NAT Detection in Secure Transport Protocols

Eric Kinnear (ekinnear@apple.com)
Tommy Pauly (tpauly@apple.com)
Christopher A. Wood (cawood@apple.com)

TLS
IETF 104, March 2019, Prague
Overview

Two drafts submitted in tandem:

**TLS**, draft-kinnear-tls-client-net-address

**QUIC**, draft-pauly-quic-address-extension

Cover use-case motivations

Examine and discuss tradeoffs

Defer wire-encoding discussions to the list
Use Cases
Middlebox keepalives

Long-lived connections need to send keepalives to avoid NAT or firewall mapping timeouts

Not detecting a NAT doesn't remove the need for handling keepalives, since firewalls may not translate addresses

Detecting a NAT can optimize a client's algorithm by giving a better heuristic for keepalive timeouts (more aggressive when NATs are certain)
Use Cases
Unique Identifiers

Is my client IP address a unique end-to-end identifier?

*For example*, if a client is behind a NAT, using separate connections for DoT queries can improve privacy; for a public IP address, this approach may actually harm privacy.
Use Cases
NAT rebinding detection

While NAT rebinding breaks TCP connections, MPTCP and QUIC can survive NAT rebindings

Getting a public IP address update notifies the client when a rebinding has occurred

Useful for QUIC migration

Identifies NAT timeout values
Use Cases
Detecting ASN for metrics

Clients behind NATs have difficulty detecting what ASN they are connected over without explicit probes

Detecting public IP addresses can help clients better identify their own network attachments using existing connections

Can be viewed as a privacy issue, but only giving information that everyone else upstream in the network already knows
Goals and Constraints

Goals
• Allow endpoints to detect the presence of address-transforming middleboxes
• Allow endpoints to discover their own "public" IP addresses

Constraints
• Address information must be encrypted in transit
• Only public addresses should be transmitted
• Cannot rely on validating address information
Protocol Proposals
TLS & QUIC

TLS Client Address extension

Asymmetric

Clients request their public address from TLS server

QUIC Address Request extension

Symmetric

Endpoints request their public address from peer
TLS Client Address
Example

Client 1.2.3.4 → 2.3.4.5 → Server 4.3.2.1

Client’s address is 2.3.4.5
TLS Client Address

Example

Client (1.2.3.4) ---------------> NAT (2.3.4.5) ---------------> Server (4.3.2.1)

ClientHello
+ ...
+ network_address

------------->

ServerHello
+ ...
+ {EncryptedExtensions}
+ network_address (2.3.4.5)
  + {CertificateRequest*}
  + {Certificate*}
  + {CertificateVerify*}
  + {Finished}

<-------------

{Certificate*}
{CertificateVerify*}
{Finished}
[Application Data]

---------

[Application Data]

Client's address is 2.3.4.5
TLS Client Address

Limitations

Cannot detect stateful firewalls that do not translate addresses

Only clients benefit (less server complexity?)

Occurs only during handshake, which may need to be changed for MPTCP
QUIC Address Request
Example

Client’s address is 2.3.4.5
QUIC Address Request Example

Client (1.2.3.4) NAT (2.3.4.5) Server (4.3.2.1)

<-------->
QUIC(TLS) Handshake
<-------->

[PUBLIC_ADDRESS_REQUEST(id=0)]

<-------->

[PUBLIC_ADDRESS_RESPONSE(id=0,
    type=0x00,
    value=2.3.4.5,
    port=4567)]
QUIC Address Request

Limitations

Like TLS, cannot detect stateful firewalls that do not translate addresses

Endpoints can request at any time, but NAT rebinding detection relies on a client asking for its address when it thinks a rebinding may have occurred (idle time, or due to PATH_CHALLENGE)
Questions?