Interface Extensions for TCP INC

draft-bittau-tcpinc-api-00

Andrea Bittau, Dan Boneh, Daniel Giffin, Mark Handley, David Mazières, and Eric Smith

November 4, 2015
Motivation

TCPINC most likely to gain deployment through phases

1. Ship with OS distributions, but disabled by default
2. Some applications and hosts enable it
3. OS distributions enable system-wide by default
4. Applications take advantage of Session ID for stronger security

Steps 2–4 require API and configuration extensions

If extensions are similar across OSes, will facilitate adoption
Leveraging existing mechanisms

- Use Linux/BSD as a concrete model
- Per connection configuration uses `setsockopt/getsockopt`
  - Precedent: `TCP_NODELAY` (enables Nagle), `TCP_FASTOPEN` (enables TFO on passive opener), ...
  - Linux currently has 24 different per-socket TCP options
- System-wide configuration set with `sysctl`
  - Precedent: `net.ipv4.tcp_sack` (enable SACK), `net.ipv4.ip_local_reserved_ports` (ports not to assign when `sin_port == 0`)
  - Linux has over 50 IP and TCP `sysctl` configuration options
### Proposed socket options

<table>
<thead>
<tr>
<th>Option</th>
<th>RW</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENABLED</td>
<td>RW</td>
<td>1 = enable, 0 = disable, -1 = system default</td>
</tr>
<tr>
<td>SESSID</td>
<td>R</td>
<td>Return session ID</td>
</tr>
<tr>
<td>NEGSPEC</td>
<td>R</td>
<td>Return negotiated spec</td>
</tr>
<tr>
<td>SPECS</td>
<td>RW</td>
<td>Get/set specs allowed in negotiation</td>
</tr>
<tr>
<td>SELF_AWARE</td>
<td>RW</td>
<td>Get/set local application-aware level</td>
</tr>
<tr>
<td>PEER_AWARE</td>
<td>R</td>
<td>Get peer application-aware level</td>
</tr>
<tr>
<td>TIEBREAKER</td>
<td>RW</td>
<td>Set ENO’s 1-bit TCP-SO tiebreaker bit</td>
</tr>
<tr>
<td>ROLE</td>
<td>R</td>
<td>0 = “A” role, 1 = “B” role</td>
</tr>
</tbody>
</table>

- Option constants prefixed with `TCP_ENO_*` (correct next draft)
**Proposed new sysctls**

**eno_enabled** Determines system-wide default for TCP_ENO_ENABLED socket option.

**eno_specs** Determines system-wide default for TCP_ENO_SPECS.

**eno_bad_localport** Sets default value of ENABLED to 0, regardless of eno_enabled, when the local port number is in one of the ranges specified.

**eno_bad_remoteport** Similar to the previous option, but disabled ENO based on remote TCP port number.

- Should be placed alongside other TCP sysctls
  - Linux: net.ipv4.tcp_*
  - BSD: net.inet.tcp_*
Automatic configuration

- Also propose STUN-like service to detect ENO failure
  - Simple protocol over HTTP to get Session ID
- DHCP hooks should disable ENO if it makes connections hang
  - But test port 80 and all other ports separately, given prevalence of interception proxies
Raw mode

- Two more socket options support "raw mode"
  - **TCPENO_TRANSCRIPT** – return ENO negotiation transcript
  - **TCPENO_RAW** – specify raw ENO option contents
    - TCP stack still sends first non-ACK ENO option
    - Disables any TCP-level encryption
- Idea: facilitate development/testing/debugging of new specs
  - E.g., could shoehorn TLS into legacy protocols this way
  - Not for TCPINC, but could be ancillary benefit of ENO