Workgroup: QUIC
Internet-Draft:
draft-duke-quic-version-aliasing-07
Published: 25 October 2021
Intended Status: Experimental
Expires: 28 April 2022
Authors: M. Duke
F5 Networks, Inc.

QUIC Version Aliasing

Abstract

The QUIC transport protocol 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.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the mailing list (quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/.

Source for this draft and an issue tracker can be found at https://github.com/martinduke/quic-version-aliasing.

Status of This Memo

This Internet-Draft will expire on 28 April 2022.

Table of Contents

1. Introduction

The QUIC version number is critical to future extensibility of the protocol ([RFC9000]). 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 deters deployment of experimental and standard versions on the internet.

Each version of QUIC has a "salt" [RFC9001] 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 Client Hello [ECHO] 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. To be compatible with version aliasing, the first client packet in a standard version MUST encode the packet type and token as if it were a QUIC version 1 initial packet. That is:

- The most significant bit MUST be 1.
- The third and fourth most significant bits MUST be zero.
- The first field after the Source Connection ID MUST be a variable-length integer including the length of a token.
- The second field after the Destination Connection ID MUST be a field, with length indicated by the previous field, that contains opaque data generated by the server.
- There must be a variable-length integer that encodes the packet length, unprotected in the header.

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 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 8.5.

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.

2.1. Relationship to ECH and QUIC Protected Initials

The TLS Encrypted Client Hello [ECHO] shares some goals with this document. It encodes an "inner" encrypted Client Hello in a TLS extension in an "outer" Client Hello. The encryption uses asymmetric keys with the server's public key distributed via an out-of-band mechanism like DNS. The inner Client Hello contains any privacy-sensitive information and is only readable with the server's private key.

Significantly, unlike QUIC Version Aliasing, ECH can operate on the first connection between a client and server. However, from the second connection QUIC version aliasing provides additional benefits. It:

- greases QUIC header fields and packet formats;
- protects all of the TLS Client Hello and Server Hello;
- mitigates Retry injection attacks;
- does not require a mechanism to distribute the public key;
- uses smaller Client Hello messages, which might allow a larger 0RTT packet in the same datagram; and
- relies on computationally cheap symmetric encryption.

If ECH is operating in "Split Mode", where a client-facing server is using the SNI information to route to a backend server, the client-facing server MUST have the cryptographic context relevant to

version aliasing at the backend server to successfully extract the SNI for routing purposes. Furthermore, either all backend servers must share this context, or the client-facing server must trial decrypt the incoming packet with all possible derived salts.

Note that in the event of the server losing state, the two approaches have a similar fallback: ECH uses information in the outer Client Hello, and Version Aliasing requires a connection using a standard version. In either case, maintaining privacy requires the outer or standard version Client Hello to exclude privacy-sensitive information. However, ECH will allow confidential transmission of data in 1 RTT, while Version Aliasing requires 2 RTTs to resume. This mechanism is also relevant to mitigation of downgrade attacks (see Section 8.3).

Similarly, the QUIC Protected Initial [QUIC-PI] uses the ECH distribution mechanism to generate secure initial keys and Retry integrity tags. While still dependent on a key distribution system, asymmetric encryption, and relatively large Initial packets, it offers similar protection properties to Version Aliasing while still not greasing the version field.

A maximally privacy-protecting client might use Protected Initials for any connection attempts for which it does not have an unexpired aliased version, and QUIC version aliasing otherwise.

See also section 1.1 of [QUIC-PI] for further discussion of tradeoffs.

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 MAY use version numbers reserved for grease in Section 15.1 of [RFC9000], even though they might be advertised in Version Negotiation Packets.

Servers MUST NOT use client-controlled information (e.g. the client IP address) in the random process, see Section 8.7.

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 [RFC9001].

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 8.5 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 (Section 8.7)

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 provisional identifier 0x5641. The contents of the value field are indicated below.

