Fuzzing the MCU of Connected Vehicle for Security and Safety

Hao Chen- Security Expert – Li Auto

About Me

Hao Chen(@flankersky)

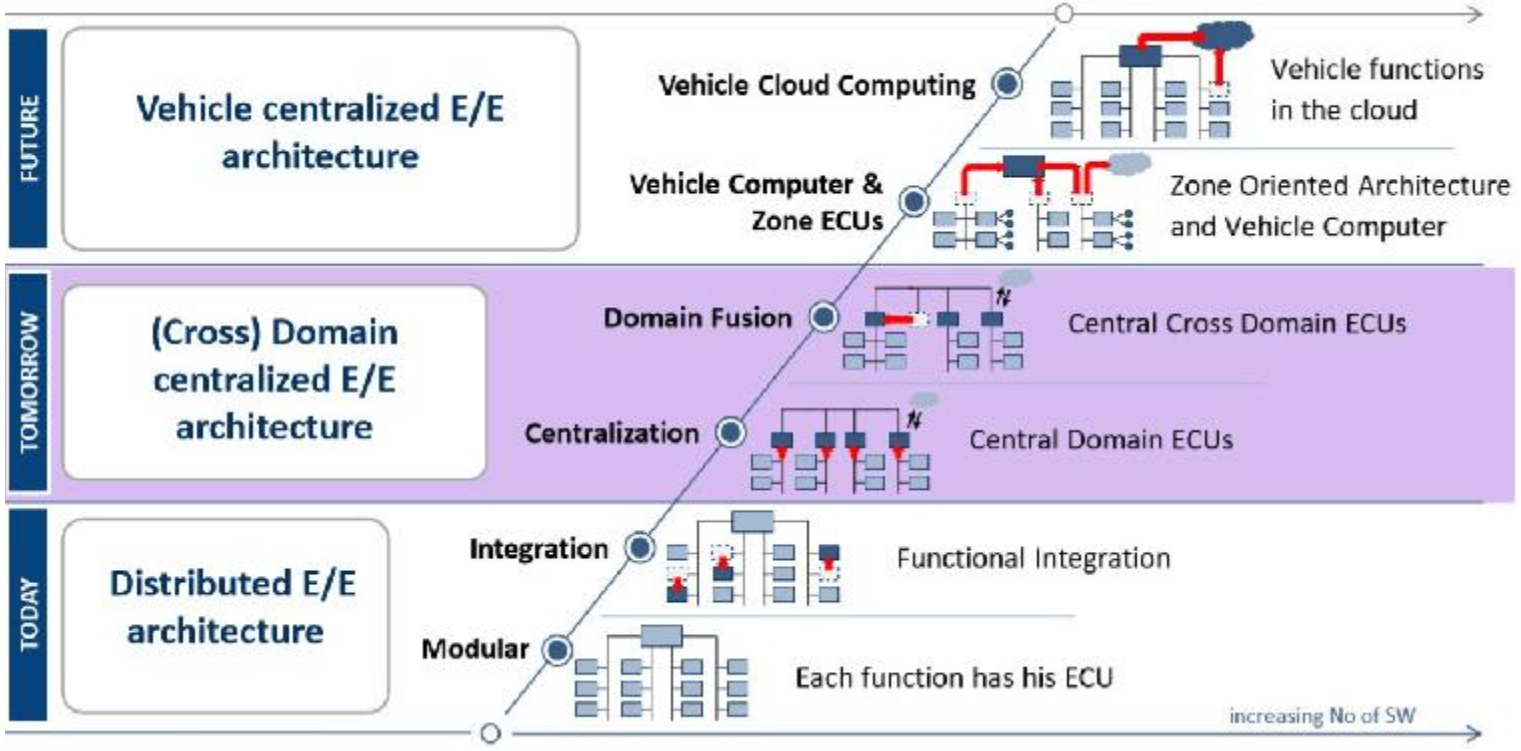
- Security Expert @Li Auto
- Bug hunting in Android, Linux kernel
- Connected Car Security & Hardware Security Newbie

