QUIC
Internet-Draft
Intended status: Experimental
Expires: October 8, 2020

M. Duke
F5 Networks, Inc.

April 6, 2020

QUIC Version Aliasing
draft-ietf-quic-version-aliasing-00

Abstract

The QUIC transport protocol [QUIC-TRANSPORT] preserves its future extensibility partly by specifying its version number. There will be a relatively small number of published version numbers for the foreseeable future. This document provides a method for clients and servers to negotiate the use of other version numbers in subsequent connections. If a sizeable subset of QUIC connections use this mechanism, this should prevent middlebox ossification around the current set of published version numbers and the contents of QUIC Initial packets.

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 October 8, 2020.

Copyright Notice

Copyright (c) 2020 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

The QUIC version number is critical to future extensibility of the protocol. Past experience with other protocols, such as TLS1.3 [RFC8446], shows that middleboxes might attempt to enforce that QUIC packets use versions known at the time the middlebox was implemented. This has a chilling effect on deploying experimental and standard versions on the internet.

Each version of QUIC has a "salt" [QUIC-TLS] that is used to derive the keys used to encrypt Initial packets. As each salt is published

in a standards document, any observer can decrypt these packets and inspect the contents, including a TLS Client Hello. A subsidiary mechanism like Encrypted SNI [ENCRYPTED-SNI] might protect some of the TLS fields inside a TLS Client Hello.

This document proposes "QUIC Version Aliasing," a standard way of servers advertising the availability of other versions inside the cryptographic protection of a QUIC handshake. These versions are syntactically identical to the QUIC version in which the communication takes place, but use a different salt. In subsequent communications, the client uses the new version number and encrypts its Initial packets with a key derived from the provided salt. These version numbers and salts are unique to the client.

If a large subset of QUIC traffic adopts his technique, middleboxes will be unable to enforce particular version numbers or policy based on Client Hello contents without incurring unacceptable penalties on users. This would simultaneously protect the protocol against ossification and improve its privacy properties.

1.1. Terminology

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 RFC 2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying significance described in RFC 2119.

A "syntax version" is a QUIC version that would be advertised in a QUIC version negotiation and conforms to a specification. Any aliased version corresponds to a syntax version in all its formats and behaviors, except for the version number field in long headers.

An "aliased version" is a version with a number generated in accordance with this document. Except for the version field in long headers, it conforms entirely to the specification of the syntax version.

2. Protocol Overview

When they instantiate a connection, servers select an alternate 32-bit version number for the next connection at random and securely derive a salt from that version number using a repeatable process. They communicate this using a transport parameter extension including the version, salt, and an expiration time for that value.

The next time a client connects to that server, if it is within the indicated expiration time, it MAY use the provided version number and encrypt its Initial Packets using a key derived from the provided salt. The server can reconstruct the salt from the requested version and proceed with the connection normally.

3. The Version Alias Transport Parameter

3.1. Version Number Generation

Servers MUST use a random process to generate version numbers. This version number MUST NOT correspond to a QUIC version the server advertises in QUIC Version Negotiation packets.

Servers MAY encode the syntax version as long as this information is cryptographically protected. For example, a server advertises support for QUIC version 1 and QUIC version 2 in Version Negotiation packets, each corresponding to a particular packet syntax. In a Version 1 connection, it might provide an aliased version in the transport parameter, 0x45f3213b, that encodes the fact the syntax version is 1. When the client initiates a connection using version 0x45f3213b, the server knows the Initial Packet is formatted in accordance with QUIC version 1. A subsequent aliased version provided in the transport parameters would also encode version 1, even though this is sent in a connection ostensibly of version 0x45f3213b.

Servers MUST NOT use client-controlled information (e.g. the client IP address) in the random process, see Section 7.3.

Servers MUST NOT advertise these versions in QUIC Version Negotiation packets.

If multiple servers represent the same entity behind a load balancer, all such servers SHOULD have a common configuration for how to encode

and extract syntax version to use. They MUST NOT generate version numbers that any of them would advertise in a Version Negotiation Packet.

3.2. Salt Generation

The salt is an opaque 20-octet field. It is used to generate a Initial connection keys using the process described in {QUIC-TLS}.

Servers MUST generate the SALT using a cryptographic method that uses the version number and only server state that is persistent across connections. That is, servers MUST implement a method that it can repeat deterministically at a later time to derive the salt from the incoming version number. It MUST NOT use client controlled information other than the version number; for example, the client's IP address and port.

3.3. Expiration Time

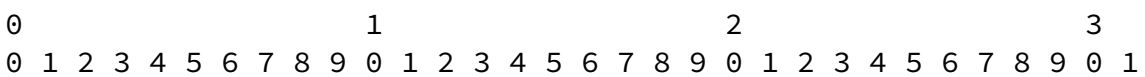
Servers should select an expiration time in seconds, measured from the instant the transport parameter is first sent. This time SHOULD be less than the time until the server expects to support new QUIC versions, rotate the keys used to encode information in the version number, or rotate the keys use in salt generation.

Furthermore, the expiration time SHOULD be short enough to frustrate a seed polling attack Section 7.3.

