Transparent TCP Timestamps
draft-scheffenegger-tcpm-timestamp-negotiation-03

Richard Scheffenegger
[rs@netapp.com]

Mirja Kühlewind
[mirja.kuehlewind@ikr.uni-stuttgart.de]

16. November 2011
82nd IETF, Taipei, Taiwan
Agenda

Timestamp Echo on SYN
TCP RTO calculation
Changes since -02 draft
TCP Timestamp Echo on SYN

+-----------------------------------------------+-----------------------------------------------+
| Kind=8 | 10 | TS Value (TSval) | TS Echo Reply (TSecr) |
+-----------------------------------------------+-----------------------------------------------+

1 1 4 4

- Some senders set TSecr != 0 on [SYN]

- RFC1323 states, a sender SHOULD set TSecr to zero, if no valid timestamp is known

TCP Timestamp Echo on SYN

1 016 245 629 [SYN] packets in trace
   – 878 264 323 (86.4%) unique [SYN]

3 634 [SYN] && TSecr != 0 (3.6 ppm)
   – 3 613 (99.4%) unique
   – 586 unique host pairs
   – 457 unique senders
   – 1 953 unique TSecr values
TCP Timestamp Echo on SYN

- Majority of TSecr values sent on [SYN] is seen in at least one preceding packet – always from a different TCP session between the two hosts

- 1 783 (of 1 953; 91.3%) [SYN] & TSecr != 0 preceded by
  - 1 191 [FIN, ACK]
  - 5 [FIN, PSH, ACK]
  - 573 [ACK]
  - 13 [PSH, ACK]

- Remaining 171 [SYN] have no packet from opposite side (unidirectional flow captured)
TCP Timestamp Echo on SYN

- passive OS signature on [SYN] && TSecr != 0

- MSS SACK TS NOP WS 3520 (96.86%)
- NOP NOP TS 56 (1.54%)
- MSS NOP WS NOP NOP TS 22 (0.61%)
- MSS NOP NOP TS 16 (0.44%)
- MSS NOP NOP TS WS 9 (0.25%)
- MSS SACK TS 8 (0.22%)
- MSS NOP WS NOP NOP TS NOP 1 (0.03%)
- MSS NOP WS NOP NOP TS NOP NOP 1 (0.03%)
- MSS NOP WS NOP NOP TS NOP WS 1 (0.03%)
TCP Timestamp Echo on SYN

- passive OS signature on [SYN] && TSecr != 0
  - p0f fails to identify senders (stringent heuristics)
  - SinFP uses “relaxed” heuristics

  1 IPv4: HEURISTIC0/P2: BSD: Darwin: 8.6.0
  2 IPv4: BH0FH0WH1OH0MH1/P2: BSD: FreeBSD: 4.4
  4 IPv4: HEURISTIC0/P2: BSD: FreeBSD: 4.10
  12 IPv4: BH0FH0WH0OH0MH1/P2: BSD: Darwin: 8.6.0

106 IPv4: unknown

  4 IPv4: BH0FH0WH1OH0MH1/P2: GNU/Linux: Linux: 2.2.x
  5 IPv4: BH0FH0WH0OH1MH0/P2: GNU/Linux: Linux: 2.4.x
  5 IPv4: BH0FH0WH1OH0MH1/P2: GNU/Linux: Linux: 2.4.x
  6 IPv4: BH0FH0WH1OH0MH0/P2: GNU/Linux: Linux: 2.4.x
  49 IPv4: BH0FH0WH0OH0MH1/P2: GNU/Linux: Linux: 2.6.x
  217 IPv4: HEURISTIC0/P2: GNU/Linux: Linux: 2.6.x
  409 IPv4: BH0FH0WH0OH0MH1/P2: GNU/Linux: Linux: 2.4.x
  2814 IPv4: HEURISTIC0/P2: GNU/Linux: Linux: 2.4.x
TCP Timestamp Echo on SYN

- 3509 sessions initiated from Linux (96.5%)
- 106 sessions initiated from unknown Host OS
- 19 sessions identified as BSD derived

- Known feature of Linux, when either `sysctl_tcp_tw_reuse` or `sysctl_tcp_tw_recycle` are enabled. This is used to shorten the TimeWait interval before sockets may be reused between two hosts.

- Observed TSecr values are not random, but preceeded in all observable instances by identical TSval (in a different TCP session), clearly indicating some kind of per-host timestamp cache.
TCP Timestamp Echo on SYN

- As timestamp values represented uptime in some way, values are not random distributed. Would lead to **reduced** false positive negotiation with current proposal.

- Most TSecr values are seen in [FIN] segments (61.3%). As [FIN] packets are closing a TCP session, small semantic change to send TSval = 0 on [FIN] would further alleviate this issue. As with timestamp offset randomization (draft-gont-tcpm-tcp-timestamps), may break linux feature - revert to regular timewait session closure.
TCP RTO calculation

Defined in RFC6298:

\[
\text{RTO} = \text{sRTT} + \max(G, k \times \text{RTTvar}) \quad ; \quad k = 4
\]

\[
\text{RTTvar} = (1 - \beta) \times \text{RTTvar} + \beta \times | \text{sRTT} - R' | \quad ; \quad \beta = 1/4
\]

\[
\text{sRTT} = (1 - \alpha) \times \text{sRTT} + \alpha \times R' \quad ; \quad \alpha = 1/8
\]

- The sampled RTT (R´) no longer includes delACK processing variability
- RTTvar will become smaller
- RTT represents primarily represents network delay
- One RTT extra delay before RTO expires added by restarting the RTO timer for each seen ACK
TCP RTO calculation – RFC1323 / 6298

- RTT includes delayed ACK delay
- Retransmission permitted after \( t_2 + \text{RTO} \)
- Typically at \( t_3 + \text{RTO} \)
TCP RTO calculation – new semantic

- RTT excludes delayed ACK delay
- Retransmission permitted after $t_2 + RTO$ – too fast?
- Explicitly stipulate $\max(t_2,t_3) + RTO$ (last received ACK + RTO)?
### Major Change since -01

- Redefined lower 16 bits
- No longer “alike” IEEE 754 floating point representation
  - No “special case” handling when evaluating Interval
  - Interval now signaled as scale/value pair like TCP window scaling
  - Conceptually, the Interval is a large integer, right-shifted “Adj” times to fit into “Int” with the most significant bit
  - Derived calculations stay identical (i.e. OWD) as with previous draft
  - Allows for implicit “clock source” quality signaling by leading zero bits in “Int”

<table>
<thead>
<tr>
<th>kind</th>
<th>len</th>
<th>TSval</th>
<th>TSecr</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
+-----+------+-------+-------+
| 1   | 2    | 5     | 8     |
| Ver | Mask | Reserved | Adj |
|     |      |         | 5    |
|     |      |         | 11   |
| Int |
```
Definition: 1 sec = 0x40 0000 0000 (39 bit length)
  – Right shifted 28 times, to fit into 11 bit
  – Int = 0x400, Adj=28

Alternative representations:
  – Int=0x200, Adj=29;
  – Int=0x100, Adj=30;
  – Int=0x080, Adj=31

Allows for clock tick intervals between ~16 sec and a few nanoseconds, each with up to 10 alternative representations (clock source quality levels)
**Major Change since -01**

- Expected to be (compile time) static value for current slow running TCP clock sources (100 µs or longer intervals)
Transparent TCP Timestamps

Questions:

- semantic change to timestamp to have TSval set to zero on [FIN]?

- Changed signaling of Interval (simple shift vs. IEEE-754 like float) less problematic?

- Ready for adoption as a WG Item?
Thank you!