Workgroup: QUIC
Internet-Draft:
draft-duke-quic-version-aliasing-03
Published: 30 October 2020
Intended Status: Experimental
Expires: 3 May 2021
Authors: M. Duke
F5 Networks, Inc.

QUIC Version Aliasing

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 and encrypts Initial Packets using secret keys instead of standard ones. 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, as well as improving the protocol's privacy properties.

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 3 May 2021.

Copyright Notice

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 for servers to advertise 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 "standard 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 standard 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 standard version.

2. Protocol Overview

When they instantiate a connection, servers select an alternate 32-bit version number, and optionally an initial token extension, for the next connection at random and securely derive a salt and Packet Length Offset from those values using a repeatable process. They communicate this using a transport parameter extension including the

version, initial token extension, salt, Packet Length Offset, and an expiration time for that value.

If a client next connects to that server 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. It adds the Packet Length Offset to the true packet length when encoding it in the long header. If the server provided an Initial Token Extension, the client puts it in the Initial Packet token field. If there is another token the client wishes to include, it appends the Initial Token Extension to that token. The server can reconstruct the salt and Packet Length Offset from the requested version and token, and proceed with the connection normally.

The Packet Length Offset is provides a low-cost way for the server to verify it can derive a valid salt from the inputs without trial decryption. This has important security implications, as described in Section 7.2.

When generating a salt and Packet Length Offset, servers can choose between doing so randomly and storing the mapping, or using a cryptographic process to transform the aliased version number and token extension into the salt. The two options provide a simple tradeoff between computational complexity and storage requirements.

Note that clients and servers MUST implement [QUIC-VERSION-NEGOTIATION] to use this specification. Therefore, servers list supported versions in Version Negotiation Packets. Both clients and servers list supported versions in Version Negotiation Transport Parameters.

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 or transport parameters. Servers SHOULD also exclude version numbers used in known specifications or experiments to avoid confusion at clients, whether or not they have plans to support those specifications.

Servers MUST NOT use client-controlled information (e.g. the client IP address) in the random process, see Section 7.4.

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

3.2. Initial Token Extension (ITE) Generation

Servers SHOULD generate an Initial Token Extension (ITE) to provide additional entropy in salt generation. Two clients that receive the same version number but different extensions will not be able to decode each other's Initial Packets.

Servers MAY choose any length that will allow client Initial Packets to fit within the minimum QUIC packet size of 1200 octets. A four-octet extension is RECOMMENDED. The ITE MUST appear to be random to observers.

If a server supports multiple standard versions, it MUST either encode the standard version of the current connection in the ITE or store it in a lookup table.

If the server chooses to encode the standard version, it MUST be cryptographically protected.

Encoded standard versions MUST be robust to false positives. That is, if decoded with a new key, the version encoding must return as invalid, rather than an incorrect value.

Alternatively, servers MAY store a mapping of unexpired aliased versions and ITEs to standard versions. This mapping SHOULD NOT create observable patterns, e.g. one plaintext bit indicates if the standard version is 1 or 2.

The server MUST be able to distinguish ITEs from Resumption and Retry tokens in incoming Initial Packets that contain an aliased version number. As the server controls the lengths and encoding of each, there are many ways to guarantee this.

3.3. Salt and Packet Length Offset Generation

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

The Packet Length Offset is a 64-bit unsigned integer with a maximum value of 2^62 - 1. Clients MUST ignore a transport parameter with a value that exceeds this limit.

To reduce header overhead, servers MAY consistently use a Packet Length Offset of zero if and only if it either (1) never sends Retry packets, or (2) can guarantee, through the use of persistent storage or other means, that it will never lose the cryptographic state required to generate the salt before the promised expiration time. Section 7.2 describes the implications if it uses zero without meeting these conditions.

Servers MUST either generate a random salt and Packet Length Offset and store a mapping of aliased version and ITE to salt and offset, or generate the salt and offset using a cryptographic method that uses the version number, ITE, and only server state that is persistent across connections.

If the latter, servers MUST implement a method that it can repeat deterministically at a later time to derive the salt and offset from the incoming version number and ITE. It MUST NOT use client controlled information other than the version number and ITE; for example, the client's IP address and port.

3.4. 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 used in salt generation.

