS client to RADIUS server, RADIUS response packet are needed to be sent from RADIUS server to RADIUS client in response to the RADIUS request packets. Therefore, the RADIUS connection should be bidirectional between Client and Server.

Based on the above description, The RADIUS request messages are initiated by the Client and the replies are needed from the Server. So the RADIUS messages MAY be mapped into one bidirectional stream whose stream type is 0x0 according to section 2.1 of [RFC9000].

To enhance transmission performance during large-scale processing of authentication, accounting, and authorization requests, RADIUS messages can be transmitted over multiple configurable bidirectional streams.

When using multiple streams, all messages belonging to the same RADIUS transaction must remain within a single stream to maintain protocol integrity. A RADIUS transaction refers to a complete protocol interaction process, including all message exchanges between the client (such as NAS) and the server (such as RADIUS server) from the request to the final response. RADIUS transactions can be divided into authentication transactions, accounting transactions, and dynamic authorization transactions.

4.1.1. Multi-Stream Use Case

As described above, there are three types of RADIUS transactions; therefore, the client can create three QUIC bidirectional streams to carry these three RADIUS transactions, as shown in Figure 1. Stream 0 is used for authentication, stream 1 for accounting, and stream 2 for authorization transactions.

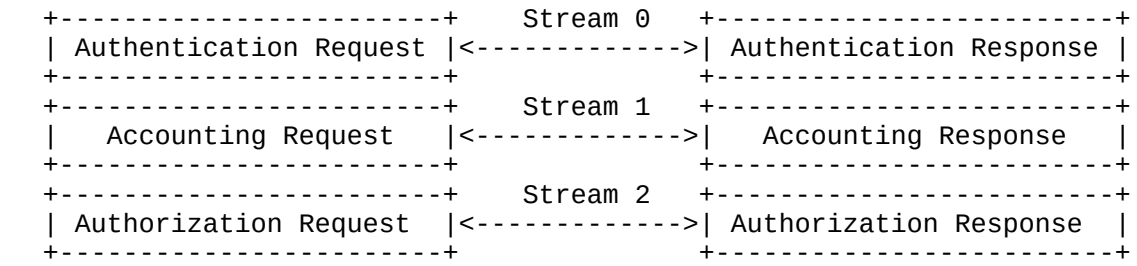

Figure 1: Multiple Stream Use Case

5. Endpoint Authentication

RADIUSoQUIC uses QUIC which uses TLS version 1.3 or greater. Therefore, the TLS handshake process can be used for RADIUSoQUIC endpoint authentication. A third-party authentication mechanism can also be applied for RADIUSoQUIC endpoint authentication, such as a TLS client certificate.

6. Operational Considerations

6.1. Configuration Parameters

The decision to use RADIUSoQUIC instead of the TCP-based/UDP-based mechanism is an operational decision, and an implementation MUST provide a configuration mechanism to enable RADIUSoQUIC on the RADIUS session.

A configuration option SHOULD be provided to enable the use of multiple streams, and the number of streams also SHOULD be configurable.

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

6.2. Packet Format

RADIUSoQUIC MUST strictly preserve all specifications of [RFC2865], [RFC2866], and [RFC5176], including packet format, attribute encoding, and security computations (authenticators, dynamic or encrypted attributes).

The use of QUIC 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. In addition, Clients and servers MUST be able to store and manage secrets based on the key described at the section 5 of [RFC9001].

The changes required to implement this specification primarily affect the RADIUS components responsible for sending and receiving network packets.

6.3. QUIC Port

The default destination port for RADIUSoQUIC is UDP/TBD (assigned by IANA), which handles authentication, accounting, and dynamic authorization without port differentiation. Source ports may be arbitrarily chosen.

RADIUS over UDP or TCP [RFC2865] [RFC6613] uses distinct ports for authentication, accounting, and dynamic authorization. In contrast, RADIUSoQUIC utilizes a single port for all packet types. However, RADIUSoQUIC maintains the fundamental client-server model where:

* Clients send authentication requests and process replies for user sessions.

* Servers receive requests, process them, and return responses.

The normative rules defining acceptable packet types for clients and servers maintain identical packet flow behavior to RADIUS over UDP or TCP.