Overview

- BackGround
- McuFuzz Design
- McuFuzz Demo
- Conclusion

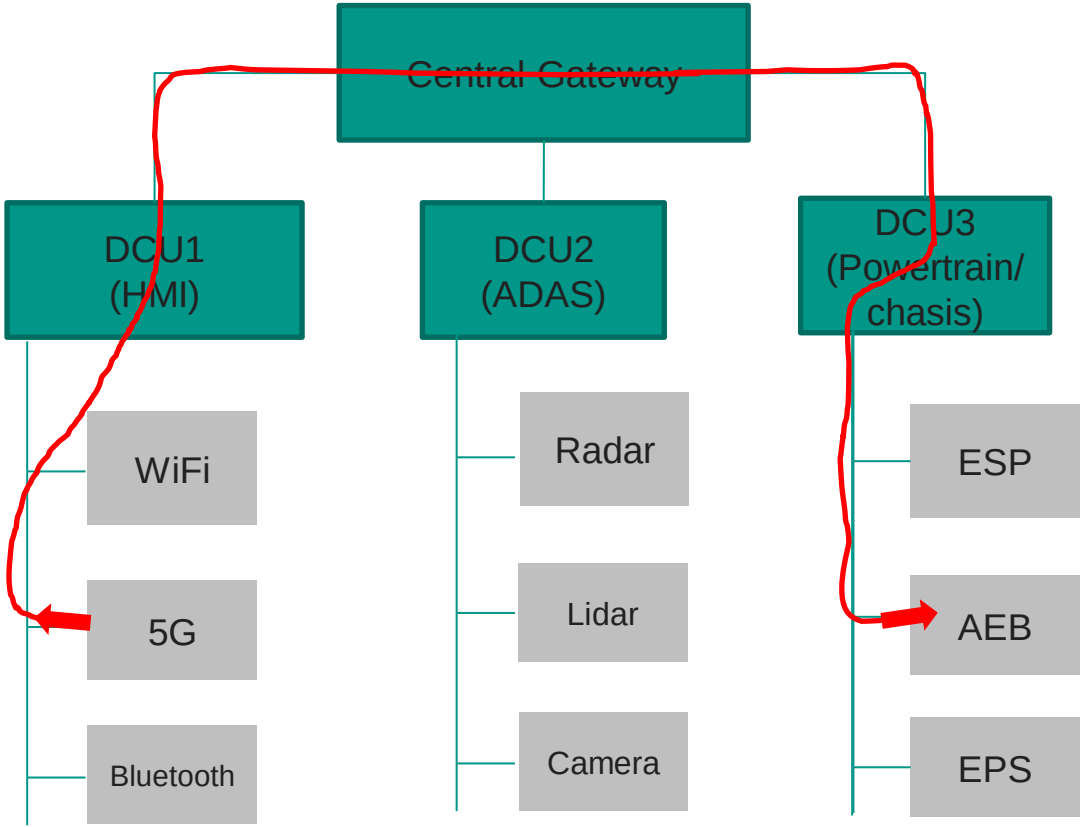
BackGround

BackGround - The usual attack vector



https://www.researchgate.net/figure/Possible-evolution-of-vehicular-E-E-architectures-3_fig1_348825146

