Retry, Retry, Retry

QUIC Interim, New York, September 2018

Martin Thomson

Connection ID Authentication

NEW_CONNECTION_ID is authenticated (good)

Handshake packets are sort-of authenticated (probably OK)

Initial and Retry packets aren't authenticated (hmm)

So, what if we can modify the connection ID?

Well, we can maybe get the receiver to interpret it as part of the packet number:

...	Destination Connection ID (Length ???)	PN Len (1-2 bit)	Packet Number (7/14/30 bit)	...

Modifying Initial packets

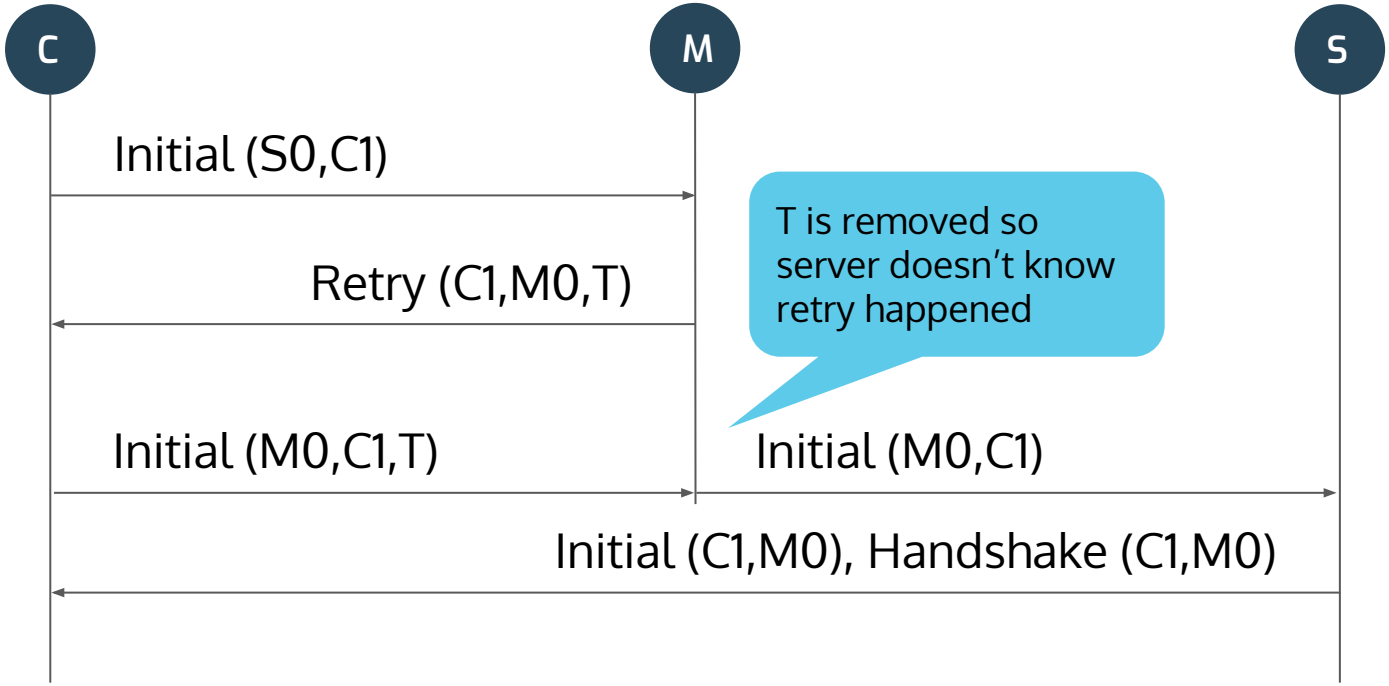
Initial from the client

- The Source Connection ID determines routing; client will drop the response if it is changed
- The Destination Connection ID determines keys; ditto

Initial from the server is right next to Handshake

- Maybe can change Source Connection ID, but would need to modify Handshake packets also; probably not doable
- (Not particularly satisfying answer here, more later)

Modifying connection ID with Retry



Credit: kazuho
Assumption: the choice of destination connection ID by the client affects the server choice of connection ID

Requirements

Unauthenticated Retry possible

- Some servers want to deploy boxes that screen for DoS and send Retry without close coordination

Authentication of Retry possible

- If Retry was sent by the server itself, or it has a closer relationship with the DoS mitigation box

No implicit or transitive authentication possible

The best way to authenticate this sort of thing is to take the data and mix it in the key derivation

- Checks are implicit: if you fail to account for the information the handshake fails
- You don't have to send anything twice

That relies on both peers remembering things, so no go here

Authenticating Retry

Simplest solution here is to add something to transport parameters

1. the entire sequence of Retry packets
   = probably too expensive
2. a hash of Retry packets
   = only works if the server knows the contents of the Retry, so no validation if Retry is sent by an independent DoS mitigation box or another server instance
3. an excerpt of Retry packets
   = requires that server help itself

Servers: self-service validation w/ excerpt

Client sends the first N octets of the token from Retry in its transport parameters

Servers that are stateful can use a random value and check that it is the right value

Otherwise, include a MAC over information that can be checked: the remainder of the Retry token, the connection IDs from the Retry, and the client address

Server can check the MAC and detect tampering (or not)

The MAC inputs need to have low P(collision) or copy-paste!

Clients: authenticate only connection ID

Proposal: Include the server's final choice of connection ID in the server transport parameters

Changes

```
    select (Handshake.msg_type) {
        case client_hello:
           QuicVersion initial_version;
           opaque retry_excerpt[32];

        case encrypted_extensions:
           QuicVersion negotiated_version;
           QuicVersion supported_versions<4..2^8-4>;
     };
     opaque final_connection_id<0..18>;
     TransportParameter parameters<22..2^16-1>;
} TransportParameters;
```

Details for later are size (32, less, more) and whether we require retry tokens to be 32 octets minimum to support this

Problem: Multiple Retries

The excerpt method doesn't work if the server sends multiple retries

It relies on the connection ID and token being taken from elsewhere in the packet - the client doesn't repeat earlier values

So...

Proposal: One Retry Only

TCP only offers one chance, this is a new feature in QUIC

No more rule about changing connection IDs on Retry

Easier and cheaper to authenticate Retry

Easier and cheaper to authenticate connection IDs

Cost is obvious: only one layer of DoS mitigation allowed