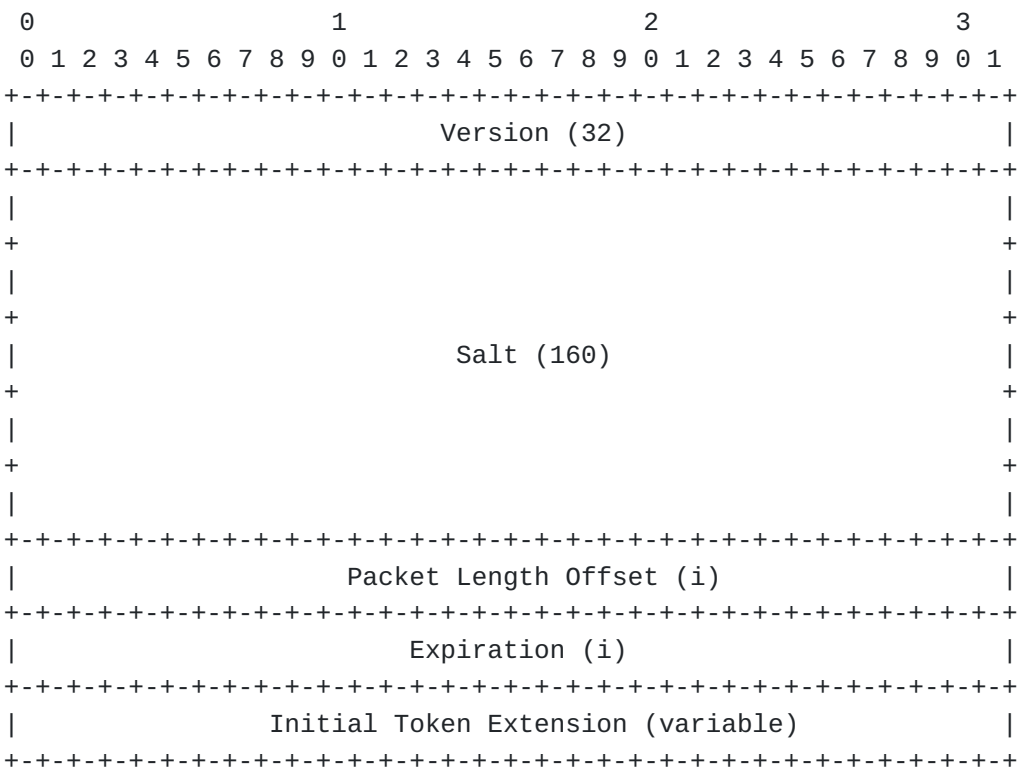


Figure 1: Version Alias Transport Parameter value

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 [RFC9000].

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

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

Note that servers that support version aliasing need not send the transport parameter on every connection. Therefore, a client MAY attempt to connect with an unexpired aliased version, even if in its most recent connection it did not receive the transport parameter.

Clients MAY remember the value in this transport parameter for future connections. Servers MUST either store the contents of the transport parameter, or preserve the state to compute the full contents based on what the client provides.

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.

3.7. Multiple Entities With One Load Balancer

If mutually mistrustful entities share the same IP address and port, incoming packets are usually routed by examining the SNI at a load balancer server that routes the traffic. This use case makes concealing the contents of the Client Initial especially attractive, as the IP address reveals less information. There are several solutions to solve this problem.

- All entities have a common crytographic context for deriving salts and Packet Length Offsets from the version number and ITE. This is straightforward but also increases the risk that the keys will leak to an attacker which could then decode Initial packets from point where the packets are observable. This is therefore NOT RECOMMENDED.

- Each entity has its own cryptographic context, shared with the load balancer. This requires the load balancer to trial decrypt each incoming Initial with each context. As there is no standard algorithm for encoding information in the Version and ITE, this involves synchronizing the method, not just the key material.

- Each entity reports its Version Aliasing Transport Parameters to the load balancer out-of-band.

- Each entity is assigned certain version numbers for use. This assignment SHOULD NOT follow observable patterns (e.g., assigning ranges to each entity), as this would allow observers to obtain the target server based on the version. The scheme SHOULD assign all available version numbers to maximize the entropy of the encoding.

Note that [ECHO] and [QUIC-PI] solve this problem elegantly by only holding the private key at the load balancer, which decodes the sensitive information on behalf of the back-end server.

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.

