Incremental Assurance for a Rust Network Stack

Galois Inc.

Michal Podhradsky, Tiago Ferreira, Ben Hamlin, Mike Dodds, Mike Beynon

* This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Tiago Ferreira

tiferrei · he/him

He/Him • Computer Scientist

@UCL-PPLV

Motivation

- Building a high assurance network stack *from scratch* is hard
    - complex protocols
    - needs to be fast *and* feature complete
    - hard to verify (timeouts, edge cases, …)
- Can we instead *make* an existing code high assurance?
    - large codebases (Linux network stack)
    - difficult to reason about (lwip, picotcp, …)

smoltcp

TCP/IP Stack for Embedded Rust

- designed for embedded systems
- written in Rust
- well documented
- unit tests
- fuzz testing
- popular
- ran on seL4 before (Camkes)

All-Time: 492,849

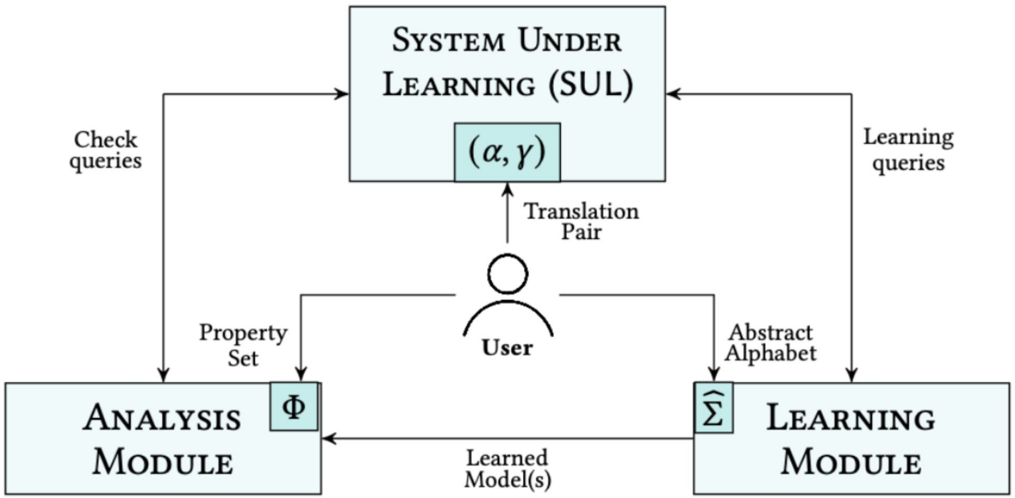
Recent: 92,090

Incremental assurance

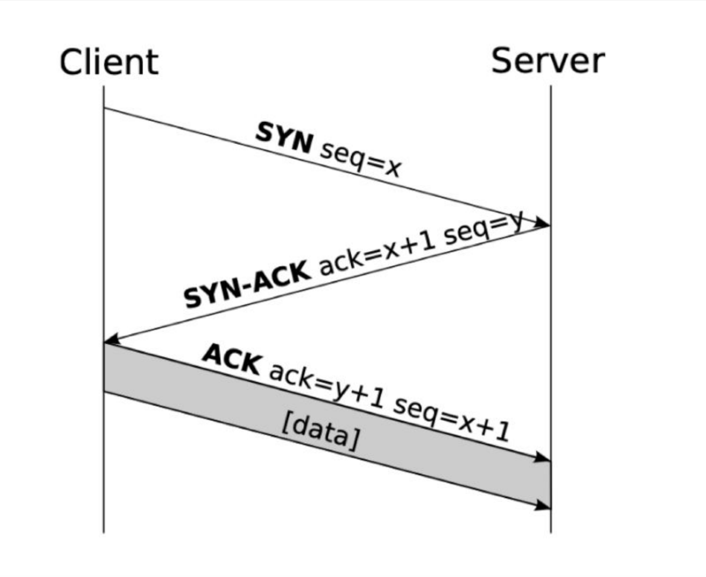
- **Prognosis**
  - https://dl.acm.org/doi/abs/10.1145/3452296.3472938
  - automated closed-box learning and analysis of models of network protocol implementations
  - model based verification of TCP protocol
- **Kani Rust verifier**
  - https://model-checking.github.io/kani/
  - symbolic execution
  - TCP protocol logic and packet format correctness

