Idea: DNS over QUIC zone transfer over QUIC or TLS/TCP

Kazunori Fujiwara, JPRS
<fujiwara@jprs.co.jp>

DNS-OARC 2015 Fall Workshop

Last Update: 2015/10/02 0453 (UTC)

My expectations for QUIC

- I listened QUIC presentations at IETF-93 BarBoF

- TCP is weak to the man-in-the-middle attack
- Anyone can terminate TCP connections by sending TCP RESET packet
  - Interrupting zone transfer is easy
- QUIC may solve the problem
- Then, I submitted my abstract to this workshop

Submitted abstract

- The presentation discusses just an idea about DNS over QUIC and zone transfer over QUIC or TLS/TCP.
  - The third transport of DNS may be QUIC.
  - Both DNS and QUIC use UDP and port 53/UDP may be possible to share.
  - (If possible, implementation status will be reported, but it seems hard.)
    →QUIC is complicated and I couldn't implement
  - And zone transfers may be performed over QUIC or TLS/TCP transport with server certificate authentication.

QUIC (Quick UDP Internet Connection)

A UDP-Based Secure and Reliable Transport for HTTP/2

- Developed by Google and chromium.org
- Specs
  - draft-tsvwg-quic-protocol-01
  - draft-tsvwg-quic-loss-recovery-00
  - [QUIC-CRYPTO] http://goo.gl/jOvOQ5

  - Currently, not well-defined and many undescribed parts
  - Need to refer current code (at chromium.org)

QUIC: Characteristics

- QUIC=TCP+TLS(+HTTP2) functions on UDP
  - Does not require any changes in OS and network
  - Implemented as a library, userland only
- 1 QUIC session can handle multiple streams
  - Each stream frame can be immediately dispatched to that stream on arrival
- Any controls and messages are authenticated (and/or encrypted)
- Large connection ID (0,8,32,64 bits)
- Large sequence number (64 bits)
  - it may be used as a cryptographic nonce

Use cases of DNS over QUIC

- IETF dprive WG
  - DNS over QUIC is equivalent to DNS over TLS/DTLS
- Zone transfer over QUIC or TLS

Zone transfer over QUIC or TLS

- Zone transfer is not encrypted
- Zone transfer over TLS may be required to protect zone contents from man-in-the-middle attacks
- ACL may be changed to check certificates or their common names

Server:

certificate: “something.pem”

Zone:

name: “example.org”

request-xfer: 192.0.2.1 TSIGKEY CommonName

provide-xfer: 192.0.2.2 TSIGKEY CommonName

After reading QUIC specs

Section 5.5 of draft-tsvwg-quic-protocol-01

- “Caveat: PUBLIC_RESET packets that reset a connection are currently not authenticated.”
  - Attacker can generate PUBLIC_RESET packet
  - My expectation was disappointed
  - (it will be fixed, I believe)

QUIC: current implementation

- http://www.chromium.org/quic
  - Written by C++
- To build standalone test server and client,
  - Requires chromium source tree (about 1.3GB)
  - Requires special build system (ninja)
  - Limited information
  - Limited documentation: need to read the source code

QUIC test on FreeBSD 10.1

- I found NEWPORT request for FreeBSD
  - [Bug 202286] [NEW PORT] net/libquic-devel
  - Tarballs listed by distinfo changed after the request
    - fixed “distinfo” file
    - Huge tarballs: 372MB distfiles/libquic-devel/
  - It works (compile libquic and test server / client)
- Result of building libquic
  - Huge library: 138M /usr/local/lib/libquic.so.0
  - Many header files: 933 in /usr/local/include/quic
  - Normal size binaries
    - 1.9MB quic_simple_client
    - 3.3MB quic_simple_server

quic_simple_client

- Smallest compile command for a client
  - c++ -I/usr/local/include/quic -fno-rtti -std=gnu++11 quic_simple_client_bin.cc quic_in_memory_cache.cc quic_simple_client.cc quic_spdy_client_stream.cc synchronous_host_resolver.cc quic_client_session.cc quic_spdy_server_stream.cc -L/usr/local/lib -lquic
  - Non-installed header files (22) are required
    - base/metrics (4) base/posix (1) net/tools/balsa (3) net/tools/quic (14)
- Main part of the client
  - client.Initialize()
  - client.Connect()
  - client.SendRequestAndWaitForResponse()
- Blocking code may be easy

quic_simple_server

- Smallest compile command for a server
  - c++ -I/usr/local/include/quic -fno-rtti -std=gnu++11 quic_simple_server_bin.cc quic_simple_server.cc quic_server_session.cc quic_in_memory_cache.cc quic_spdy_server_stream.cc quic_simple_per_connection_packet_writer.cc quic_simple_server_packet_writer.cc quic_per_connection_packet_writer.cc quic_time_wait_list_manager.cc quic_dispatcher.cc -L/usr/local/lib –lquic
- Main part of the server
  - net::tools::QuicSimpleServer server()
  - server.Listen()
  - base::RunLoop().Run();
- Main loop may be in QUIC library
  - Similar to Boost::asio and libevent
  - Need to integrate QUIC + DNS server
  - Need C++ DNS server implementations (Bundy or PowerDNS ?)

Port 53/UDP problem

- Port 53/UDP may be shared with both DNS over UDP and DNS over QUIC
- Assumption
  - Ignore address/port change in QUIC
- Client knows the response packet type
  - The response packet type should be known
- Servers need to accept both DNS/UDP and QUIC (and DNS/TCP)
  - Need to determine that received packet is DNS or QUIC at UDP query receiver

How to determine DNS or QUIC

- Basic idea
  - Parse it as DNS/UDP or QUIC. If errors occur, it's another protocol packet
  - If QUIC connection established, record the combination of (source/destination address/port)
- Proposed idea
  - If (source/destination address/port combination is established QUIC session) then it is QUIC
  - If the packet does not seem to be DNS/UDP, try QUIC (heuristic)
    - QR=0, QDCOUNT=1, higher byte of *COUNT=0
  - Parse received packet as DNS/UDP
  - If it parsed as a FORMERR, it is QUIC packet

Pseudo server code

```
Initialize
Create UDP/TCP socket;
Event loop (Boost::ASIO, libevent, select, etc.) {
  if (packet received) {
    recvfrom();
    if (established QUIC ?) { do QUIC; }
    else if (heuristic check !DNS ?) { do QUIC; }
    else {
      Parse received packet as DNS/UDP;
      If (FORMERR) {
        do QUIC;
        if (QUIC error) { response FORMERR; }
      }
    }
  }
  Do other works (timer, responder, …)
}
```

Current status

- The current QUIC specification is unsatisfactory and incomplete
- Implementation exists, but too complicated
  - Do you know simpler QUIC implementation written by C ?
- If possible, I would like to integrate DNS server (written by C++) and libquic
- Questions ?