Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 4, 2019

V. Vasiliev
Google

May 3, 2019

WebTransport over QUIC
draft-vvv-webtransport-quic-00

Abstract

WebTransport [OVERVIEW] is a protocol framework that enables clients constrained by the Web security model to communicate with a remote server using a secure multiplexed transport. This document describes QuicTransport, a transport protocol that uses a dedicated QUIC [QUIC-TRANSPORT] connection and provides support for unidirectional streams, bidirectional streams and datagrams.

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 November 4, 2019.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

QUIC [QUIC-TRANSPORT] is a UDP-based multiplexed secure transport. It is the underlying protocol for HTTP/3 [I-D.ietf-quic-http], and as such is reasonably expected to be available in web browsers and server-side web frameworks. This makes it a compelling transport to base a WebTransport protocol on.

This document defines QuicTransport, an adaptation of QUIC to WebTransport model. The protocol is designed to be low-overhead on the server side, meaning that server software that already has a working QUIC implementation available would not require a large amount of code to implement QuicTransport. Where possible, WebTransport concepts are mapped directly to the corresponding QUIC concepts.

1.1. Terminology

The keywords "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.

This document follows terminology defined in Section 1.2 of [OVERVIEW].

2. Protocol Overview

Each QuicTransport uses a single dedicated QUIC connection. This allows the peers to exercise a greater level of control over the way their data is being transmitted. However, this also means that multiple instances of QuicTransport cannot be pooled, and thus do not benefit from sharing congestion control context with other potentially already existing connections. Http3Transport [I-D.vvv-webtransport-http3] can be used in situations where such pooling is beneficial.

When a client requests a QuicTransport to be created, the user agent establishes a QUIC connection to the specified address. It verifies that the the server is a QuicTransport endpoint using ALPN, and that the client is allowed to connect to the specified endpoint using "web_accepted_origins" transport parameter. Once the verification succeeds and the QUIC connection is ready, the client can send and receive streams and datagrams.

WebTransport streams are provided by creating an individual unidirectional or bidirectional QUIC stream. WebTransport datagrams are provided through the QUIC datagram extension [QUIC-DATAGRAM].

3. Connection Establishment

In order to establish a QuicTransport session, a QUIC connection must be established. From the client perspective, the session becomes established when the client receives a TLS Finished message from the server.

3.1. Identifying as QuicTransport

In order to identify itself as a WebTransport application,

QuicTransport relies on TLS Application-Layer Protocol Negotiation [RFC7301]. The user agent MUST request the ALPN value of "wq" and it MUST NOT establish the session unless that value is accepted.

3.2. Verifying the Origin

In order to verify that the client is authorized to access a specific WebTransport server, QuicTransport has a mechanism to verify the origin [RFC6454] associated with the client. The server MUST send a "web_accepted_origins" transport parameter which SHALL be one of the following:

- A value "*", indicating that any origin is accepted.
- A comma-separated list of accepted origins, serialized as described in Section 6 of [RFC6454].

In the latter case, the user agent MUST verify that one of the origins is identical (as defined in Section 5 of [RFC6454]) to the origin of the client; otherwise, it MUST abort the session establishment.

3.3. 0-RTT

QuicTransport provides applications with ability to use the 0-RTT feature described in [RFC8446] and [QUIC-TRANSPORT]. 0-RTT allows a client to send data before the TLS session is fully established. It provides a lower latency, but has the drawback of being vulnerable to replay attacks as a result. Since only the application can make the decision of whether some data is safe to send in that context, 0-RTT requires the client API to only send data over 0-RTT when specifically requested.

0-RTT support in QuicTransport is OPTIONAL, as it is in QUIC and TLS 1.3.

4. Streams

QuicTransport unidirectional and bidirectional streams are created by creating a QUIC stream of corresponding type. All other operations (read, write, close) are also mapped directly to the operations as

defined in [QUIC-TRANSPORT]. The QUIC stream IDs are the stream IDs that are exposed to the application.