Prognosis

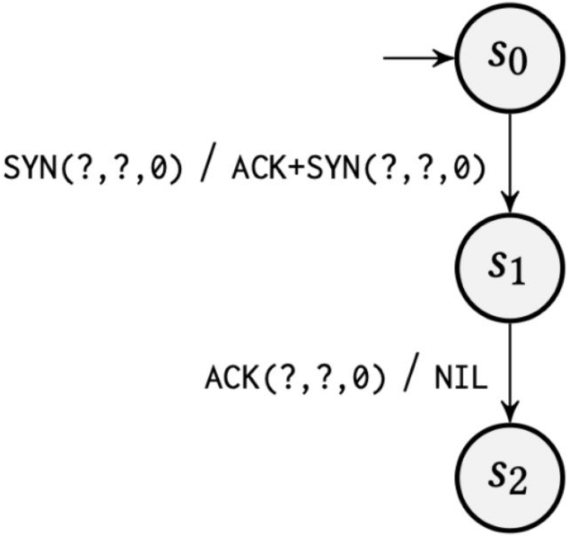
- An automated, closed-box tool for protocol inference.
- Based on Automata Learning, adapted for industry use.



Protocols as State Machines

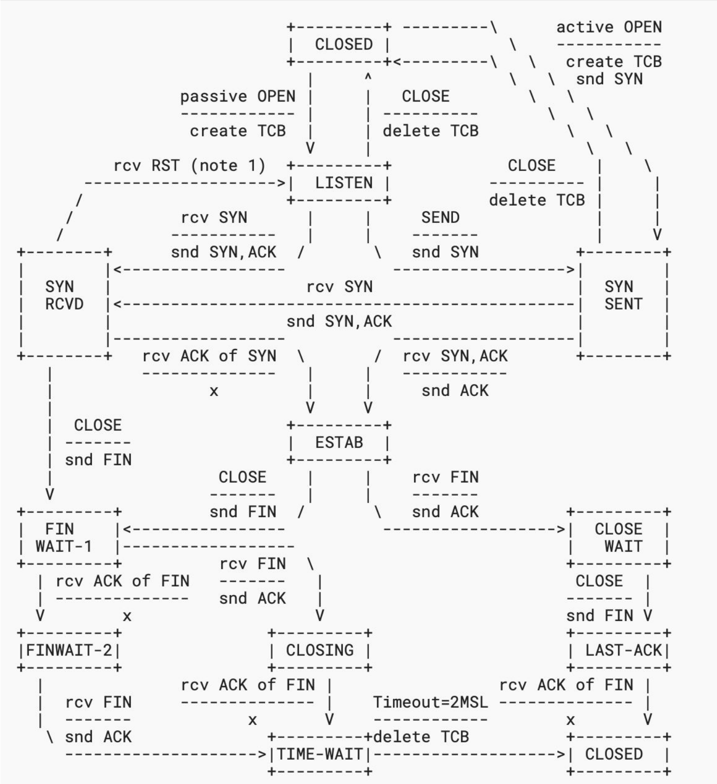


≊



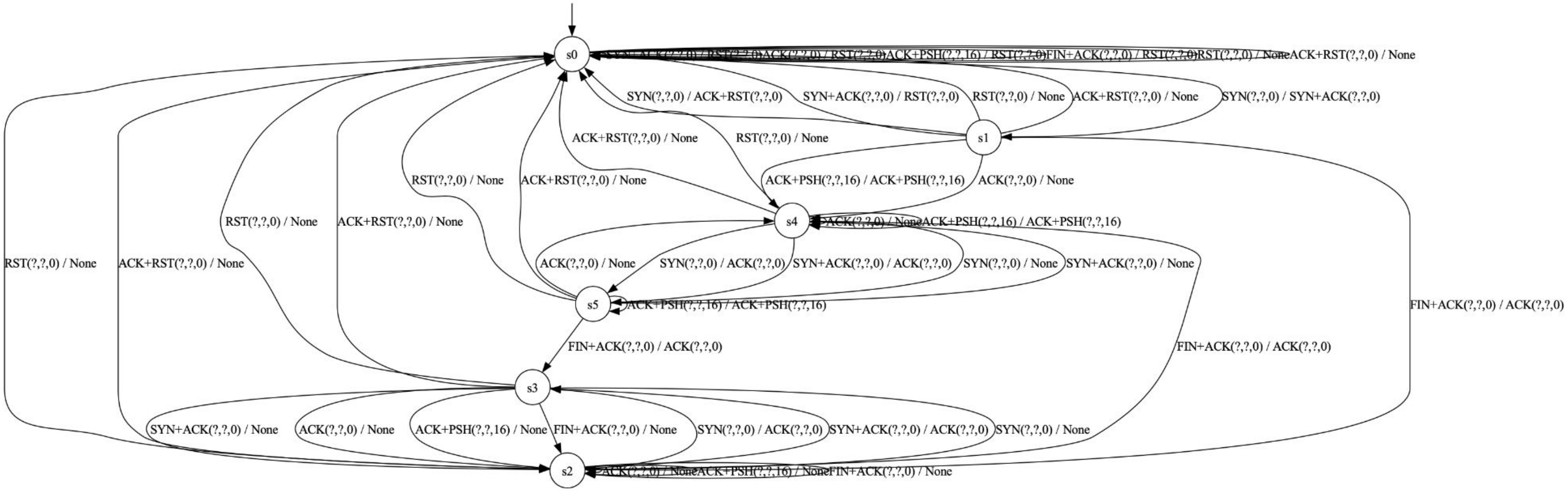
The TCP State Machine

- Defined in RFC 9293.
- Defines how implementations should behave according to the packets they receive.
- An idealised view that is often simpler than what happens in reality.
- Hard to implement right – procedural code is very different to graph-based automata.



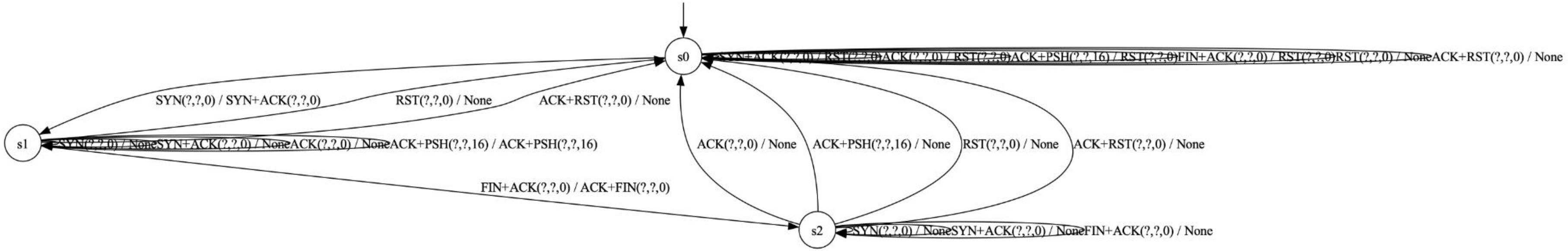
```
                            +---------+ ---------\      active OPEN
                            |  CLOSED |            \    -----------
                            +---------+<---------\   \   create TCB
                              |     ^              \   \  snd SYN
                 passive OPEN |     |   CLOSE        \   \
                 ------------ |     | ----------       \   \
                  create TCB  |     | delete TCB         \   \
                              V     |                      \   \
          rcv RST (note 1)  +---------+            CLOSE    |    \
       -------------------->|  LISTEN |          ---------- |     |
      /                     +---------+          delete TCB |     |
     /           rcv SYN      |     |     SEND              |     |
    /           -----------   |     |    -------            |     V
+--------+      snd SYN,ACK  /       \   snd SYN          +--------+
|        |<-----------------           ------------------>|        |
|  SYN   |                    rcv SYN                     |  SYN   |