BackGround - The usual attack vector

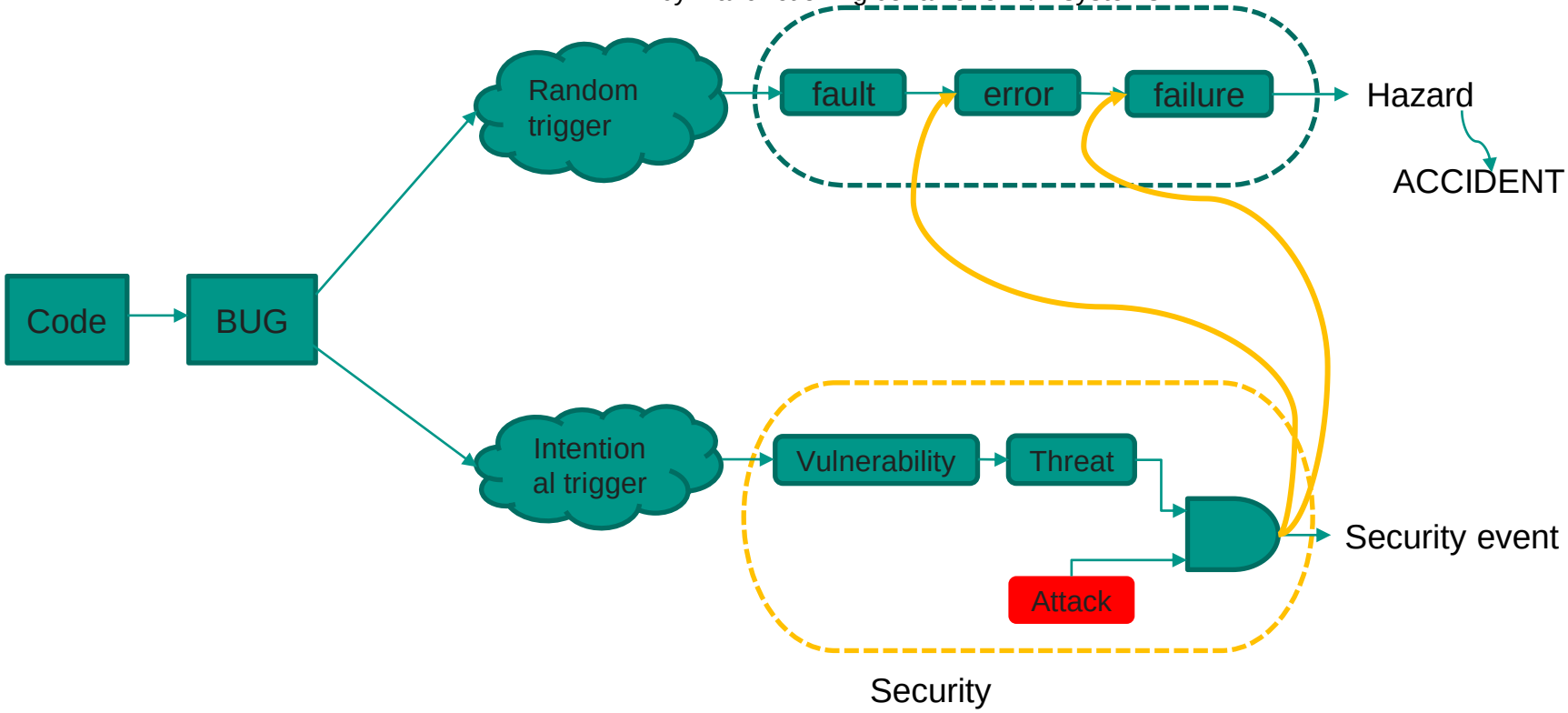


The Powertrain/chasis Domain:

- Hacker's ultimate goal
- Safety Critical
- Security Critical

BackGround – Security vs Safety(software)

Safety:Absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems



BackGround -Current MCU software Test

- Code Walk-through
- Semi-formal verification
- Formal verification
- Interface test
- Unit test
- Fault injection test
- Static code analysis
- Data flow analysis

- statement coverage
- branch coverage
- MC/DC

Coverage-guided fuzzing maybe helpful.

There are never enough ways to test.

The Mcu Fuzzing

McuFuzz – What we have & need

Resources of a processor for Vehicles

- Up to 3x Arm Cortex-M7 DCLS
- Up to 8 MB SRAM
- Runing AUTOSAR or FreeRTOS

Coverage guided?