5. Datagrams

QuicTransport uses the QUIC DATAGRAM frame [QUIC-DATAGRAM] to provide WebTransport datagrams. A QuicTransport endpoint MUST negotiate and support the DATAGRAM frame. The datagrams provided by the application are sent as-is. The datagram ID SHALL be absent.

The datagrams sent using QuicTransport MUST be subject to congestion control.

6. Transport Properties

QuicTransport supports most of WebTransport features as described in Table 1.

Property	Support
Stream independence	Always supported
Partial reliability	Always supported
Pooling support	Not supported
Connection mobility	Implementation-dependent

Table 1: Transport properties of QuicTransport

7. Security Considerations

QuicTransport satisfies all of the security requirements imposed by [OVERVIEW] on WebTransport protocols, thus providing a secure framework for client-server communication in cases when the the client is potentially untrusted.

QuicTransport uses QUIC with TLS, and as such, provides the full

range of security properties provided by TLS, including confidentiality, integrity and authentication of the server.

QUIC is a client-server protocol where a client cannot send data until either the handshake is complete or a previously established session is resumed. This ensures that the user agent will prevent the client from sending data to network endpoints that are not QuicTransport endpoints. Furthermore, the QuicTransport session can be immediately aborted by the server through a connection close or a stateless reset, causing the user agent to stop the traffic from the client. This provides a defense against potential denial-of-service attacks on the network by untrusted clients.

QUIC provides a congestion control mechanism [I-D.ietf-quic-recovery] that limits the rate at which the traffic is sent. This prevents potentially malicious clients from overloading the network.

QuicTransport prevents the WebTransport clients connecting to arbitrary non-Web servers through the use of ALPN. Unlike TLS over TCP, successfully ALPN negotiation is mandatory in QUIC. Thus, unless the server explicitly picks "wq" as the ALPN value, the TLS handshake will fail. It will also fail unless the "web_accepted_origins" is present.

QuicTransport uses a QUIC transport parameter to provide the user agent with an origin whitelist. The origin is not sent explicitly,

as TLS ClientHello messages are sent in cleartext; instead, the server provides the user agent with a whitelist of origins that are allowed to connect to it.

In order to avoid the use of QuicTransport, the user agents MUST NOT allow the clients to distinguish different connection errors before the correct ALPN is received from the server.

Since each instance of QuicTransport opens a new connection, a malicious client can cause resource exhaustion, both on the local system (through depleting file descriptor space or other per-connection resources) and on a given remote server. Because of that, the user agegts SHOULD limit the amount of simultaneous connections opened. The server MAY limit the amount of connections open by the same client.

8. IANA Considerations

8.1. ALPN Value Registration

The following entry is added to the "Application Layer Protocol Negotiation (ALPN) Protocol IDs" registry established by [RFC7301]:

The "wq" label identifies QUIC used as a protocol for WebTransport:

Protocol: QuicTransport

Identification Sequence: 0x77 0x71 ("wq")

Specification: This document

8.2. QUIC Transport Parameter Registration

The following entry is added to the "QUIC Transport Parameter Registry" registry established by [QUIC-TRANSPORT]:

The "web_accepted_origins" parameter allows the server to indicate origins that are permitted to connect to it:

Value: 0x????

Parameter Name: web_accepted_origins

Specification: This document

9. References

9.1. Normative References

[OVERVIEW]
Vasiliev, V., "The WebTransport Protocol Framework", draft-vvv-webtransport-overview-00 (work in progress).

[QUIC-DATAGRAM]
Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", draft-pauly-quic-datagram-latest (work in progress).

[QUIC-TRANSPORT]
Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-latest (work in progress).

[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>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[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>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

9.2. Informative References

[I-D.ietf-quic-http]
Bishop, M., "Hypertext Transfer Protocol Version 3 (HTTP/3)", draft-ietf-quic-http-20 (work in progress), April 2019.

[I-D.ietf-quic-recovery]

Iyengar, J. and I. Swett, "QUIC Loss Detection and Congestion Control", draft-ietf-quic-recovery-20 (work in progress), April 2019.

Author's Address

Victor Vasiliev
Google

Email: vasilvv@google.com