Enhanced Performance and Privacy for Core Internet Protocols

Erik Sy
Transport protocols should not allow distinguishing Alice and Bob as the sender of a message.
Motivation

- **Increase the quality of experience for web users**
  - The delay of the connection establishments presents a significant overhead of an average web flow
Investigated Protocols

- **Application**:
  - DNS
  - HTTP/2

- **Security**:
  - TLS
  - QUIC

- **Transport**:
  - TCP
  - UDP

- **Network**:
  - IP
Introducing Resolver-Less DNS

Application
- DNS

Security
- TLS

Transport
- TCP

Network
- IP

Higher layers
- HTTP/2

Transport and Application
- QUIC

Network
- UDP
Introducing Resolver-Less DNS

- Web server provides relevant DNS records to it’s clients
  - Improves client’s privacy posture towards resolver & reduces delay

Validation Mechanisms of Resolver-Less DNS

- Client does not send application data to presented IP address before a successful validation of the used DNS record

- Preferred validation mechanism uses server authentication during connection establishment

- Fallback validation mechanism includes traditional DNS lookup to make a comparison between both DNS records
1% of clients saves at least 80ms per DNS query compared to status quo
Introducing TCP Fast Open (RFC 7413, Dec 2014)
Introducing TCP Fast Open (RFC 7413)

- Allows validating the client’s IP address without an additional round trip

1) Initial handshake

Client

SYN + Fast Open cookie request

Server

ACK + application data

regular TCP data flow can follow...

2) Abbreviated handshake

Client

SYN + Fast Open cookie_1

Server

SYN-ACK (SYN, application data)

ACK

application data

regular TCP data flow can follow...
User Tracking via TCP Fast Open

- **Main findings**
  - Fast Open cookies present a kernel-based tracking mechanism
  - Tracking feasible for network observer
  - Feasible tracking periods are unrestricted
  - Enables tracking across private browsing modes, browser restarts, and different applications

- **Reactions by browser vendors**
  - Mozilla stopped using TFO within Firefox
  - Microsoft stopped using TFO within the private browsing mode of Edge
Performance Limitation of TCP Fast Open

- Requirement of matching server IP address for abbreviated handshakes does not anticipate real-world load balancing.
Proposed TCP Fast Open Privacy

- Cross-layer approach to mitigate privacy and performance issues of TFO

1) Initial handshake

Client

SYN + TFO cookie request

 SYN-ACK

ACK

CHLO_{TLS}

SHLO_{TLS}

NewPSK_{TLS}

regular TLS data flow can follow...

Stores PSK in TLS cache

Server

Forwards cookie request to TLS

Uses cookie found in PSK

Appends TFO cookie to PSK

2) Abbreviated handshake

Client

SYN + TFO cookie

CHLO_{TLS}(PSK) + Request

SYN-ACK(SYN, TLS data)

SHLO_{TLS} + Response

ACK

regular TLS data flow can follow...

Server

Validates cookie + accepts TLS data
Introducing TLS Session Resumption

Application
- DNS
- HTTP/2

Security
- TLS
- QUIC

Transport
- TCP
- UDP

Network
- IP
Introduction to TLS Session Resumption

- Allows a client-server pair to establish a new TLS connection with a previously exchanged symmetric key
  - Reduces the delay and the computational overhead of TLS handshakes
  - Server can uniquely identify clients based on this secret key

- Deployment on the Internet
  - 96% of TLS-enabled Alexa Top Million Sites support TLS resumption
  - All popular web browsers support this feature, which is included in every TLS version
Main findings

- Safari and Firefox can be tracked for at least 24h using this mechanism
- Prolongation attack extends feasible tracking periods
- Only TLS v1.3 protects against tracking by network observer
- Most browsers do not protect against third-party tracking via TLS SR

Domain Trees of popular Websites

- Alexa Top 1K Site requires on average 20.24 connections to different hosts
- These hostnames form on average 9.49 TLS trust groups

Proposed TLS 1.3 Extension

- TLS 1.3 allows resumptions across hostnames, if the corresponding hostnames can be validated via the same server certificate.

- Server signals that a group of hostnames mutually support TLS resumptions
  - Presented server certificate needs to be valid for these hostnames.

- SAN-list of certificate can be used to define this group
  - Adds complexity to the generation of server certificates
  - Helps to avoid resumptions to hostnames for which the cert is not valid.