Furthermore, the expiration time SHOULD be short enough to frustrate a salt polling attack ({salt-polling}})

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

3.5. Format

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

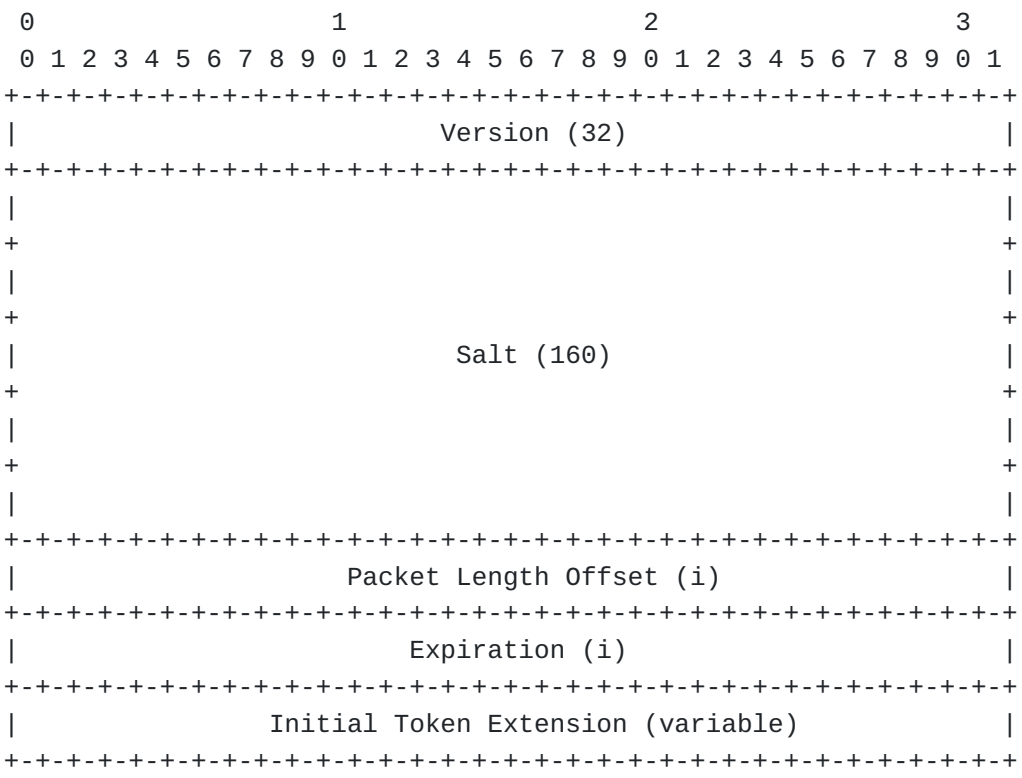

Figure 1: Version Alias Transport Parameter value

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].

The Packet Length Offset is also encoded as a Variable Length Integer.

Clients can determine the length of the Initial Token Extension by subtracting known and encoded field lengths from the overall transport parameter length.

3.6. Multiple Servers for One Domain

If multiple servers serve the same entity behind a load balancer, all such servers SHOULD either have a common configuration for encoding standard versions and computing salts, or share a common database of mappings. They MUST NOT generate version numbers that any of them would advertise in a Version Negotiation Packet or Transport Parameter.

4. Client Behavior

When a client receives the Version Alias Transport Parameter, it MAY cache the version number, ITE, salt, Packet Length Offset, and the expiration of these values. It MAY use the version number and ITE in

a subsequent connection and compute the initial keys using the provided salt.

Clients MUST NOT advertise aliased versions in the Version Negotiation Transport Parameter unless they support a standard version with the same number. Including that number signals support for the standard version, not the aliased version.

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

Clients MAY decline to use the provided version number or salt in more than one connection. It SHOULD do so if its IP address has changed between two connection attempts. Using a consistent version number can link the client across connection attempts.

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

If the server provided an ITE, the client MUST append it to any Initial Packet token it is including from a Retry packet or NEW_TOKEN frame, if it is using the associated aliased version. If there is no such token, it simply includes the ITE as the entire token.

The QUIC Token Length field MUST include the length of both any Retry or NEW_TOKEN token and the ITE.

The Length fields of all Initial, Handshake, and 0-RTT packets in the connection are set to the value described in [QUIC-TRANSPORT] plus the provided Packet Length Offset, modulo 2^62.

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.

If a client receives an aliased version number that matches a standard version that the client supports, it SHOULD assume the server does not support the standard version and MUST use aliased version behaviors in any connection with the server using that version number.

If a client receives a Version Negotiation packet or Version Negotiation transport parameter advertising a version number the server previously sent as an aliased version, and the client verifies any Version Negotiation Packet is not a Version Downgrade attack (Section 7.1), it MUST discard the aliased version number,

ITE, packet length offset, and salt and not use it in future connections.

5. Server Actions on Aliased Version Numbers

When a server receives an Initial Packet with an unsupported version number, it SHOULD send a Version Negotiation Packet if it is specifically configured not to generate that version number at random.

Otherwise, it extracts the ITE, if any, and either looks up the corresponding salt in its database or computes it using the technique originally used to derive the salt from the version number and ITE.

The server similarly obtains the Packet Length Offset and subtracts it from the provided Length field, modulo 2^62. If the resulting value is larger than the entire UDP datagram, the server discards the packet and SHOULD send a Version Negotiation Packet.

If the server supports multiple standard versions, it uses the standard version extracted by the ITE or stored in the mapping to parse the decrypted packet.

In all packets with long headers, the server uses the aliased version number and adds the Packet Length Offset to the length field.