The Client MUST NOT use the contents of a Version Alias transport parameter if the handshake does not (1) later authenticate the

server name or (2) result in both endpoints computing the same 1-RTT keys. See Section 8.1. The authenticated server name MAY be a "public name" distributed as described in [ECHO] rather than the true target domain.

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.

When using an aliased version, the client MUST include a aliasing_parameters transport parameter in its Client Hello.

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 [RFC9000] plus the provided Packet Length Offset, modulo 2^62.

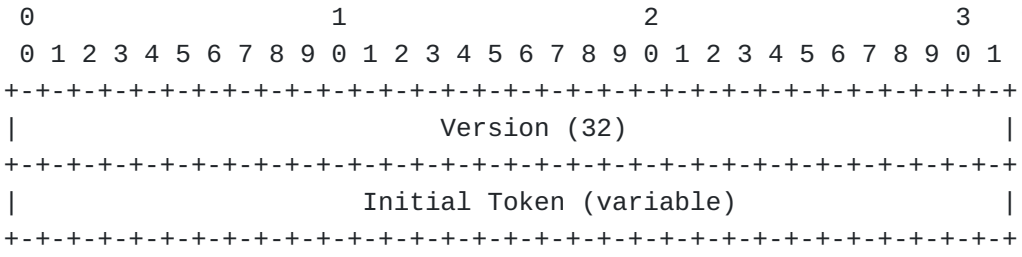
If a client receives an aliased version number that matches a standard version that the client supports, it SHOULD assume the server does not support the standard version and MUST use aliased version behaviors in any connection with the server using that version number.

If the response to an Initial packet using the provided version is a Version Negotiation Packet, the client SHOULD assume that the server no longer supports version aliasing and attempt to connect with one of the advertised versions (while observing the considerations in Section 8.3).

If the response to an Initial packet is a Bad Salt packet, the client follows the procedures in Section 6.

4.1. The aliasing_parameters Transport Parameter

This transport parameter has the following format. Its provisional type is 0x4150.



The Version field matches the one in the packet header.

The Initial Token field matches the Initial Token in the packet header, including any Retry token, NEW_TOKEN token, and Initial Token Extension. Its length is inferred from the specified length of the parameter.

The purpose of this parameter is to validate the contents of these header fields by including it in the TLS handshake transcript.

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 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 follow the procedure in Section 6. The server MAY apply further checks (e.g. against the minimum QUIC packet length) to further reduce the very small probability of a false positive.

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. The server should drop these packets in case this is the result of packet corruption along the path.

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 8.7 and MAY cease to clients that are consistently connecting with standard versions.

If there is no aliasing_parameters transport parameter, or the contents do not match the fields in the Initial header, the server MUST terminate the connection with a TRANSPORT_PARAMETER_ERROR.

6. Fallback

If the server has lost its encryption state, it may not be able to generate the correct salts from previously provided versions and ITEs. The fallback mechanism provides a means of recovering from this state while protecting against injection of messages by attackers.

When the packet length computation in Section 5 fails, it signals either that the packet has been corrupted in transit, or the client is using a transport parameter issued before a server failure. In either case, the server sends a Bad Salt packet.

6.1. Bad Salt Packets

The Bad Salt packet has a long header and a reserved version number, because it must not be confused with a legitimate packet in any standard version. They are not encrypted, not authenticated, and have the following format:

```
Bad Salt Packet {
    Header Form (1) = 1,
    Unused (7),
    Version (32) = TBD (provisional value = 0x56415641),
    Destination Connection ID Length (8),
    Destination Connection ID (0..2040),
    Source Connection ID Length (8),
    Source Connection ID (0..2040),
    Supported Version (32) ...,
    Integrity Tag (128),
}
```

Unused: The unused field is filled randomly by the sender and ignored on receipt.

Version: The version field is reserved for use by the Bad Salt packet.

Destination and Source Connection IDs and Lengths: These fields are copied from the client packet, with the source fields from the client packet written into the destination fields of the Bad Salt, and vice versa.

Supported Version: A list of standard QUIC version numbers which the server supports. The number of versions is inferred from the length of the datagram.