- Extension for the NewSessionTicket frame.
Performance of TLS 1.3 Connection Establishments

- **Elapsed time**

<table>
<thead>
<tr>
<th>Network latency</th>
<th>Initial</th>
<th>1-RTT resumed</th>
<th>0-RTT resumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 ms</td>
<td>29.2 ms</td>
<td>6.3 ms</td>
<td>6.6 ms</td>
</tr>
<tr>
<td>50 ms</td>
<td>190.1 ms</td>
<td>160.1 ms</td>
<td>109.6 ms</td>
</tr>
<tr>
<td>100 ms</td>
<td>340.8 ms</td>
<td>310.3 ms</td>
<td>209.7 ms</td>
</tr>
</tbody>
</table>

- **CPU time**

<table>
<thead>
<tr>
<th>Peer</th>
<th>Initial</th>
<th>1-RTT resumed</th>
<th>0-RTT resumed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>7.8 ms</td>
<td>2.3 ms</td>
<td>2.6 ms</td>
</tr>
<tr>
<td>Client</td>
<td>9.2 ms</td>
<td>2.4 ms</td>
<td>2.5 ms</td>
</tr>
</tbody>
</table>
Results for an average Website

- Converts about 58.7% of the required full TLS handshakes to resumed connection establishments

- Reduces the required CPU time for the TLS connection establishments by about 44%

- Reduces the elapsed time to establish all required TLS connections by up to 30.6%
Introducing QUIC

- Application: DNS, HTTP/2
- Security: TLS
- Transport: TCP, UDP
- Network: IP

QUIC encompasses DNS, TLS, TCP, and UDP at the application, security, transport, and network layers, respectively.
Introduction to the QUIC Transport Protocol

- QUIC is going to replace TLS over TCP in HTTP/3

- Improves problems of TLS over TCP
  - Protocol Entrenchment
  - Implementation Entrenchment
  - Handshake Delay
  - Head-of-line Blocking
  - Mobility

- Google’s QUIC protocol is already widely deployed on the Internet
  - Accounts for 7% of global Internet traffic
  - Supported by Google Chrome (approx. 60% browser market share)
Source-address token speed up the validation of the client’s IP address in subsequent connections between the same peers.
Tracking via QUIC’s Server Config

- QUIC’s server config contains a public key used to bootstrap the cryptographic connection establishment

- Client reuses server config across different connections

- Tracking feasible if server distributes unique server configs/ server config identifiers to its clients
Tracking via QUIC

- **Main findings**
  - Default configuration of Chrome enables unlimited tracking periods
  - Third-party tracking feasible via this mechanism for Chrome
  - Network observers may track user’s via QUIC’s server config

- **Reactions by browser vendors**
  - Google Chrome restricts feasible tracking periods to one week

---

Shared Client IP Address Validation

- QUIC server having a TLS trust-relation accept source-address tokens generated by each other
  - Each accepted source-address token allows client-server pair to save a round trip time during the connection establishment

- Novel QUIC transport parameter is used to inform the client about other hosts accepting a provided validation token
Proposal saves a round-trip time on 58.75% of the established connections
Performance Improvements for the average Website (2/2)

- Longest path of sequential connections with retry is reduced by 39.1%

![Graph showing CDF Alexa Top 1K Sites vs. Number of sequential initial handshakes with retry required to load the website. The graph compares using the proposal vs. without the proposal.]
Shared IP Address Validation using Out-Of-Band Token

Distribution of out-of-band validation token via DNS resolver or other QUIC server

Client queries domain name

DNS resolver

Server

queries domain name

address, token

ClientHello, token

peers proceed with connection establishment ...

7: Sy et al. “QUICker Connection Establishment with Out-Of-Band Validation Tokens” (2019)
Performance gains based on Out-Of-Band Validation Token

- Each initial QUIC connection establishment can save up to a RTT
Introducing the QuicSocks Design

- Assumes a QuicSocks Proxy colocated with the DNS resolver

8: Sy et al. “Accelerating QUIC's Connection Establishment on High-Latency Access Networks” (2019)
Proposal achieves better performance if $RTT_{Server} < RTT_{direct}$

<table>
<thead>
<tr>
<th>Stateless retry</th>
<th>Latency to establish connection (incl. DNS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>w/o</td>
<td>$RTT_{DNS} + RTT_{direct}$</td>
</tr>
<tr>
<td>with</td>
<td>$RTT_{DNS} + 2 \cdot RTT_{direct}$</td>
</tr>
</tbody>
</table>

$RTT_{DNS}$ + $RTT_{Server}$
Empirical Performance Evaluation

- 24.3% of nodes saves at least 15ms without and 30ms with stateless retry
Recommended Privacy Protections

- Deactivate TCP Fast Open
- Applications restricting tracking via HTTP cookies should apply the same limitations to tracking via the presented mechanisms in TLS and QUIC
- Deploying resolver-less DNS
Short lifetime for the investigated tracking mechanisms provides already significant performance gains while limiting feasible tracking periods.
Conclusion

- TCP Fast Open, TLS, and QUIC contain mechanisms that can severely harm the privacy of users
- Popular browsers do not sufficiently protect against these privacy risks
- Investigated mechanisms should be used with a short expiration time to balance the performance versus privacy trade-off
- Several performance optimizations are feasible for core Internet protocols
Thank you

Questions and Answers

E-mail: Brave@erik-sy.de
Slides: https://erik-sy.de/brave