- How to trace memory access
- How to trace code coverage

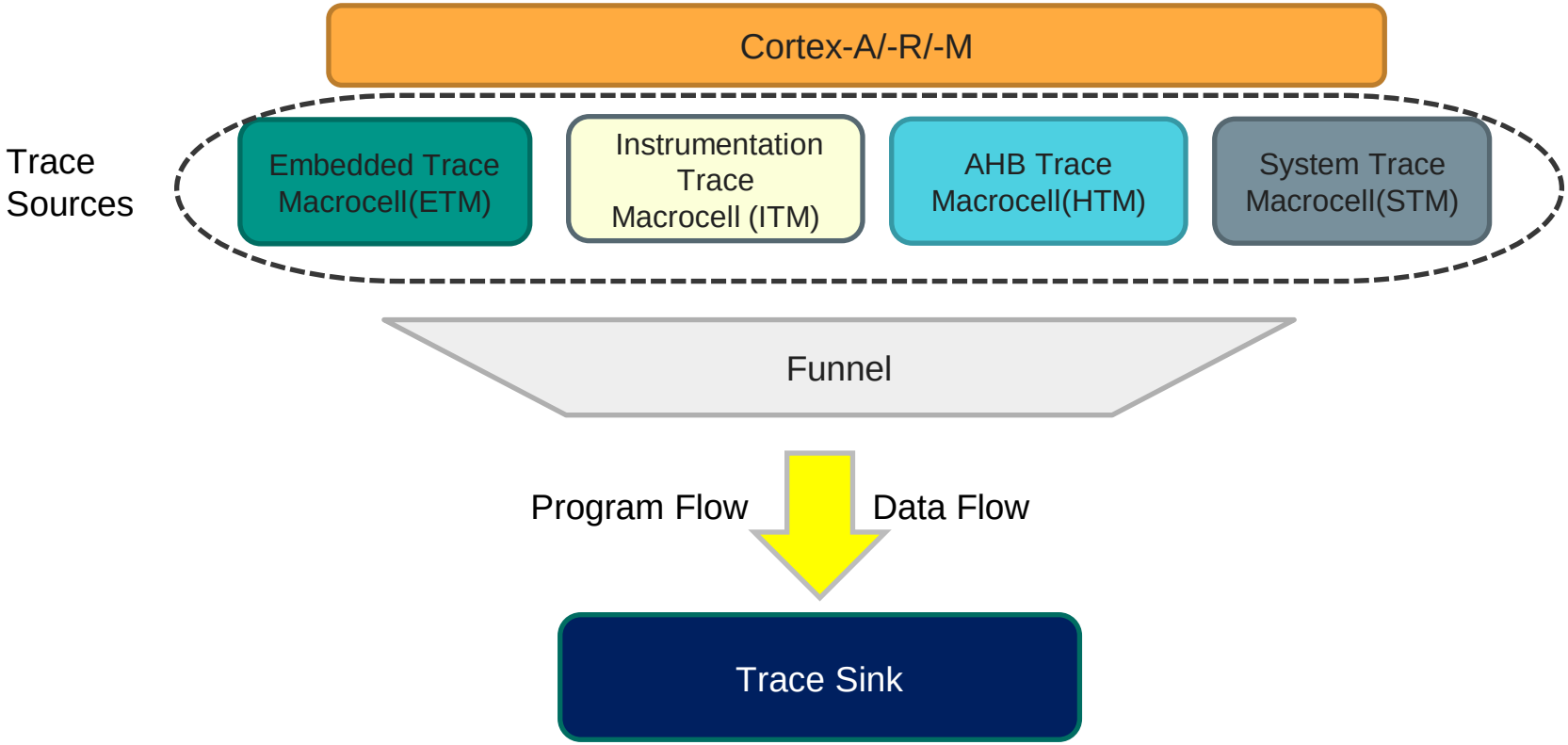
What's the problem

- SRAM is really small
- real-time OS, no MMU
- Gcc or Clang are rarely used

Solution

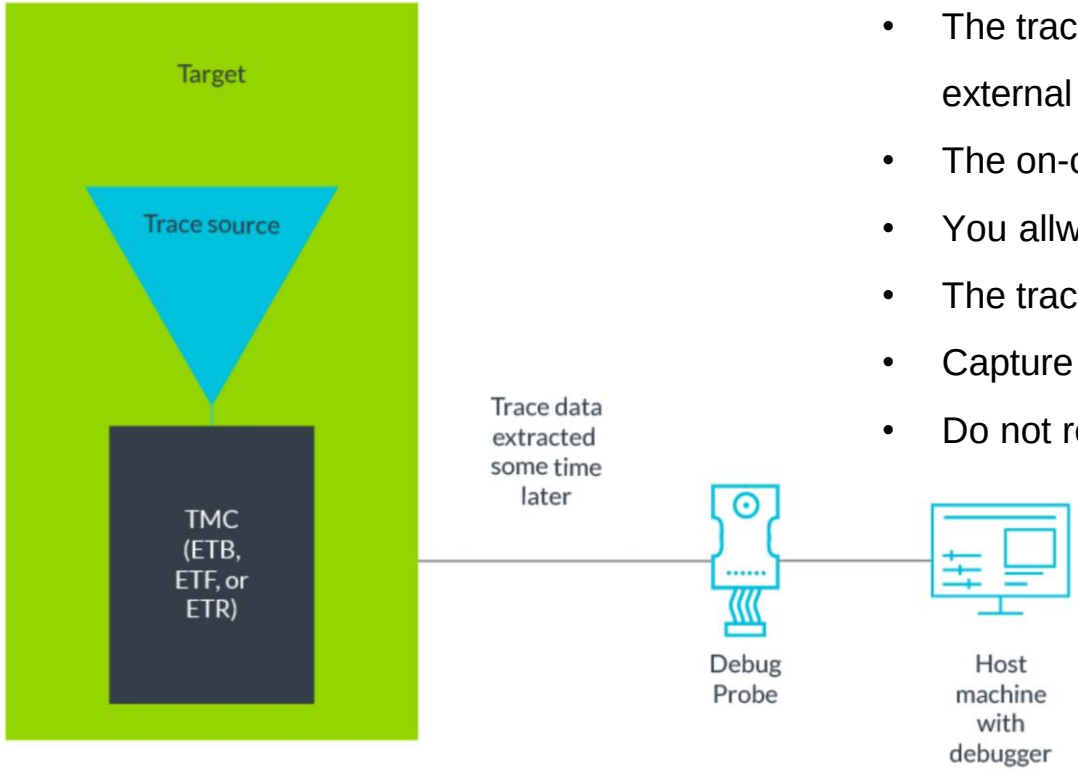
Hardware Assisted

McuFuzz - Introduction to ARM Trace



https://developer.arm.com/documentation/dgi0012/d/CoreSight-Components-and-Systems/CoreSight-components/Trace-sources

