TCPLS: Closely Integrating TCP and TLS

Florentin Rochet, Emery Assogba, Olivier Bonaventure
UCLouvain, Belgium
Motivations

• Towards cross-layer TLS/TCP?
Motivations

- Towards cross-layer TLS/TCP?
  - The world changes! TCP without TLS over untrusted networks becomes unrealistic.
Motivations

- Towards cross-layer TLS/TCP?
  - The world changes! TCP without TLS over untrusted networks becomes unrealistic

- Maybe we can benefit from merging the stack?
Motivations

- Towards cross-layer TLS/TCP?
  - The world changes! TCP without TLS over untrusted networks becomes unrealistic

1999  2006  2008  2016  2018

TLS 1.0  TLS 1.1  Let’s Encrypt  TLS 1.3
TLS 1.2  DV Cert

- Maybe we can benefit from merging the stack?
- TCP needs a boost to compete with QUIC in the future
  - Improving on Header space issue; middlebox interferences
Motivations

- Towards cross-layer TLS/TCP?
  - The world changes! TCP without TLS over untrusted networks becomes unrealistic

- Maybe we can benefit from merging the stack?
- TCP needs a boost to compete with QUIC in the future
  - Improving on Header space issue; middlebox interferences
- Towards more application tuning

1999 2006 2008 2016 2018

TLS 1.0 TLS 1.1 Let’s Encrypt TLS 1.3
TLS 1.2 DV Cert
Motivations

- Towards cross-layer TLS/TCP?
  - The world changes! TCP without TLS over untrusted networks becomes unrealistic.

- Maybe we can benefit from merging the stack?
- TCP needs a boost to compete with QUIC in the future.
  - Improving on Header space issue; middlebox interferences.
- Towards more application tuning.
  - Lack of complex transport features exposed to the application.
TCPLS's Secure Control Channel
- We aim at a synergy with recent efforts in the Linux kernel for more eBPF in TCP
- TCPLS messages are indistinguishable from TLS 1.3 APPDATA messages
How?

- Major improvement of TCP's extensibility and deployability
  - TCPLS's Secure Control Channel
    - We aim at a synergy with recent efforts in the Linux kernel for more eBPF in TCP
    - TCPLS messages are indistinguishable from TLS 1.3 APPDATA messages
How?

- API to export complex transport features: composable basic blocks
  - Multihoming, multipathing, QUIC-like streams, 0-RTT, Happy Eyeball, TCP options, eBPF injection, ...
  - E.g., notion of path, notion of streams: implication of composing streams with paths
How?

- API to export complex transport features: composable basic blocks
  - Multihoming, multipathing, QUIC-like streams, 0-RTT, Happy Eyeball, TCP options, eBPF injection, ...
  - E.g., notion of path, notion of streams: implication of composing streams with paths

- Showing the similarities and the nuanced differences between QUIC and TCPLS
TCPLS Secure Channel

- Goal: Provides an encrypted and authenticated channel to negotiate TCP/IP extensions

<table>
<thead>
<tr>
<th>Type</th>
<th>Version</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Option</td>
<td>ADD_ADDR</td>
<td>v6</td>
</tr>
<tr>
<td></td>
<td>a11:90d5:be:11c5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>META</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Client

Server

v4 v6

SYN+CH

SYN+ACK+SH+Opt

CF

CH+JOIN

Cookie, ConnID
TCPLS Secure Channel

- Goal: Provides an encrypted and authenticated channel to negotiate TCP/IP extensions

- Use TLS 1.3's protocol extensibility design
  - The "visible" type (Type) is APPDATA
  - The true type (TType) is located at the end of the payload
TCPLS API

- Goal: Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
TCPLS API

- Goal: Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
- Implementation advancement
  - Early results on: QUIC-like 0-RTT, multipath, QUIC-like streams, TCP options securely exchanged, eBPF injection (Congestion Control), Connection Migration.
  - Ongoing work on: Failover, better multipath control, API
TCPLS API

- Goal: Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
- Implementation advancement
  - Early results on: QUIC-like 0-RTT, multipath, QUIC-like streams, TCP options securely exchanged, eBPF injection (Congestion Control), Connection Migration.
  - Ongoing work on: Failover, better multipath control, API

Figure 3: API Workflow example. * means optional call, [ ] means optional call flow, and } means encrypted.
TCPLS API

