Network Working Group
Internet-Draft
Intended status: Informational
Expires: 14 September 2026

J. Chen
Tsinghua University

13 March 2026

# Mitigating Logical Vulnerabilities in QUIC Implementations
draft-chen-quic-logical-vuln-mitigations-00

## Abstract

This document describes protocol and implementation practices for mitigating logical vulnerabilities in QUIC implementations. It focuses on denial-of-service and correctness failures caused by unbounded resource retention, unsafe state transitions, and insufficiently defensive handling of adversarial but parseable protocol inputs.

The document provides actionable guidance for protocol designers, implementers, and operators. It proposes concrete resource bounds, defensive validation rules, queue-management practices, and state-machine invariants intended to reduce exposure to memory exhaustion, CPU exhaustion, queue blow-up, connection starvation, and crash-triggering state confusion in QUIC endpoints.

## 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 14 September 2026.

## Copyright Notice

Table of Contents



## 1. Introduction

QUIC provides low-latency connection establishment, encrypted transport, stream multiplexing, and path migration over UDP. The core transport is specified in [RFC9000], TLS integration in [RFC9001], and loss detection and congestion control in [RFC9002].

QUIC has broad deployment and therefore broad attack surface. While protocol analyses and implementation testing have historically focused on memory corruption and parser defects, recent work shows that QUIC endpoints are also vulnerable to _logical vulnerabilities_: failures induced by valid or near-valid protocol behaviors that exploit ambiguous requirements, missing limits, or brittle implementation assumptions. Wang et al. evaluated 16 widely used QUIC implementations and reported 14 previously unknown logical vulnerabilities affecting projects such as quiche, xquic, aioquic, picoquic, h2o, lsquic, and neqo [QUICLOGIC].

This document does not redefine QUIC. Instead, it provides a defensive robustness profile for QUIC endpoints. Unless explicitly stated otherwise, the requirements in this document are implementation robustness recommendations and do not update the wire protocol specified by [RFC9000], [RFC9001], or [RFC9002]. The goal is to preserve interoperability while making denial-of-service and state-confusion attacks materially harder to trigger.

## 2. Conventions and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [BCP14] when, and only when, they appear in all capitals, as shown here.

The term "logical vulnerability" refers to an implementation weakness that can be triggered without memory corruption and that results from unsafe protocol-state transitions, unbounded resource retention, insufficient input normalization, or inconsistent handling of malformed but parseable protocol events.

## 3. Threat Model and Scope

This document considers off-path and on-path attackers that can create new QUIC connections, send large numbers of frames, manipulate timing, withhold acknowledgments, or send frame sequences that are valid, ambiguous, duplicated, reordered, or otherwise adversarially chosen. The primary attacker objective is denial of service through memory exhaustion, CPU exhaustion, send-queue growth, connection starvation, or assertion failure.

The recommendations in this document target server and client implementations, though the operational risk is generally higher for publicly reachable servers. The document focuses on transport-layer behaviors and does not address application-layer misuse above QUIC except where application scheduling interacts with transport state.

# 4. Observed Vulnerability Classes

## 4.1. CRYPTO Flood

QUIC CRYPTO frames carry TLS handshake data. [RFC9000] requires endpoints to support buffering at least 4096 bytes of out-of-order CRYPTO data, and it explicitly notes that, because CRYPTO frames are not flow-controlled, a peer could force unbounded buffering if implementations do not apply limits. Wang et al. observed implementations that retained many out-of-order CRYPTO fragments in memory, resulting in memory exhaustion [QUICLOGIC].

## 4.2. PATH_CHALLENGE Flood

During path validation, a peer sends PATH_CHALLENGE frames and expects corresponding PATH_RESPONSE frames as described in [RFC9000]. Wang et al. observed implementations in which rapid arrival of PATH_CHALLENGE frames, combined with delayed or missing acknowledgment of corresponding responses, caused memory growth, buffer retention, or unbounded send-queue expansion [QUICLOGIC].

## 4.3. NEW_CONNECTION_ID Flood

QUIC connection migration relies on NEW_CONNECTION_ID and RETIRE_CONNECTION_ID frame exchanges. [RFC9000] includes requirements around active_connection_id_limit and recommends limiting the number of locally retired connection IDs that remain unacknowledged. Wang et al. observed implementations where repeated NEW_CONNECTION_ID processing and delayed retirement acknowledgment created unbounded RC-frame accumulation or send-queue growth [RFC9000] [QUICLOGIC].