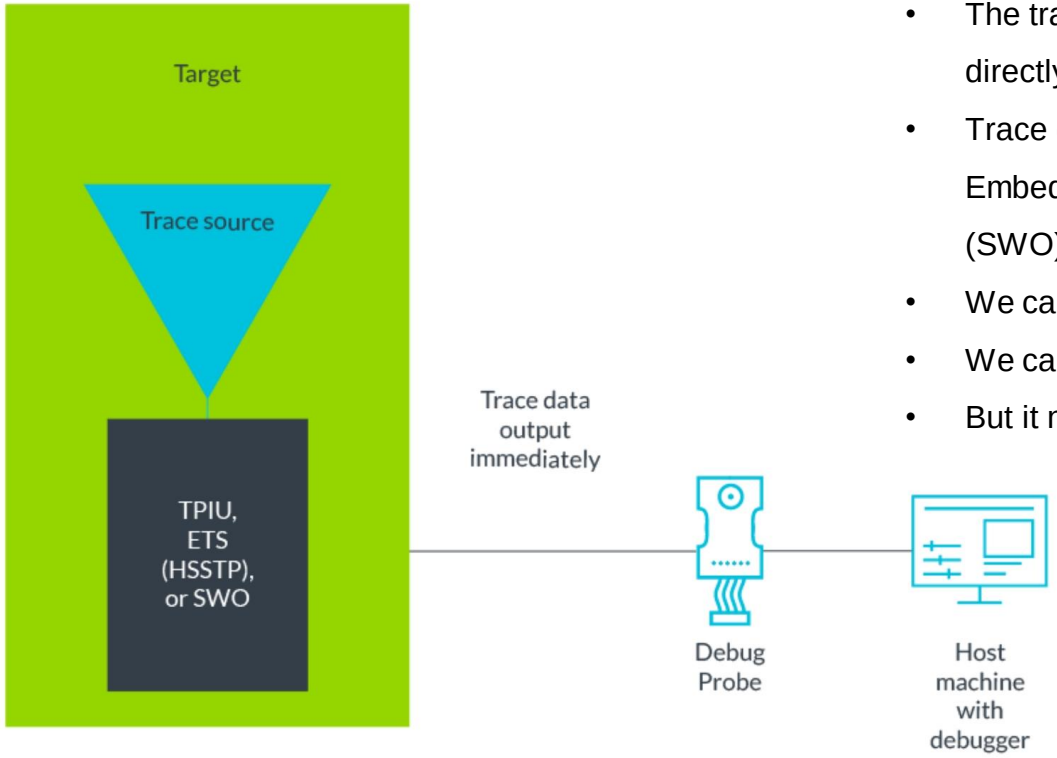
McuFuzz – ETM on-chip trace



- The trace data is on chip and is exported to the external debugger.
- The on-chip buffer is usually small.
- You allways need filtering.
- The trace data is heavily compressed.
- Capture trace at a much higher speed.
- Do not require any trace pins,JTAG is enough.

https://developer.arm.com/documentation/102119/latest/

McuFuzz – ETM off-chip trace



- The trace data is output from the target to a debug unit or directly to the external debugger.
- Trace data is output by the Trace Port Interface Unit (TPIU), Embedded Trace Streamer (ETS), or Serial Wire Output (SWO) that is on the target to an external debugger.
- We can have more buffer to store trace data.
- We can trace over a long period.
- But it need additional hardware pins.