Conversely, an extremely short expiration time will often force the client to use standard QUIC version numbers and salts.

3.4. Format

This document defines a new transport parameter extension for QUIC with identifier 0x5641. The contents of the value field are indicated below.



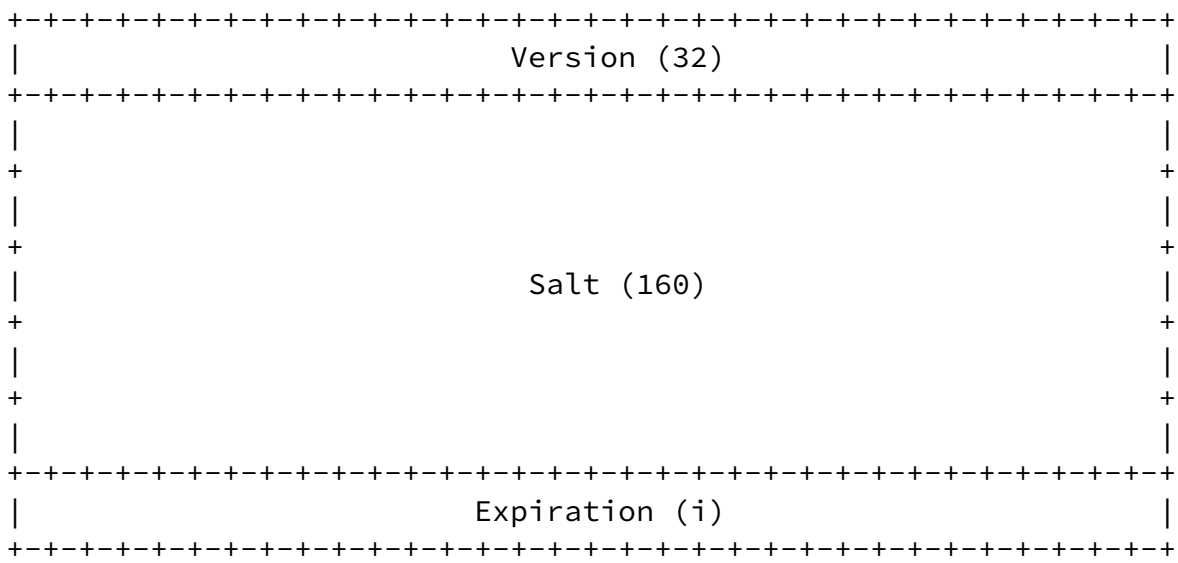


Figure 1: Version Alias Transport Parameter value

The definition of the fields is described above. Note that the "Expiration" field is in seconds, and its length is encoded using the Variable Length Integer encoding from Section 16 of [QUIC-TRANSPORT].

4. Client Behavior

When a client receives the Version Alias Transport Parameter, it MAY cache the version number, salt, and the expiration of this value. It MAY use the version number in a subsequent connection and compute the initial keys using the provided salt.

Clients SHOULD NOT attempt to use the provided version number and salt after the provided Expiration time has elapsed.

Clients SHOULD NOT use the provided version number or salt in more than one connection, particularly if its IP address has changed between two connection attempts. Using a consistent version number can link the client across IP address changes.

Clients MUST use the same syntax version to format the Initial Packet as the syntax version used in the connection that provided the aliased version.

If the response to an Initial packet using the provided version is a Version Negotiation Packet, the client SHOULD cease attempting to use that version and salt to the server unless it later determines that the packet was the result of a version downgrade, see Section 7.1.

5. Server Actions on Non-standard Version Numbers

When a server receives an Initial Packet with an unknown version number, it SHOULD send a Version Negotiation Packet if it is specifically configured not to generate that version number at random. Otherwise, it derives a salt from the version number using the algorithm and inputs it uses to generate salts to put in transport parameters.

If the syntax version was encoded in the version number, the server extracts it so that it can properly parse the packet. If not, it can try trial parsing of the packet for each syntax version it supports.

If the computed seed results in a packet that fails authentication, or the encoded syntax version is not supported at the server, or trial parsing fails for all supported versions, the server SHOULD send a Version Negotiation Packet.

Servers SHOULD provide a new Version Alias transport parameter, with a new version number and salt, each time a client connects, to reduce linkability for the client. However, issuing version numbers to a client SHOULD be rate-limited to mitigate the seed polling attack Section 7.3.

6. Considerations for Retry Packets

QUIC Retry packets reduce the load on servers during periods of stress by forcing the client to prove it possesses the IP address before the server decrypts any Initial Packets or establishing any connection state. Version aliasing substantially complicates the process.

If a server has to send a Retry packet, the required format is ambiguous without understanding which syntax version to use. If all supported syntax versions use the same Retry format, it simply uses that format with the provided version number.

If the supported syntax versions use different Retry formats, the server MUST either extract the syntax version from the version field and format the Retry accordingly using the aliased version number, or it MUST send a valid Retry packet for each supported version using the syntax version number instead of the aliased version number. It MUST NOT do both.

The Retry integrity Tag of a Retry Packet for an aliased version uses the procedure in Section 5.8 of [QUIC-TLS]. However, the secret key K uses the first 16 octets of the aliased salt instead of the key provided in the specification.

