TLS at a Root Experiment

Wes Hardaker (team USC/ISI) with help from Puneet Sood (team Google)

2022-07-23

A USC/ISI and Google TLS Experiment

Background

- Google has been experimenting with deploying authoritative DNS over TLS (DoT)
- Questions USC/ISI wanted to answer about deploying TLS at B:
  - How would enabling DoT affect our operational infrastructure?
  - What would the operational cost be?
  - Could we separate TLS from non-TLS during evaluation?
  - I.E. is there a viable path to safely deploying TLS?
- USC/ISI and Google jointly started a small TLS experiment

Experiment overview

- Google's side:
  - syn-probe b.root-servers.net for TLS/853
  - When available, limit TLS traffic to a total of 40-50%
  - Important: our results are not 100% TLS
- USC/ISI's side:
  - Isolated one backend at SIN
  - Installed bind 9.18.2
  - Configured to matching our existing deployment but with TLS
    - Note: No additional TLS tuning performed
  - Routed all google IPs to that backend
- Experiment:
  - Week 1: measured traffic/cpu-load without TLS
  - Week 2: enable TLS and measure again

Isolation Architecture

- Firewall's role:
  - Isolate normal production from experiment traffic
  - Filter by port
    - (eg, 853)
  - Filter by address
    - (eg, google)
    - *This report*
- SIN traffic flow:
  - TLS backend: Address (e.g. google) or TLS traffic
  - Other backends: Normal UDP/TCP production traffic

Results

Measurement Results Overview

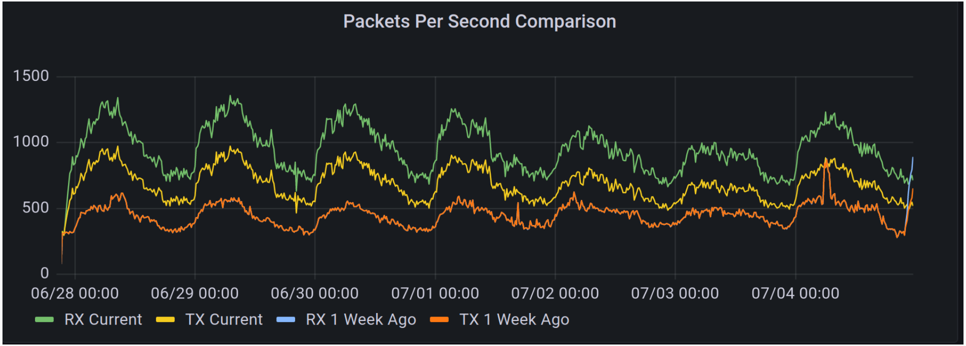
Measurements taken:

- Packets per second
- Bandwidth
- CPU load

In the following graphs we will see:

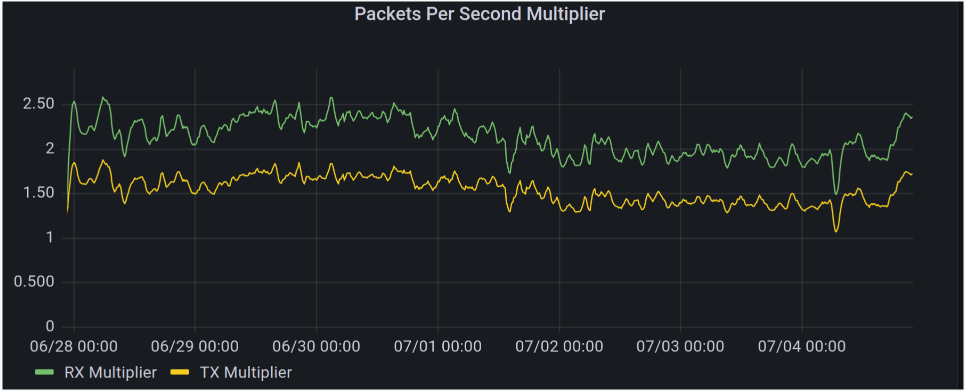
- A week long graph of each measurement
  - Measurement with TLS disabled
  - Measurement with TLS enabled
- A week long graph showing the multiplication factor
  - basically: $smooth_{1h}(\frac{NEW}{OLD-7d})$
  - (i.e., using a 1-hour smoothing window)

Packets Per Second Comparison



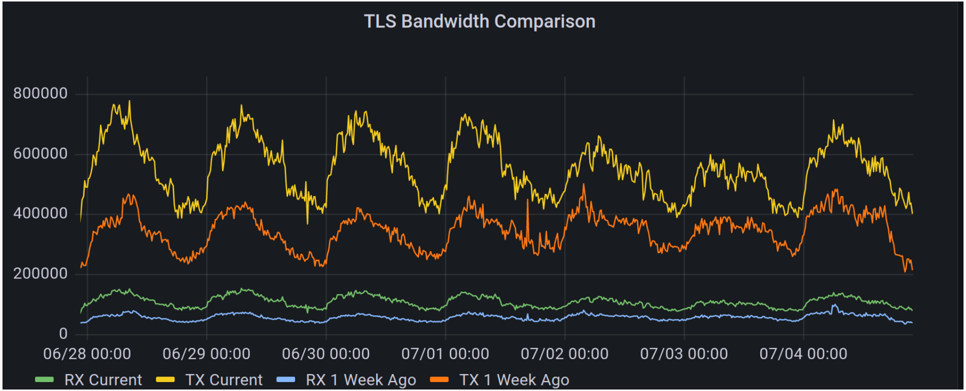
- Bottom line: a week of normal UDP/TCP RX/TX traffic (they overlap)
- Top 2 lines: a week of UDP/TLS experiment's RX/TX PPS