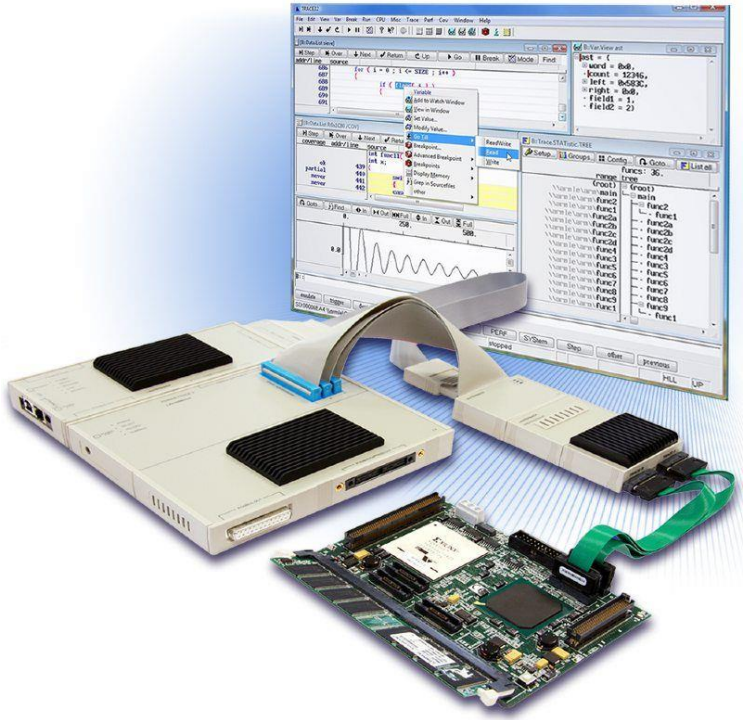
https://developer.arm.com/documentation/dgi0012/d/Trace-Capture/Designing-your-trace-system/Differences-between-on-chip-and-off-chip-storage

McuFuzz - ETM's features Summary

- A trace source,part of ARM coresight
- Instruction and data trace
- ETM supports trace filtering
- Can generate cycle-accurate trace
- Can insert timestamps into trace data
- Support on-chip and off-chip trace
- Supported in most Arm-based systems

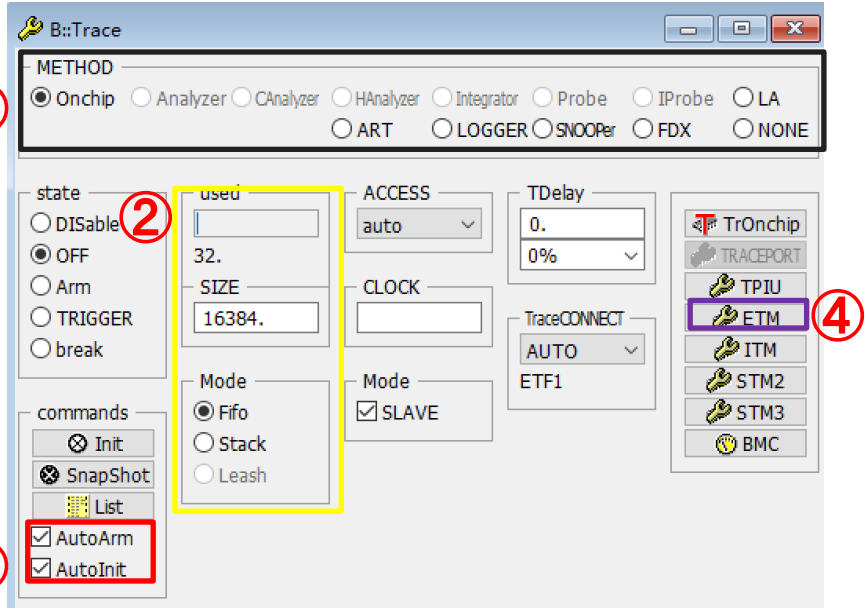
McuFuzz– Use Trace32 to trace

- PowerView, a universal GUI
- PowerDebug tools for debugging
- PowerTrace tools for program/data flow trace
- Support Cortex-A/-R/-M, TriCore, RISC-V, Power Architecture



https://www.lauterbach.com/frames.html?home.html

McuFuzz – Trace32: Enable ETM



- **Trace method**
  Trace sink is on-chip buffer

- **On-chip trace buffer**
  Trace buffer usage status
  Fifo mode: If the trace is full, new records will overwrite older records.

