TCPLS : Closely Integrating TCP and TLS

Florentin Rochet, Emery Assogba, Olivier Bonaventure

UCLouvain

This work was partially supported by the Walloon government within the MQUIC project
Our current stack

- **TLS 1.3**
  - provides security
  - More and more used on WANs and by a variety of applications

- **TCP**
  - provides connection abstraction, reliability, congestion control
  - Most popular transport protocol

- In the future, TCP could **always** be used with TLS
**Control and data separation in TCP**

- Very simple

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>THL</th>
<th>Reserved</th>
<th>Flags</th>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Control**
- **Data**

restricts the length of TCP Options
# TLS 1.3 in one slide

## Secure Handshake

- **ClientHello**
  - ... *Extension*
- **ServerHello**
  - ... *EncryptedExt*
- **Finished**
- **Application Data**
- **Finished**

0-rtt mode in parallel with the TCP handshake is possible.

## The encrypted TLS records

<table>
<thead>
<tr>
<th>Type</th>
<th>Version</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TLS 1.3</td>
<td></td>
</tr>
</tbody>
</table>

- *Encrypted and authenticated*
- *TrueType*

Always 23 App Data

Encrypted and thus invisible to middleboxes.
An integrated stack

- Key idea
- Use new TLS record types to carry TCP control plane information
  - TLS record to carry TCP option
  - TCP option inside ClientHello Extension
  - TCP option inside ServerHello EncryptedExt
- TCPLS has 2 different channels for TCP control
  - regular TCP options
  - Encrypted TLS records
The TCPLS control channels

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequence number</td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
</tr>
</tbody>
</table>

TCP Options

Enc. TLS rec #1 (control)
Enc. TLS rec #2 (data)

Application

TCPLS
TLS
Socket
TCP

Application

TCPLS
TLS
Socket
TCP

TLS records

TCP header
Use case: Securing Multipath TCP

• Security concerns
  • token is exchanged inside SYN/SYN+ACK
  • ADD_ADDR authentication
  • ADD_ADDR not reliable

• With TCPLS
  • Derive token from TLS secrets
  • TCPLS record for ADD_ADDR
    • reliable and authenticated
  • REMOVE_ADDR could still be sent as TCP option
Use case: Stronger TFO

• Concern
  • The security of TFO is limited by the length of the TCP options in the SYN to encode the cookie

• TCPLS approach
  • Use TLS’s 0-RTT and
    • send ClientHello inside SYN payload with TCPLS cookie
    • send ServerHello inside SYN+ACK payload with TCPLS cookie

  • Cookies can be longer and more secure by leveraging the existing TLS mechanisms
  • Middlebox interference
    • Apple’s measurements do not seem to indicate that the length of the payload in the SYN is a strong factor in middlebox interference
Use case : More space for TCP options

• TCPLS approach
  • More options during the handshake
    • Leverage the 0-RTT handshake
      • ServerHello inside SYN+ACK and TCP Options as ServerHello EncryptedExt
    • Define TLS record type to carry TCP options

• Late negotiation of TCP extensions
  • Since TLS records are reliably exchanged, we could also negotiate a TCP extension after
    the establishment of a connection
Use case : True keepalives

• Concern
  • Keepalives really part of TCP

TCPLS approach
• New ping/pong TCPLS record type
  • Hosts can send ping/pong records including data without interfering with payload
  • TCPLS can negotiate keepalive intervals and other informations

RFC1122
4.2.3.6 TCP Keep-Alives
Implementors MAY include "keep-alives" in their TCP implementations, although this practice is not universally accepted. If keep-alives are included, the application MUST be able to turn them on or off for each TCP connection, and they MUST default to off.

Keep-alive packets MUST only be sent when no data or acknowledgement packets have been received for the connection within an interval. This interval MUST be configurable and MUST default to no less than two hours.
Use case: Secure session release

• Concern
  • Middleboxes or attackers can force the termination of TCP connections using RST or FIN

• TCPLS approach
  • New authenticated record type indicating end of TCPLS connection
  • If RST or FIN are received before the exchange of this record, then the underlying TCP connection can be automatically reestablished
Use case: Happy eyeballs

- Server supports IPv4 and IPv6
  - Client learns addresses from DNS and initiates IPv6 and later IPv4 connection

- With TCPLS
  - Server uses EncryptedExt in ServerHello to advertise its alternate address
    - Similar to what QUIC connection migration or MPTCP’s ADD_ADDR
  - Client learns alternate server address during handshake
  - Client can create connection to alternate address, test it and migrate the connection
Use case: Connection migration

• Concern
  • Smartphone wants to move to cellular while preserving established TCPLS session

• Implemented TCPLS approach
  • Server provides connection identifier and cookie in ServerHello
  • Client creates second TCPLS subflow to server using this information
  • Server and client move data transfer to new TCPLS subflow
Conclusion

• Don’t consider TCP and TLS as separate and independent protocols
• TLS 1.3 can be efficiently combined with TCP to improve it
• More details are available in our Hotnets’20 paper

• There is running code based on picotls at https://pluginized-protocols.org
Conclusion

• Don’t consider TCP and TLS as separate and independent protocols
• TLS 1.3 can be efficiently combined with TCP to improve it
• More details are available in our Hotnets’20 paper
• There is running code based on picotls at https://pluginized-protocols.org