Packets Per Second Multiplier

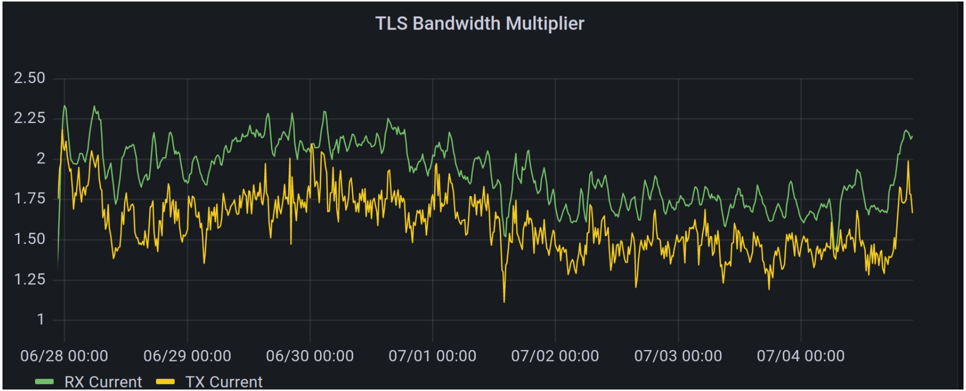


Dividing TLS PPS by normal traffic loads: RX = 2.12x, TX = 1.54x

Bandwidth Comparison

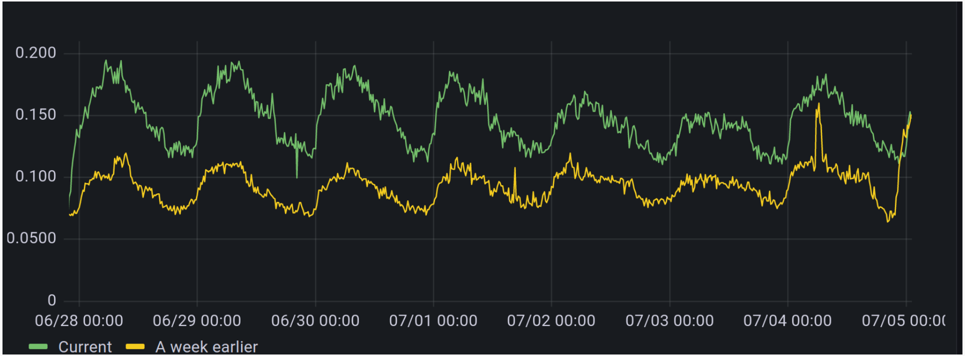


Bandwidth Multiplier

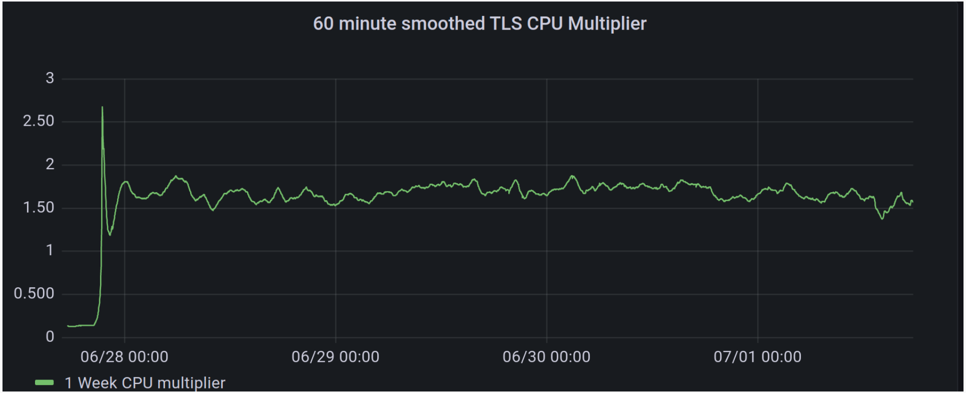


Multipliers: RX = 1.90x, TX = 1.60x

CPU Usage Comparison



CPU Usage Multiplier



Multipliers: CPU = 1.6x

Resource Multiplier Summary

Summarizing the multiplication graphs:

Measurement	Multiplier
PPS RX	2.12
PPS TX	1.54
Bandwidth RX	1.90
Bandwidth TX	1.60
CPU Load	1.60

Reminder: Reminder: traffic simulates a 40-50% TLS

Resource Multiplier Summary

Summarizing the multiplication graphs:

Measurement	Multiplier
PPS RX	2.12
PPS TX	1.54
Bandwidth RX	1.90
Bandwidth TX	1.60
CPU Load	1.60

Reminder: Reminder: traffic simulates a 40-50% TLS

Take-away: operationally feasible, but with a ~1.5 - 2x cost for 50% load

Future considerations

Future deployment considerations TBD

1. Optimize performance
   - TCP tuning
   - TLS tuning (e.g. tunnel reuse parameters)
   - Larger load testing
2. Measure other parameters
   - e.g. open files, memory, etc
3. Compare results with other studies
4. Deploy safely to more sites
5. . . .
6. Profit

What would TLS at the roots mean for RSSAC-002?

Will not be affected by TLS:

- load-time
- zone-size

Affected by TLS but easily measurable:

- traffic-volume *(requires spec change for "tls-" prefix?)*
- unique-sources

Requires internal name-server logging:

- traffic-sizes
- rcode-volume

What would TLS mean for DITL?

- Currently all DITL collections record IP/QName/DNS-details in PCAP
- With TLS:
  - PCAPs alone would hide DNS query details
  - In-server capture/logging needed to retain full-DNS details
    - Both bind and knot (at least) support dnstap today
    - But not PCAP based output
- What would the OARC community expect/want?
  - We would need to ask them