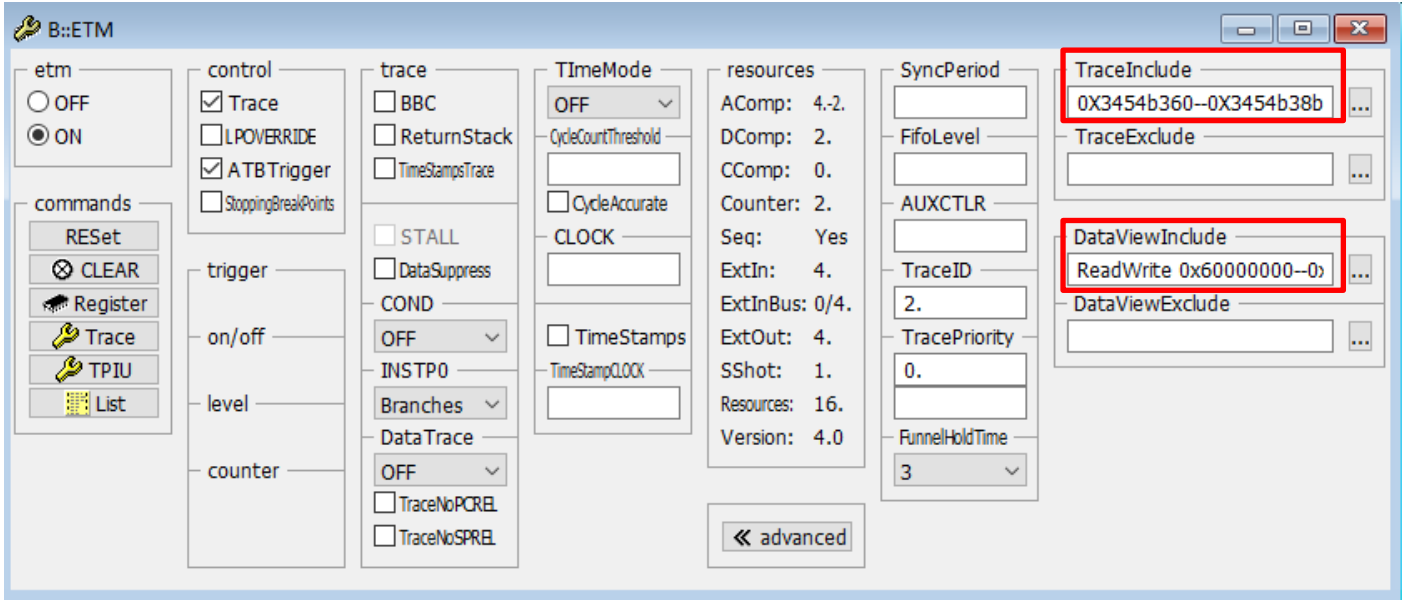
- AutoInit
  - Trace memory contents is erased and previous records are no longer visible
  - The trigger unit is set to its initial state.

- AutoArm
  - Recording and if available triggering is prepared whenever the program execution is started.
  - Recording and if available triggering is stopped whenever the program execution is stopped.

