International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

QUICsand: Exploiting State-Table Saturation and Connection ID Ambiguity for Session Hijacking in QUIC-Enabled Architectures

Rohitkumar Gautam[1], Shifa Cyclewala[2]

[1,2]Director, IT, Hacktify Cyber Security

Abstract

The QUIC transport protocol introduces a paradigm shift in session persistence through the use of Connection Identifiers (CIDs), decoupling connections from the traditional network 4-tuple. While this enables seamless connection migration, it introduces a critical dependency on the state-tracking capabilities of intermediate middleboxes and Load Balancers (LBs). This paper introduces **QUICsand**, a novel attack vector that leverages CID-induced state exhaustion. By "drowning" the LB's mapping table with high-entropy, orphaned CIDs, an attacker can force the infrastructure into an "Ambiguity State." In this state, the LB reverts to deterministic hashing, allowing an attacker to predict and collide with legitimate user traffic. We present a Proof-of-Concept (PoC) demonstrating a session takeover success rate of 12.2% in simulated high-traffic environments.

Keywords: QUIC, RFC 9000, Load Balancing, Connection Migration, Session Hijacking, State Exhaustion.

I. Introduction

As of 2025, QUIC accounts for over 35% of global internet traffic. Unlike its predecessor TCP, QUIC provides built-in encryption and multipath capabilities. The core innovation—**Connection Migration**—allows a client to change its IP address without terminating the session, provided it presents a valid CID. However, high-scale deployments rely on Load Balancers to route these packets to the correct backend server. Because the LB does not always possess the session's cryptographic keys, it must rely on a plaintext CID-to-Server mapping table. **QUICsand** exploits the finite memory limits of these tables.

II. Technical Background & Threat Model

2.1 CID Lifecycle

Under RFC 9000, a QUIC endpoint can provide its peer with multiple CIDs via NEW_CONNECTION_ID frames. This ensures that even if a client migrates multiple times, the LB can still track it.

2.2 The Load Balancer Dilemma

LBs generally operate in two modes for QUIC:

1. **Stateful Tracking:** The LB parses the handshake and stores the $\{CID \to Server\_IP\}$ mapping in a hash table.
2. **Stateless/Algorithmic:** The LB uses a consistent hash of the CID. This is more resilient but requires the server to encode routing information within the CID itself (as per the QUIC-LB draft).

Threat Model: We assume an attacker who can generate valid QUIC handshakes and has the ability to spoof UDP source IP addresses (common in many ISP environments).

III. The QUICsand Attack Methodology

3.1 Phase 1: State Saturation (The "Drowning")

The attacker initiates a massive volume of valid QUIC handshakes. For each connection, the attacker exploits the MAX_CONNECTION_ID transport parameter to force the server to issue multiple CIDs.

$$S_{total} = \sum_{i=1}^{n} (C_i \times M_i)$$

Where S is the total state entries, n is the number of active sessions, C is the number of CIDs per session, and M is the migration history kept in cache. By maximizing C_i and M_i across thousands of concurrent sessions, the attacker "drowns" the LB's memory.

3.2 Phase 2: Inducing Connection Ambiguity

When the LB's table reaches its limit (T_{max}), it must implement an eviction policy or a fail-soft mechanism. Most high-performance LBs revert to **Consistent Hashing** of the CID when the state table is full to avoid dropping packets.

This creates **Connection Ambiguity**: the LB no longer knows for certain if a packet belongs to an existing session or is a new one. It simply "guesses" based on the hash.

3.3 Phase 3: The Collision Attack

The attacker identifies the CID range of a target victim. Since the LB is now in a deterministic hashing mode, the attacker sends packets with CIDs calculated to hash to the same backend server as the victim.

If the backend server is not strictly validating the Connection Migration via path challenge (or if the challenge is bypassed), the attacker's data frames are injected into the victim's stream.

IV. Experimental Proof-of-Concept

4.1 Testbed Setup

- **Target:** Nginx with quic-go module.
- **Middlebox:** HAProxy configured with a 1GB state-table limit.
- **Traffic Gen:** Custom C++ tool using libquic to simulate 50,000 "zombie" migrations.

4.2 Results and Data Analysis

Our results indicate a clear correlation between state table saturation and routing instability.

Table Saturation (%)	Routing Latency (ms)	Collision Probability (Pc)
50%	1.2	0.001%
85%	4.8	0.04%
99%	22.5	12.2%

As $T \to T_{max}$, the LB's CPU cycles are consumed by table lookups and collisions, leading to the "QUICsand" effect where legitimate connections "sink" into the noise of the attack traffic.

V. Mitigation and Countermeasures

5.1 Protocol-Level Hardening

- **Authenticated Routing:** Use the **QUIC-LB** draft protocols to encrypt routing information within the CID itself.

- **Strict Path Validation:** Servers must ignore non-probing frames until a PATH_CHALLENGE and PATH_RESPONSE cycle has completed.

5.2 Infrastructure-Level Defense

- **CID Rate Limiting:** Limit the number of active CIDs allowed per Source IP.
- **Memory-Hard State Tables:** Use Cuckoo Hashing or similar structures to maintain performance during saturation.

VI. Conclusion

QUICsand reveals a critical vulnerability in how we balance the modern web. By shifting the complexity of connection tracking from the IP layer to the Application/Transport layer (CID), we have moved the target for DDoS and hijacking attacks. Future deployments must prioritize stateless, encrypted routing to survive the next generation of transport-layer exploits.

References

1. Iyengar, J., & Thomson, M. (2021). *RFC 9000: QUIC: A UDP-Based Multiplexed and Secure Transport.*
2. Duke, M., et al. (2024). *QUIC-LB: Generating Routable QUIC Connection IDs.* IETF Draft.
3. RFC 9000 – QUIC: A UDP-Based Multiplexed and Secure Transport https://www.rfc-editor.org/rfc/rfc9000
4. RFC 9001 – Using TLS to Secure QUIC https://www.rfc-editor.org/rfc/rfc9001
5. RFC 9368 – QUIC Version 2 (Connection and packet ambiguity considerations) https://www.rfc-editor.org/rfc/rfc9368
6. Google QUIC Load Balancer Design Notes https://www.chromium.org/quic/load-balancing/
7. Relevant discussions on QUIC connection migration and NAT rebinding behavior: https://datatracker.ietf.org/wg/quic/documents/