Integrity Tag: To compute the integrity tag, the server creates a pseudo-packet by contents of the entire client Initial UDP payload, including any coalesced packets, with the Bad Salt packet:

```
Bad Salt Pseudo-Packet {
    Client UDP Payload (9600..),
    Header Form (1) = 1,
    Unused (7),
    Version (32) = TBD (provisional value = 0x56415641),
    Destination Connection ID Length (8),
    Destination Connection ID (0..2040),
    Source Connection ID Length (8),
    Source Connection ID (0..2040),
    Supported Version (32) ...,
}
```

In a process similar to the Retry Integrity Tag, the Bad Salt Integrity Tag is computed as the output of AEAD_AES_128_GCM with the following inputs:

*The secret key, K, is 0xbe0c690b9f66575a1d766b54e368c84e.

*The nonce, N, is 0x461599d35d632bf2239825bb.

*The plaintext, P, is empty.

*The associated data, A, is the Bad Salt pseudo-packet.

These values are derived using HKDF-Expand-Label from the secret 0x767fedaff519a2aad117d8fd3ce0a04178ed205ab0d43425723e436853c4b3e2 and labels "quicva key" and "quicva iv".

The integrity tag serves to validate the integrity of both the Bad Salt packet itself and the Initial packet that triggered it.

6.2. Client Response to Bad Salt

Upon receipt of a Bad Salt packet, the client SHOULD wait for a Probe Timeout (PTO) to check if the Bad Salt packet was injected by an attacker, and a valid response arrives from the actual server.

After waiting, the client checks the Integrity Tag using its record of the Initial it sent. If this fails, the client SHOULD assume packet corruption and resend the Initial packet.

If the verification succeeds, the client SHOULD attempt to connect with one of the listed standard versions. It SHOULD observe the privacy considerations in Section 8.2. It MUST include a version_aliaising Transport Parameter in the Client Hello, that enumerates the aliased version and parameters it just tried to connect with.

Once it sends this transport parameter, the client MUST NOT attempt to connect with that aliased version again.

The original Client Initial is not part of the new connection. Therefore, the Connection IDs can change, and the original client hello is not part of the transcript for TLS key derivation.

Note that the client never sends this transport parameter with an aliased version. A server that receives such a packet MUST terminate the connection with a TRANSPORT_PARAMETER_ERROR.

6.3. Server Response to version_aliasing Transport Parameter

A client version_aliasing transport parameter tells the server that the client received a Bad Salt packet. The server checks if could have generated that version aliasing transport parameter given its current configuration; i.e. if using the version and ITE as inputs results in the same salt and Packet Length Offset.

If the salt or Packet Length Offset are invalid, the server SHOULD continue with the connection and SHOULD issue a new version_aliasing transport parameter.

If the salt and Packet Length Offset are valid, the server MUST terminate the connection with the error code INVALID_BAD_SALT.

7. 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 [RFC9001]. 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 Bad Salt as described above.

8. 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.

8.1. Endpoint Impersonation

An on-path attacker might respond to an Initial Packet with a standard version with a Version Aliasing Transport Parameter that then caused the client to reveal sensitive information in a subsequent Initial.

As described in Section 4, clients cannot use the contents of a Version Aliasing transport parameter until they have authenticated the source as a trusted domain, and have verified that the 1RTT key derivation is identical at both endpoints.

8.2. First-Connection Privacy

As version aliasing requires one connection over a standard QUIC version to acquire initial state, this initial connection leaks some information about the true target.

The client MAY alter its Initial Packet to sanitize sensitive information and obtain another aliased version before proceeding with its true request. However, the client Initial must lead to the authentication of a domain name the client trusts to provide accurate Version Aliasing information (possibly the public_name from an Encrypted Client Hello configuration from [ECHO]). Advice for the Outer ClientHello in Section 10.5 of [ECHO] applies here.

Endpoints are encouraged to instead use [ECHO] or [QUIC-PI] to increase privacy on the first connection between a client and server.

8.3. Forcing Downgrade

An attacker can attempt to force a client to send an Initial that uses a standard version by injecting a Version Negotiation packet (which implies the server no longer supports aliasing) or a Bad Salt packet (which implies the server has a new cryptographic context).

The weak form of this attack observes the Initial and injects the Version Negotiation or Bad Salt packet, but cannot drop the Initial. To counteract this, a client SHOULD NOT respond to these packets until they have waited for Probe Timeout (PTO) for a valid server Initial to arrive.