## 4.4. Connection Hold-On Flood

Wang et al. reported a scheduling and lifecycle bug in which connection entries that were effectively closed or inactive still occupied positions in a traversal structure used to service active connections. Under attacker-induced connection churn, legitimate connections were starved even though CPU and memory usage appeared normal. This creates a "silent" denial of service that is visible in service availability rather than host resource alarms [QUICLOGIC].

## 4.5. Double Unregistered CID

Wang et al. also observed failures in connection ID retirement logic where an implementation could unregister or retire the same logical CID state twice under adversarial sequencing, ultimately triggering a crash condition [QUICLOGIC].

### 4.6. ACK Confusion

ACK frames represent received packet ranges. Wang et al. generated ACK frames containing overlapping ranges and observed divergence among implementations: some ignored them, some closed the connection, and at least one repeatedly mutated acknowledgment state for already acknowledged packets, causing CPU exhaustion or crash behavior [QUICLOGIC]. Robust handling of malformed or redundant ACK range encodings is therefore an interoperability and security concern.

## 5. Robustness Requirements for QUIC Endpoints

### 5.1. Explicit Resource Bounds

#### 5.1.1. CRYPTO Reassembly Bounds

An endpoint MUST maintain an explicit upper bound on buffered out-of-order CRYPTO data per connection. The implementation MUST apply the bound independently of allocator behavior or container growth policy.

This document RECOMMENDS a default limit of 512 KiB of buffered out-of-order CRYPTO data per connection, consistent with the concrete example suggested by Wang et al. [QUICLOGIC]. An implementation MAY use a different value, but any larger value SHOULD be justified by deployment needs such as unusually large certificate chains or delegated credentials.

If the configured bound is exceeded during the handshake, an endpoint SHOULD either temporarily expand the buffer only as needed to complete the handshake or close the connection with CRYPTO_BUFFER_EXCEEDED, consistent with [RFC9000]. After the handshake, an endpoint SHOULD prefer discarding excess CRYPTO data or terminating the connection rather than retaining unbounded state.

#### 5.1.2. Path Validation Bounds

An endpoint MUST bound the number of outstanding path-validation items it is willing to track per connection and per candidate path. This bound MUST apply to pending PATH_CHALLENGE state, queued PATH_RESPONSE state, retransmission bookkeeping, and any implementation-specific metadata derived from those frames.

This document RECOMMENDS accepting no more than 256 outstanding PATH_CHALLENGE-related transactions for a connection before path validation completes, after which the counter can be reset as suggested by Wang et al. [QUICLOGIC]. Endpoints SHOULD additionally apply per-path rate limits and SHOULD drop or coalesce redundant validation work when newer probes supersede older ones.

#### 5.1.3. CID Retirement and Issuance Bounds

Implementations MUST cap all internal state associated with NEW_CONNECTION_ID and RETIRE_CONNECTION_ID processing, including:

* active connection IDs, as bounded by active_connection_id_limit;
* locally retired but unacknowledged CIDs;
* queued RETIRE_CONNECTION_ID frames;
* per-path metadata associated with issued CIDs; and
* historical retirement bookkeeping used to prevent duplicate processing.

[RFC9000] already recommends that endpoints limit the number of locally retired connection IDs whose RETIRE_CONNECTION_ID frames have not been acknowledged and allow at least twice the peer's active_connection_id_limit for tracking. This document further RECOMMENDS an implementation-level hard cap of 256 CID retirements in progress before migration completes, in line with the example suggested by Wang et al. [RFC9000] [QUICLOGIC].

If the implementation reaches any retirement or issuance threshold, it SHOULD stop allocating additional per-CID state and SHOULD either reject the triggering input with a connection error or defer processing until existing retirement work is completed.

### 5.2. State-Machine Safety

#### 5.2.1. Closed or Inactive Connections

Closed, draining, or otherwise non-serviceable connections MUST NOT remain in scheduling structures in a way that can block forward progress for active connections. Implementations SHOULD ensure that connection iteration skips closed entries without terminating the scheduling loop for the remaining active set.

Implementations SHOULD enforce a separate lifecycle invariant: connection cleanup MUST eventually remove all inactive connection entries from traversal and lookup structures, even if application streams close before transport state is fully reclaimed.

#### 5.2.2. CID Retirement Idempotence

CID retirement and unregistration logic MUST be idempotent. Reprocessing a retirement event for the same CID sequence number MUST NOT free, unregister, erase, or otherwise invalidate the same underlying object twice.

Implementations SHOULD maintain explicit per-CID state markers such as issued, active, retiring, retired, and destroyed. A transition to retired or destroyed MUST be monotonic and MUST be checked before any destructive action is taken.

