Interface Extensions for TCP-ENO

draft-bittau-tcpinc-api

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

IETF95

Thursday, April 7, 2016

Review: 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

Overview

Define two sets of configuration variables

- Per-connection (e.g., `setsockopt`/`getsockopt` on BSD/Linux)
- System-wide (e.g., sysctl on BSD/Linux)

Ample precedent for TCP behavior tweak APIs

- `TCP_NODELAY` (enables Nagle),
  `TCP_FASTOPEN` (enables TFO on passive opener), …
- `net.ipv4.tcp_sack` (enable SACK),
  `net.ipv4.ip_local_reserved_ports` (ports not to assign when `sin_port == 0`)
- Linux currently has 24 different per-socket TCP options and over 50 IP and TCP sysctl configuration options

What's new?

- Separate system-wide configuration variables to enable by default on active vs. passive connections
- New socket options `TCP_ENO_LOCAL_NAME` and `TCP_ENO_PEER_NAME`
- Table presents system-wide configuration more systematically
- Provide guidance on error numbers
- Configuration suggestions broken off into new document `draft-bittau-tcpinc-bcp`

Per-socket options

Option	RW	Meaning
`ENABLED`	RW	1 = enable, 0 = disable, -1 = system default
`SESSID`	R	Return session ID
`NEGSPEC`	R	Return negotiated spec
`SPECS`	RW	Get/set specs allowed in negotiation
`SELF_AWARE`	RW	Get/set local application-aware level
`PEER_AWARE`	R	Get peer application-aware level
`ROLEOVERRIDE`	RW	Set “b” bit in general suboption
`ROLE`	R	0 = “*A*” role, 1 = “*B*” role
`LOCAL_NAME`	R	role byte and session ID, concatenated
`PEER_NAME`	R	!(role byte) and session ID, concatenated

Option constants prefixed with `TCP_ENO_*`

Errors

Option	Existing use
`EINVAL`	General error
`EISCONN`	Calling `connect` twice
`ENOTCONN`	Calling `getpeername` when not connected

Map most failure conditions to one of three error codes

- `EINVAL`: can never work (e.g., request session ID when ENO disabled)
- `EISCONN`: too late to set parameter
- `ENOTCONN`: too early to read value

System-wide options

eno_enable_connect Default to use when TCP_ENO_ENABLED is −1 on connect

eno_enable_listen Default to use when TCP_ENO_ENABLED is −1 on accept

eno_bad_connect_ports Disables ENO when TCP_ENO_ENABLED is −1 and destination port is in one of the ranges specified, regardless of eno_enable_connect

eno_bad_listen_ports Similar to previous option, but based on local port number during accept

eno_specs Determines system-wide default for TCP_ENO_SPECS

Raw mode

Two more socket options support "raw mode"

`TCPENO_TRANSCRIPT` **– return ENO negotiation transcript**

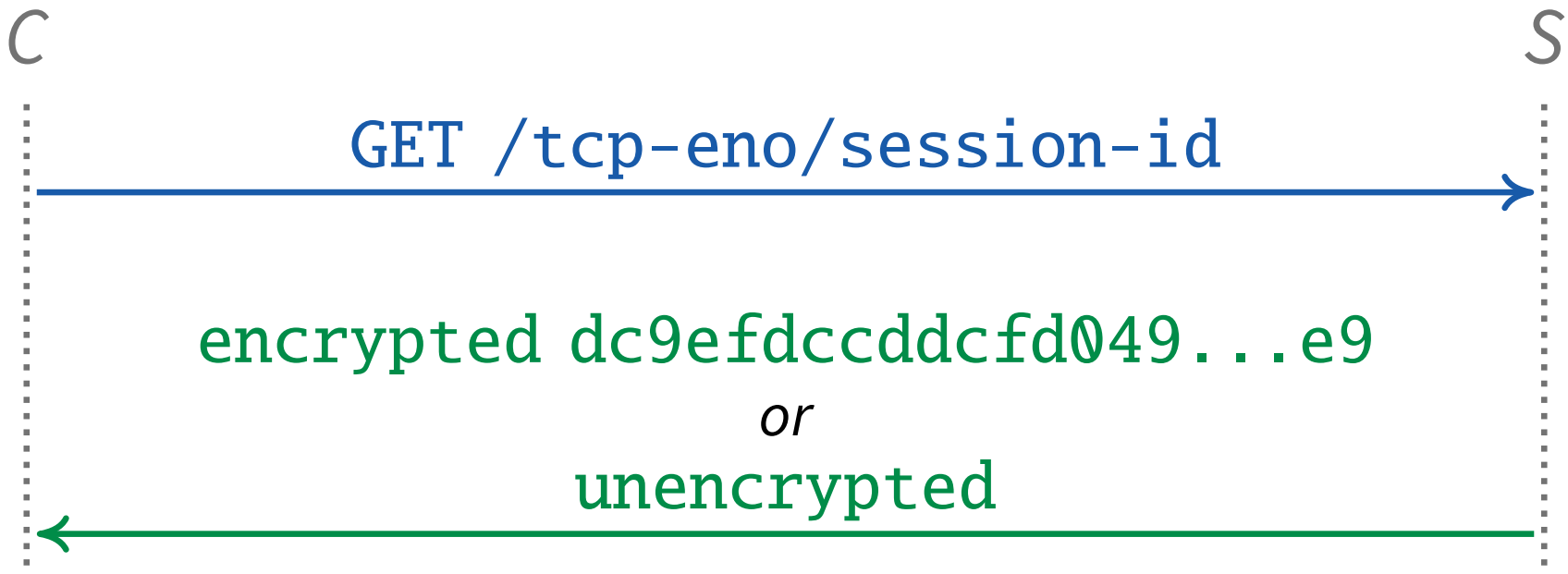
`TCPENO_RAW` **– specify raw ENO option contents**

- TCP stack still sends first non-ACK ENO option
- Disables any TCP-level encryption

Idea: facilitate development/testing/debugging of new specs

- Not for TCPINC, but could be ancillary benefit of ENO

Automatic configuration



Previously proposed STUN-like service to detect ENO failure

- Simple protocol over HTTP can be used by DHCP hooks
- Disable ENO if TCP connection (not just encryption) fails

Now in separate BCP document `draft-bittau-tcpinc-bcp`

- Need volunteers to coauthor or take over