6.4. Management Information Base (MIB)

The MIB modules defined in [RFC4668], [RFC4669], [RFC4670], [RFC4671], [RFC4672], and [RFC4673] were specifically designed for RADIUS over UDP transport and therefore do not support either RADIUS over TCP [RFC6613] or RADIUSoQUIC implementations. These existing MIB modules should not be reused for RADIUSoQUIC connection statistics monitoring, as they lack the necessary constructs to properly track QUIC-specific features such as connection migration, cryptographic handshakes, and stream multiplexing. Future updates to the MIB specifications will be required to adequately support these newer transport protocols.

6.5. Detecting Live Servers

In RADIUS' hop-by-hop architecture, proxies inherently isolate clients from downstream server information, preventing direct awareness of backend server states. While clients can assess their immediate proxy's operational status, they lack visibility into subsequent hops since proxies primarily function as transparent forwarders between NAS and RADIUS servers without generating independent responses (per [RFC2865]). This opacity creates ambiguity when requests go unanswered, as failures could stem from multiple points: packet loss in either the NAS-proxy or proxy-server segments, proxy malfunctions, or actual server outages, with no built-in mechanism to distinguish between these scenarios.

RADIUS clients using RADIUSoQUIC must consider a connection DOWN when either: (1)All active QUIC Connection IDs enter DRAINING state, or (2)The connection receives a valid CONNECTION_CLOSE frame. However, if any Connection ID remains ACTIVE, clients should continue treating the connection as operational until: (1)Path validation fails for all available Connection IDs, and (2)The application-layer watchdog timer expires (as specified in [RFC3539] Appendix A). A RADIUS server should only be declared unresponsive when: (1)All existing QUIC connections have terminated, and (2)No new connections can be established using remaining Connection IDs.

The RADIUSoQUIC do not forbid the practice of a client proactively closing connections or marking a server as DOWN due to an administrative decision.

6.6. Malformed Packets and Unknown Clients

For RADIUSoQUIC implementations, due to QUIC's stream-based reliable delivery, when encountering any condition that would require closing a TCP connection under Section 2.6.4 of [RFC6613], implementations MUST terminate the affected QUIC stream immediately. Additionally, implementations MAY choose to terminate the entire QUIC connection based on local policy.

For two exceptional cases permitting silent discarding for TCP under Section 2.6.4 of [RFC6613], RADIUSoQUIC performs the same operation.

Implementations MAY either close only the offending stream while maintaining the QUIC connection, or terminate the entire QUIC connection. In all cases, the invalid packet MUST be discarded without processing. This approach maintains security while leveraging QUIC's ability to isolate stream failures - a compromised stream won't necessarily disrupt other concurrent RADIUS transactions.

6.7. Limitations of the ID Field

For RADIUSoQUIC, the one-octet RADIUS ID field similarly limits each QUIC connection to 256 simultaneous in-flight RADIUS transactions. However, QUIC's stream multiplexing allows multiple independent transactions to coexist efficiently on a single connection without head-of-line blocking.

To scale beyond 256 concurrent RADIUS transactions, implementations should establish additional QUIC connections as needed. This approach benefits from QUIC's efficient connection establishment, particularly its 0-RTT resumption capability, which minimizes latency overhead when creating new connections. While the ID space limitation remains per connection (as described in Section 2.4 of [RFC3539]), QUIC's reduced connection overhead makes the multi-connection strategy more practical compared to TCP.

6.8. EAP Sessions

For RADIUSoQUIC implementations handling Extensible Authentication Protocol (EAP) sessions [RFC3579], clients SHOULD use the same QUIC stream for all packets within a single EAP session to ensure in-order delivery through QUIC's stream sequencing while maintaining fault isolation and efficient multiplexing. Regarding retransmissions: EAP-level retransmissions MUST NOT cause RADIUS packet retransmissions on the same QUIC stream, though alternative streams or connections may be used if the original becomes unavailable; importantly, QUIC's native loss recovery handles all necessary

transport-layer retransmissions automatically.

7. IANA Considerations

This document creates a new registration for the identification of RADIUSoQUIC in the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established in [RFC7301].