In the extremely unlikely event that the Packet Length Offset resulted in a legal value but the salt is incorrect, the packet may fail authentication. If so, or the encoded standard version is not supported at the server, the server SHOULD send a Version Negotiation Packet.

To reduce linkability for the client, servers SHOULD provide a new Version Alias transport parameter, with a new version number, ITE, salt, and Packet Length Offset, each time a client connects. However, issuing version numbers to a client SHOULD be rate-limited to mitigate the salt polling attack Section 7.4.

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 establishes 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 standard version to use. If

all supported standard versions use the same Retry format, it simply uses that format with the client-provided version number.

If the supported standard versions use different Retry formats, the server obtains the standard version via lookup or decoding and formats a Retry containing the aliased version number accordingly.

Servers generate the Retry Integrity Tag of a Retry Packet using the procedure in Section 5.8 of [QUIC-TLS]. However, for aliased versions, the secret key K uses the first 16 octets of the aliased salt instead of the key provided in the specification.

Clients MUST ignore Retry packets that contain a QUIC version other than the version it used in its Initial Packet.

Servers MUST NOT reply to a packet with an incorrect Length field in its long header with a Retry packet; it SHOULD reply with Version Negotiation as described above.

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 are required to implement [QUIC-VERSION-NEGOTIATION] to detect downgrades.

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 standard 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 content they receive, keeping the connection open long enough to deliver their transport parameters, to support this mechanism.

7.2. Retry Injection

QUIC Version 1 Retry packets are spoofable, as they follow a fixed format, are sent in plaintext, and the integrity protection uses a widely known key. As a result, QUIC Version 1 has verification mechanisms in subsequent packets of the connection to validate the origin of the Retry.

Version aliasing largely frustrates this attack. As the integrity check key is derived from the secret salt, packets from attackers will fail their integrity check and the client will ignore them.

The Packet Length Offset is important in this framework. Without this mechanism, servers would have to perform trial decryption to verify the client was using the correct salt. As this does not occur before sending Retry Packets, servers would not detect disagreement on the salt beforehand and would send a Retry packet signed with a different salt than the client expects. Therefore, a client that received a Retry packet with an invalid integrity check would not be able to distinguish between the following possibilities:

- a Retry packet corrupted in the network, which should be ignored;
- a Retry packet generated by an attacker, which should be ignored; or
- a Retry packet from a server that lost its cryptographic state, meaning that further communication with aliased versions is impossible and the client should revert to using a standard version.

The Packet Length Offset introduces sufficient entropy to make the third possibility exceedingly unlikely.

7.3. Increased Linkability

As each version number and ITE is 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.

7.4. Salt Polling Attack

Observers that wish to decode Initial Packets might open a large number of connections to the server in an effort to obtain part of the mapping of version numbers and ITEs to salts for a server. While storage-intensive, this attack could increase the probability that

at least some version-aliased connections are observable. There are three mitigations servers can execute against this attack:

- use a longer ITE to increase the entropy of the salt,
- 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 this mechanism.

7.5. 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 and Packet Length Offset.

Note that a nonzero Packet Length Offset will allow the server to drop all but approximately 1 in every 2^49 packets, so trial decryption is unnecessary.

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

7.6. 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

[QUIC-TLS] Thomson, M., Ed. and S. Turner, Ed., "Using Transport Layer Security (TLS) to Secure QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-tls-latest, <https://tools.ietf.org/html/draft-ietf-quic-tls-latest>.

[QUIC-TRANSPORT] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", Work in Progress, Internet-Draft, draft-ietf-quic-transport, <https://tools.ietf.org/html/draft-ietf-quic-transport>.

[QUIC-VERSION-NEGOTIATION] Schinazi, D., Ed. and E. Rescorla, Ed., "Compatible Version Negotiation for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-version-negotiation-latest, <https://tools.ietf.org/html/draft-ietf-quic-version-negotiation-latest>.

9.2. Informative References

[ENCRYPTED-SNI] Rescorla, E., Ed., Oku, K., Ed., Sullivan, N., Ed., and C.A. Wood, Ed., "Encrypted Server Name Indication for TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-esni-latest, <https://tools.ietf.org/html/draft-ietf-tls-esni-latest>.

[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.

B.1. since draft-duke-quic-version-aliasing-01

*Fixed all references to "seed" where I meant "salt."

*Added the Packet Length Offset, which eliminates Retry Injection Attacks

B.2. since draft-duke-quic-version-aliasing-00

*Added "Initial Token Extensions" to increase salt entropy and make salt polling attacks impractical.

*Allowed servers to store a mapping of version number and ITE to salt instead.

*Made standard version encoding mandatory. This dramatically simplifies the new Retry logic and changes the security model.

*Added references to Version Negotiation Transport Parameters.

*Extensive readability edit.

Author's Address

Martin Duke
F5 Networks, Inc.

Email: martin.h.duke@gmail.com