#### 5.2.3. ACK Range Normalization and Validation

Before mutating loss-recovery or acknowledgment state, an endpoint SHOULD validate that ACK ranges are well-formed, non-overlapping, and strictly descending according to the QUIC ACK encoding model. Receipt of an ACK frame whose reconstructed ranges overlap, repeat packet numbers in a way the implementation does not intentionally support, or underflow range arithmetic SHOULD be treated as a connection error of type FRAME_ENCODING_ERROR or PROTOCOL_VIOLATION.

An endpoint MUST NOT repeatedly mutate per-packet acknowledgment state for the same packet number solely because that packet appears multiple times in a malformed ACK frame. If the implementation chooses not to close the connection immediately, it MUST at minimum normalize duplicate ranges so that each acknowledged packet is processed at most once.

### 5.3. Queue Management and Backpressure

Every QUIC endpoint SHOULD enforce quotas on internal queues whose growth can be influenced by peer input, including send queues, retransmission queues, CRYPTO reassembly buffers, path-validation queues, ACK processing work queues, and CID-retirement queues.

Implementations SHOULD define admission-control behavior for each such queue. Once a queue reaches its configured limit, the endpoint SHOULD drop superseded work, coalesce equivalent work items, defer further processing, or close the connection. Blindly appending new work items is NOT RECOMMENDED.

Endpoints SHOULD also maintain per-connection and per-peer fairness controls so that one connection cannot indefinitely consume processing budget needed by others.

### 5.4. Implementation Guidance

Wang et al. observed that resource-consumption bugs were more common in implementations written in higher-level languages using dynamic containers such as lists, dictionaries, or vectors [QUICLOGIC]. This document therefore RECOMMENDS that implementations:

* attach explicit quotas to all attacker-influenced dynamic containers;

* separate protocol correctness from allocator growth behavior;

* instrument memory, queue depth, and per-connection state cardinality;

* use invariant checks in debug and fuzzing builds for CID and ACK state; and

* treat path validation, retirement, and reassembly structures as security-sensitive resources.

### 5.5. Continuous Differential and Adversarial Testing

The vulnerabilities motivating this document were discovered through adversarial black-box fuzzing and differential testing across implementations [QUICLOGIC]. QUIC stacks SHOULD be tested continuously with:

* long frame sequences rather than single-frame mutations;

* delayed, dropped, duplicated, and reordered frame interactions;

* resource-usage assertions, not only crash detection;

* cross-implementation differential checks for connection behavior; and

* invariants covering connection cleanup, ACK processing, and CID retirement.

## 6. Suggested Areas for Future QUIC Specification Clarification

Wang et al. argue that some implementation divergence stems from places where existing RFC text provides minimum behavior but not explicit upper bounds [QUICLOGIC]. Based on that observation, future QUIC maintenance or applicability documents should consider clarifying:

* the expectation that CRYPTO buffering is both mandatory and bounded;

* the expectation that path-validation state is bounded and rate-limited;

* the expectation that CID retirement tracking is finite and subject to backpressure;

* the handling of malformed or overlapping ACK ranges; and

* implementation obligations for cleanup of inactive connection state.

Such clarifications would improve interoperability by reducing the space of "technically plausible but operationally unsafe" implementation choices.

## 7. Security Considerations

This entire document is about security considerations. Logical vulnerabilities in QUIC can enable denial-of-service attacks without violating packet protection or exploiting memory corruption. The recommended mitigations reduce exposure to memory exhaustion, CPU exhaustion, queue blow-up, connection starvation, and crash-triggering state confusion.

None of the mitigations in this document eliminate the need for authentication, address validation, anti-amplification controls, or congestion control as defined by the base QUIC specifications. Instead, they complement those mechanisms by constraining state growth and hardening transport logic against adversarial but syntactically parseable inputs.

## 8. IANA Considerations

This document has no IANA actions.

## 9. Normative References

[BCP14] Best Current Practice 14,
<https://www.rfc-editor.org/info/bcp14>.
At the time of writing, this BCP comprises the following:

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>.

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

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

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

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

## 10. Informative References

[QUICLOGIC] Wang, K., Chen, J., Chen, P., Zhuge, J., Bai, J., and H. Duan, "Identifying Logical Vulnerabilities in QUIC Implementations", NDSS Symposium 2026, February 2026, <https://www.ndss-symposium.org/ndss-paper/identifying-logical-vulnerabilities-in-quic-implementations/>.

## Author’s Address

Jianjun Chen
Tsinghua University
China
Email: jianjun@tsinghua.edu.cn