The "RADIUSoQ" string identifies RADIUSoQUIC:

* Protocol: RADIUSoQUIC

* Identification Sequence: 0x52 0x41 0x44 0x49 0x55 0x53 0x6f 0x51 ("RADIUSoQ")

* Specification: This document

In addition, it is requested for IANA to reserve a UDP port TBD for ’RADIUS over QUIC’.

8. Security Considerations

This document replaces the transport protocol layer of RADIUS from other transport protocols to QUIC. The basic protocol specification of RADIUS is not modified, and therefore the new security risks are not introduced to the basic RADIUS protocol. RADIUSoQUIC enhances transport-layer security for RADIUS session according to [RFC9000].

This document does not require to support third-party authentication (e.g., backend Authentication) due to the fact that TLS does not specify this way of authentication. If third-party authentication is needed, TLS client certificates are recommended to be used here.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <https://www.rfc-editor.org/info/rfc2865>.

[RFC3539] Aboba, B. and J. Wood, "Authentication, Authorization and Accounting (AAA) Transport Profile", RFC 3539, DOI 10.17487/RFC3539, June 2003, <https://www.rfc-editor.org/info/rfc3539>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

9.2. Informative References

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, DOI 10.17487/RFC2866, June 2000, <https://www.rfc-editor.org/info/rfc2866>.

[RFC3579] Aboba, B. and P. Calhoun, "RADIUS (Remote Authentication Dial In User Service) Support For Extensible Authentication Protocol (EAP)", RFC 3579, DOI 10.17487/RFC3579, September 2003, <https://www.rfc-editor.org/info/rfc3579>.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <https://www.rfc-editor.org/info/rfc4301>.

[RFC4668] Nelson, D., "RADIUS Authentication Client MIB for IPv6", RFC 4668, DOI 10.17487/RFC4668, August 2006, <https://www.rfc-editor.org/info/rfc4668>.

[RFC4669] Nelson, D., "RADIUS Authentication Server MIB for IPv6", RFC 4669, DOI 10.17487/RFC4669, August 2006, <https://www.rfc-editor.org/info/rfc4669>.

[RFC4670] Nelson, D., "RADIUS Accounting Client MIB for IPv6", RFC 4670, DOI 10.17487/RFC4670, August 2006, <https://www.rfc-editor.org/info/rfc4670>.

[RFC4671] Nelson, D., "RADIUS Accounting Server MIB for IPv6", RFC 4671, DOI 10.17487/RFC4671, August 2006, <https://www.rfc-editor.org/info/rfc4671>.

[RFC4672] De Cnodder, S., Jonnala, N., and M. Chiba, "RADIUS Dynamic Authorization Client MIB", RFC 4672, DOI 10.17487/RFC4672, September 2006, <https://www.rfc-editor.org/info/rfc4672>.

[RFC4673] De Cnodder, S., Jonnala, N., and M. Chiba, "RADIUS Dynamic Authorization Server MIB", RFC 4673, DOI 10.17487/RFC4673, September 2006, <https://www.rfc-editor.org/info/rfc4673>.

[RFC5080] Nelson, D. and A. DeKok, "Common Remote Authentication Dial In User Service (RADIUS) Implementation Issues and Suggested Fixes", RFC 5080, DOI 10.17487/RFC5080, December 2007, <https://www.rfc-editor.org/info/rfc5080>.

[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, DOI 10.17487/RFC5176, January 2008, <https://www.rfc-editor.org/info/rfc5176>.

[RFC6613] DeKok, A., "RADIUS over TCP", RFC 6613, DOI 10.17487/RFC6613, May 2012, <https://www.rfc-editor.org/info/rfc6613>.

[RFC6614] Winter, S., McCauley, M., Venaas, S., and K. Wierenga, "Transport Layer Security (TLS) Encryption for RADIUS", RFC 6614, DOI 10.17487/RFC6614, May 2012, <https://www.rfc-editor.org/info/rfc6614>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

Authors' Addresses

Shengnan Yue
China Mobile
Beijing
China
Email: yueshengnan@chinamobile.com

Changwang Lin
New H3C Technologies
Beijing
China
Email: linchangwang.04414@h3c.com

Feng Yang
China Mobile
Beijing
China
Email: yangfeng@chinamobile.com