|  RCVD  |<-----------------------------------------------|  SENT  |
|        |                  snd SYN,ACK                   |        |
|        |------------------           -------------------|        |
+--------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +--------+
   |         --------------   |     |   -----------
   |                x         |     |     snd ACK
   |                          V     V
   |  CLOSE                 +---------+
   | -------                |  ESTAB  |
   | snd FIN                +---------+
   |                 CLOSE    |     |    rcv FIN
   V                -------   |     |    -------
+---------+         snd FIN  /       \   snd ACK         +---------+
|  FIN    |<----------------          ------------------>|  CLOSE  |
| WAIT-1  |------------------                            |   WAIT  |
+---------+          rcv FIN  \                          +---------+
  | rcv ACK of FIN   -------   |                          CLOSE  |
  | --------------   snd ACK   |                         ------- |
  V        x                   V                         snd FIN V
+---------+               +---------+                    +---------+
|FINWAIT-2|               | CLOSING |                    | LAST-ACK|
+---------+               +---------+                    +---------+
  |              rcv ACK of FIN |                 rcv ACK of FIN |
  |  rcv FIN     -------------- |    Timeout=2MSL -------------- |
  |  -------            x       V    ------------        x       V
   \ snd ACK              +---------+delete TCB          +---------+
     -------------------->|TIME-WAIT|------------------->| CLOSED  |
                          +---------+                    +---------+
```