The strong form features an attacker that can drop Initial packets. In this case, the client can either abandon the connection attempt or connect with an standard version.

If it connects with a standard version, it should consider the privacy advice in Section 8.2.

Furthermore, if it received a Bad Salt packet, the client sends a Version Aliasing transport parameter to detect the downgrade attack, and the server will terminate the connection if the Bad Salt packet was an attack.

If the client received a Version Negotiation packet, it MUST implement a downgrade detection mechanism such as [I-D.ietf-quic-version-negotiation] or abandon the connection attempt. If it subsequent detects a downgrade detection, or discovers that the server does not support the same mechanism, it terminates the connection attempt.

8.4. Initial Packet Injection

QUIC version 1 handshakes are vulnerable to DoS from observers for the short interval that endpoints keep Initial keys (usually ~1.5 RTTS), since Initial Packets are not authenticated. With version aliasing, attackers do not have the necessary keys to launch such an attack.

8.5. 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.

8.6. 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.

8.7. Salt Polling

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.

8.8. Server Fingerprinting

The server chooses its own ITE length, and the length of this ITE is likely to be discoverable to an observer. Therefore, the destination server of a client Initial packet might be decipherable with an ITE length along with other observables. A four-octet ITE is RECOMMENDED. Deviations from this value should be carefully considered in light of this property.

Servers with acute needs for higher or lower entropy than provided by a four- octet ITE are RECOMMENDED to converge on common lengths to reduce the uniqueness of their signatures.

8.9. 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 computation 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.

8.10. 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.

8.11. Request Forgery

Section 21.4 of [RFC9000] describes the request forgery attack, where a QUIC endpoint can cause its peer to deliver packets to a victim with specific content.

Version aliasing allows the server to specify the contents of the version field and part of the token field in Initial packets sent by the client, potentially increasing the potency of this attack.

9. IANA Considerations

9.1. QUIC Version Registry

This document request that IANA add the following entry to the QUIC version registry:

Value: TBD

Status: permanent

Specification: This document

Change Controller: IETF

Contact: QUIC WG

9.2. QUIC Transport Parameter Registry

This document requests that IANA add the following entries to the QUIC Transport Parameters Registry:

Value	Parameter Name	Specification
TBD	Version Aliasing	This Document
TBD	aliasing_parameters	This Document

Table 1

9.3. QUIC Transport Error Codes Registry

This document requests that IANA add the following entry to the QUIC Transport Error Codes registry:

Value: TBD (provisional: 0x4942)

Code: INVALID_BAD_SALT

10. References

10.1. Normative References

[I-D.ietf-quic-version-negotiation] Schinazi, D. and E. Rescorla, "Compatible Version Negotiation for QUIC", Work in Progress, Internet-Draft, draft-ietf-quic-version-negotiation-04, 26 May 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-quic-version-negotiation-04>.

[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/rfc/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/rfc/rfc9001>.

10.2. Informative References

[ECHO] Rescorla, E., Oku, K., Sullivan, N., and C. A. Wood, "TLS Encrypted Client Hello", Work in Progress, Internet-Draft, draft-ietf-tls-esni-13, 12 August 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-esni-13>.

[QUIC-PI] Duke, M. and D. Schinazi, "Protected QUIC Initial Packets", Work in Progress, Internet-Draft, draft-duke-quic-protected-initial-02, 13 May 2021, <https://datatracker.ietf.org/doc/html/draft-duke-quic-protected-initial-02>.

[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/rfc/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/rfc/rfc8446>.

Appendix A. Acknowledgments

Marten Seemann was the original creator 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-05

*Revised security considerations

*Discussed multiple SNIs behind one load balancer

*Removed VN from the fallback mechanism

B.2. since draft-duke-quic-version-aliasing-04

*Relationship with Encrypted Client Hello (ECH) and QUIC Protected Initials

*Corrected statement about version negotiation

B.3. since draft-duke-quic-version-aliasing-03

*Discussed request forgery attacks

B.4. since draft-duke-quic-version-aliasing-02

*Specified 0RTT status of the transport parameter

B.5. 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.6. 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