Clients MUST accept Retry packets that contain either the aliased version or syntax version. It MUST ignore Retry packets with other syntax versions. It it receives Retry packets with both the aliased version and the correct syntax version, it MUST discard the second one it receives in accordance with section 17.2.5 of [QUIC-TRANSPORT] unless the other one failed integrity validation.

After a client receives a Retry, it sends a new Initial Packet with the provided Retry token. It MAY use the aliased version and salt or the syntax version and salt, regardless of which type of Retry it received. Note that if the server is not able to generate the correct salt for an aliased version due to lost keys or other errors, this might result in a Version Negotiation packet, which violates the usual order of server responses (QUIC servers would normally send Version Negotiation before Retry).

Clients that receive a Version Negotiation packet in response to an Initial with a valid Retry token MAY interpret this to mean that the server can no longer process the aliased version. They can retry the connection with a syntax version number, but see Section 7.1. These MUST include the Retry token so that the client can verify that the Retry was authentic.

7. Security and Privacy Considerations

This document intends to improve the existing security and privacy properties of QUIC by dramatically improving the secrecy of QUIC Initial Packets. However, there are new attacks against this mechanism.

7.1. Version Downgrade

A countermeasure against version aliasing is the downgrade attack. Middleboxes may drop a packet containing a random version and imitate the server's failure to correctly process it. Clients and servers MUST implement the parts of [QUIC-VERSION-NEGOTIATION] relevant to downgrade detection.

Note that downgrade detection only works after receiving a response from the server. If a client immediately responds to a Version Negotiation Packet with an Initial Packet with a syntax version number, it will have exposed its request in a format readable to observers before it discovers if the Version Negotiation Packet is authentic. A client SHOULD wait for an interval to see if a valid response comes from the server before assuming the version negotiation is valid. The client MAY also alter its Initial Packet (e.g., its ALPN field) to sanitize sensitive information and obtain another aliased version before proceeding with its true request.

Servers that support version aliasing SHOULD be liberal about the Initial Packet formats they receive, keeping the connection open long enough to deliver their transport parameters, to support this mechanism.

7.2. Increased Linkability

As each version number is theoretically unique to each client, if a client uses one twice, those two connections are extremely likely to be from the same host. If the client has changed IP address, this is a significant increase in linkability relative to QUIC with a standard version numbers.

7.3. Seed Polling Attack

Observers that wish to decode Initial Packets might open a large number of connections to the server in an effort to obtain a large portion of the mapping of version numbers to salts to a server. While storage-intensive, this attack could increase the probability that at least some version-aliased connections are observable. There are two mitigations servers can execute against this attack:

- rate-limit transport parameters sent to a particular client; and/or
- set a low expiration time to reduce the lifetime of the attacker's database.

Segmenting the version number space based on client information, i.e. using only a subset of version numbers for a certain IP address range, would significantly amplify an attack. Observers will generally be on the path to the client and be able to mimic having an identical IP address. Segmentation in this way would dramatically reduce the search space for attackers. Thus, servers are prohibited from using these mechanisms.

7.4. Increased Processing of Garbage UDP Packets

As QUIC shares the UDP protocol number with other UDP applications, in some deployments it may be possible for traffic intended for other UDP applications to arrive at a QUIC server endpoint. When servers support a finite set of version numbers, a valid version number field is a strong indicator the packet is, in fact, QUIC. If the version number is invalid, a QUIC Version Negotiation is a low-cost response that triggers very early in packet processing.

However, a server that provides version aliasing is prepared to accept almost any version number. As a result, many more sufficiently sized UDP payloads with the first bit set to '1' are potential QUIC Initial Packets that require generation of a salt, some initial connection state, and a decryption operation.

While not a more potent attack then simply sending valid Initial Packets, servers may have to provision additional resources to address this possibility.

7.5. Increased Retry Overhead

This document requires two small cryptographic operations to build a Retry packet instead of one, placing more load on servers when already under load.

8. IANA Considerations

This draft chooses a transport parameter (0x5641) to minimize the risk of collision. IANA should assign a permanent value from the QUIC Transport Parameter Registry.

9. References

9.1. Normative References

[ENCRYPTED-SNI]
Rescorla, E., Ed., Oku, K., Ed., Sullivan, N., Ed., and C. Wood, Ed., "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-latest (work in progress).

[QUIC-TLS]
Thomson, M., Ed. and S. Turner, Ed., "Using Transport Layer Security (TLS) to Secure QUIC", draft-ietf-quic-tls-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 (work in progress).

[QUIC-VERSION-NEGOTIATION]
Schinazi, D., Ed. and E. Rescorla, Ed., "Compatible Version Negotiation for QUIC", draft-ietf-quic-version-negotiation-latest (work in progress).

9.2. Informative 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>.

[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>.

Appendix A. Acknowledgments

Marten Seemann was the original progenitor of the version aliasing approach.

Appendix B. Change Log

RFC Editor's Note: Please remove this section prior to publication of a final version of this document.

Author's Address

Martin Duke
F5 Networks, Inc.

Email: martin.h.duke@gmail.com