The (real) Linux TCP State Machine


s0
SYN+ACK(?,?,0) / RST(?,?,0)ACK(?,?,0) / RST(?,?,0)ACK+PSH(?,?,16) / RST(?,?,0)FIN+ACK(?,?,0) / RST(?,?,0)RST(?,?,0) / NoneACK+RST(?,?,0) / None
SYN(?,?,0) / ACK+RST(?,?,0)
SYN+ACK(?,?,0) / RST(?,?,0)
RST(?,?,0) / None
ACK+RST(?,?,0) / None
SYN(?,?,0) / SYN+ACK(?,?,0)
s1
ACK+RST(?,?,0) / None
RST(?,?,0) / None
RST(?,?,0) / None
ACK+RST(?,?,0) / None
ACK+PSH(?,?,16) / ACK+PSH(?,?,16)
ACK(?,?,0) / None
s4
ACK(?,?,0) / NoneACK+PSH(?,?,16) / ACK+PSH(?,?,16)
RST(?,?,0) / None
ACK+RST(?,?,0) / None
RST(?,?,0) / None
ACK+RST(?,?,0) / None
ACK(?,?,0) / None
SYN(?,?,0) / ACK(?,?,0)
SYN+ACK(?,?,0) / ACK(?,?,0)
SYN(?,?,0) / None
SYN+ACK(?,?,0) / None
s5
ACK+PSH(?,?,16) / ACK+PSH(?,?,16)
FIN+ACK(?,?,0) / ACK(?,?,0)
FIN+ACK(?,?,0) / ACK(?,?,0)
FIN+ACK(?,?,0) / ACK(?,?,0)
s3
SYN+ACK(?,?,0) / None
ACK(?,?,0) / None
ACK+PSH(?,?,16) / None
FIN+ACK(?,?,0) / None
SYN(?,?,0) / ACK(?,?,0)
SYN+ACK(?,?,0) / ACK(?,?,0)
SYN(?,?,0) / None
s2
ACK(?,?,0) / NoneACK+PSH(?,?,16) / NoneFIN+ACK(?,?,0) / None

Distribution Statement A: Approved for public release. Distribution is unlimited.

The smoltcp TCP State Machine



Distribution Statement A: Approved for public release. Distribution is unlimited.

Protocol Violations

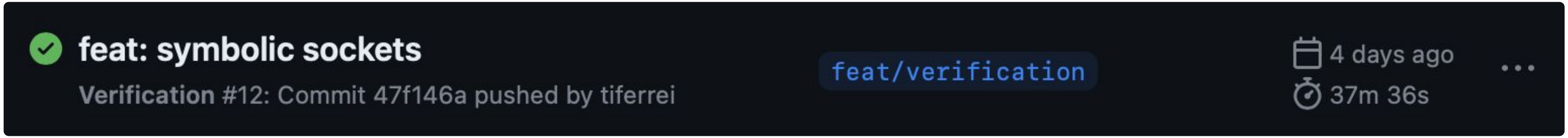
1. **RST on repeated SYN:** smoltcp fails to reset the connection when repeated SYN packets are sent. It instead silently drops the repeated packets.

2. **Data carrying SYN:** The specification allows for data transmission on synchronize packets. This data should be buffered and delivered after the handshake completes. smoltcp drops the data instead.

3. **Sending RST not resetting state:** When smoltcp sends a reset packet, it does not reset its own state, instead resetting only the client. This is has so far not been manifested as an issue due to smoltcp's collapsed states.

Kani

- Developed by Amazon, similar to Crux-MIR - https://crux.galois.com
- Performs complete model checking of program properties through symbolic execution.
- Allows us to prove correctness of finer details such as packet handling.
- Runs in a CI environment ensuring that proofs stay valid on every new commit.
- So far, we have proved the packet parsing and construction parts of TCP.



✓ **feat: symbolic sockets**

Verification #12: Commit 47f146a pushed by tiferrei

`feat/verification`

4 days ago

37m 36s

Summary & next steps

- **Prognosis**
  - 5 protocol violations found
  - responsible disclosure, patches in the works
- **Kani**
  - proven round-trip property of TCP packets
  - Continuous Verification
- **Future work:**
  - apply Prognosis to other protocols (DHCP, DNS, TLS, …)
  - increase coverage with symbolic execution (ideally 100%)
  - the go-to network stack for seL4?

galois