- **Goal:** Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
- **Implementation advancement**
  - Early results on: QUIC-like 0-RTT, multipath, QUIC-like streams, TCP options securely exchanged, eBPF injection (Congestion Control), Connection Migration.
  - Ongoing work on: Failover, better multipath control, API

![API Workflow example](image)

*Figure 3: API Workflow example.* `*` means optional call, `[ ]` means optional call flow, and `{ }` means encrypted.
TCPLS API

- Goal: Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
- Implementation advancement
  - Early results on: QUIC-like 0-RTT, multipath, QUIC-like streams, TCP options securely exchanged, eBPF injection (Congestion Control), Connection Migration.
  - Ongoing work on: Failover, better multipath control, API

![Diagram of API Workflow example](image-url)

Figure 3: API Workflow example. * means optional call, [ ] means optional call flow, and { } means encrypted.
TCPLS API

- **Goal:** Export complex transport level features to the applications
  - "Export" means that applications make decisions about the transport features
  - What features should be exported, what should not?
- **Implementation advancement**
  - Early results on: QUIC-like 0-RTT, multipath, QUIC-like streams, TCP options securely exchanged, eBPF injection (Congestion Control), Connection Migration.
  - Ongoing work on: Failover, better multipath control, API
Example: App-level Con Migration

tcpls_handshake_properties_t prop = {NULL};
prop.client.mpjoin = 1;
prop.client.zero_rtt = 1;
prop.client.dest = dest_addr;
/** Make a tcpls mpjoin handshake */
ret = tcpls_handshake(tcpls, &prop);
if (!ret) {
  /** Create a stream on the new connection and attach it now */
tcpls_stream_new(tcpls, NULL, dest_addr);
tcpls_streams_attach(tcpls, 0, 1);
  /** Close the stream on the initial connection */
tcpls_stream_close(tcpls, streamid_initial, 1);
}
Example: App-level Con Migration

```c
1 tcpls_handshake_properties_t prop = {NULL};
2 prop.client.mpjoin = 1;
3 prop.client.zero_rtt = 1;
4 prop.client.dest = dest_addr;
5 /** Make a tcpls mpjoin handshake */
6 ret = tcpls_handshake(tcpls, &prop);
7 if (!ret) {
8     /** Create a stream on the new connection and attach it now */
9     tcpls_stream_new(tcpls, NULL, dest_addr);
10    tcpls_streams_attach(tcpls, 0, 1);
11 /** Close the stream on the initial connection */
12    tcpls_stream_close(tcpls, streamid_initial, 1);
13 }
```
- Download of a 60MB file over a virtual network with two 30mbps links
- Multipath mode activated during the migration
Research Agenda

- Applicability of TCPLS's ideas
  - A more secure MPTCP?
    - MPTCP ADD_ADDR and RM_ADDR inside the TCPLS secure channel + new setsockopt
    - We can drop the in clear symmetric key exchange and the truncated HMAC
    - Significant but highly beneficial redesign of MPTCP
Research Agenda

- Applicability of TCPLS's ideas
  - Helpful for detecting Middleboxes messing with TCP?
    - Send options in TCP, send them also in TCPLS's control channel, and compare
Research Agenda

- Applicability of TCPLS's ideas
  - Pluginizing TCPLS?
    - Similarly to PQUIC and xBGP; advancing towards pluginized protocols -- e.g.:
      - Deploying cutting-edge research in AEAD ciphers through plugins!
      - Letting the sender send and set the multipath scheduler to the receiver
      - Configuring the peer's TCP stack -- in line with current efforts in the kernel
Research Agenda

- Applicability of TCPLS's ideas
  - Thinking about the efficiency of the cross-layer approach?
    - Performance gain at the cost of design complexity. e.g.:
      - TCPLS can have a zero-copy code path on the receiver if the size of the TLS records matches the sender window (i.e., no record fragmentation)
Research Agenda

- Applicability of TCPLS's ideas
  - Thinking about the efficiency of the cross-layer approach?
    - Performance gain at the cost of design complexity. e.g.:
      - TCPLS can have a zero-copy code path on the receiver if the size of the TLS records matches the sender window (i.e., no record fragmentation)
  - Much more :-)
Research Agenda

- Applicability of TCPLS's ideas
  - Thinking about the efficiency of the cross-layer approach?
    - Performance gain at the cost of design complexity. e.g.:
      - TCPLS can have a zero-copy code path on the receiver if the size of the TLS records matches the sender window (i.e., no record fragmentation)
  - Much more :-)