- ETM

https://www2.lauterbach.com/pdf/trace_arm_etm.pdf

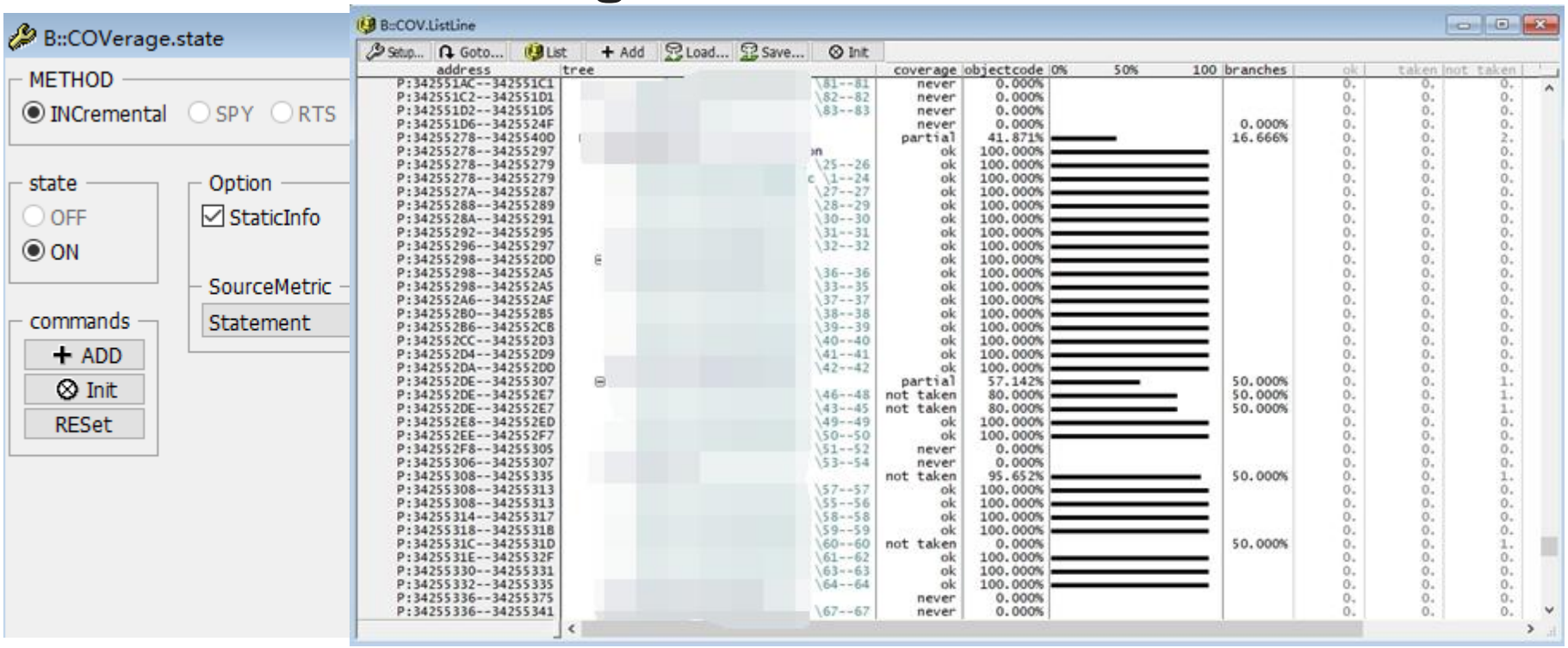
McuFuzz – ETM trace filter



- Code Filter example
  ETM.TraceInclude Execute 0x34000000-- 0x34000fff 0x35000000-- 0x35000fff

- Memory access filter example
  ETM.DataViewInclude ReadWrite 0x60000000--0x61ffffff

McuFuzz – The coverage result

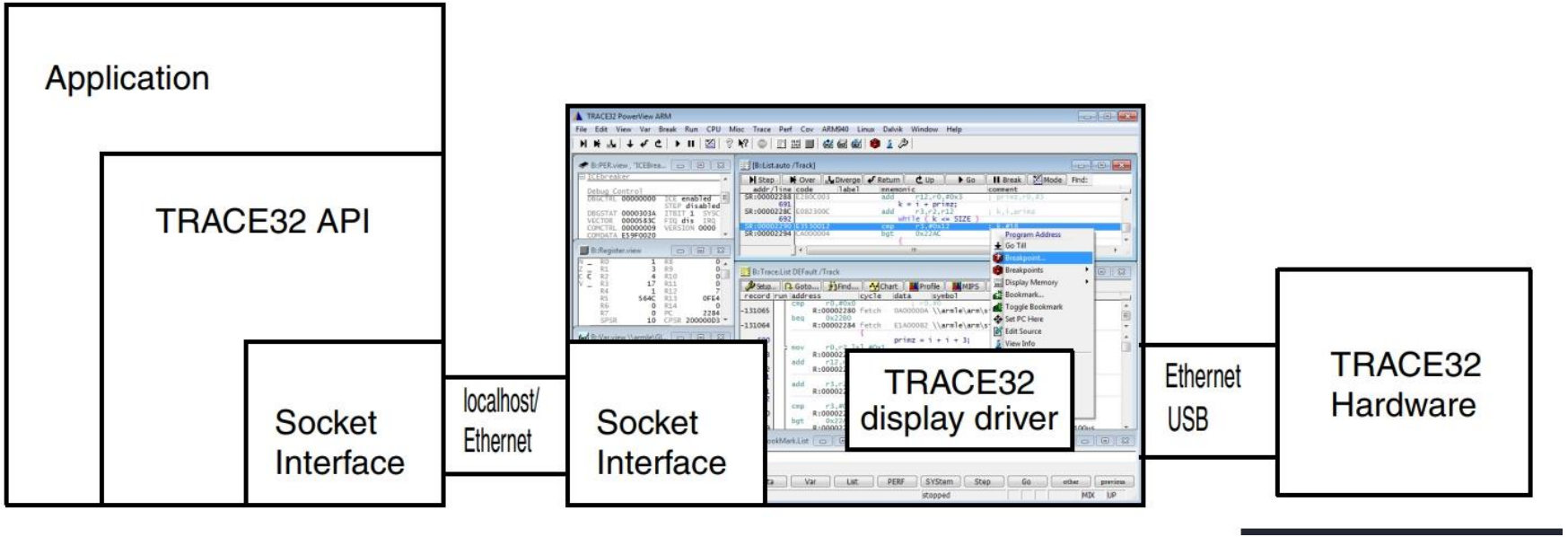


This will slow down the fuzzing speed.

https://www2.lauterbach.com/pdf/app_code_coverage.pdf

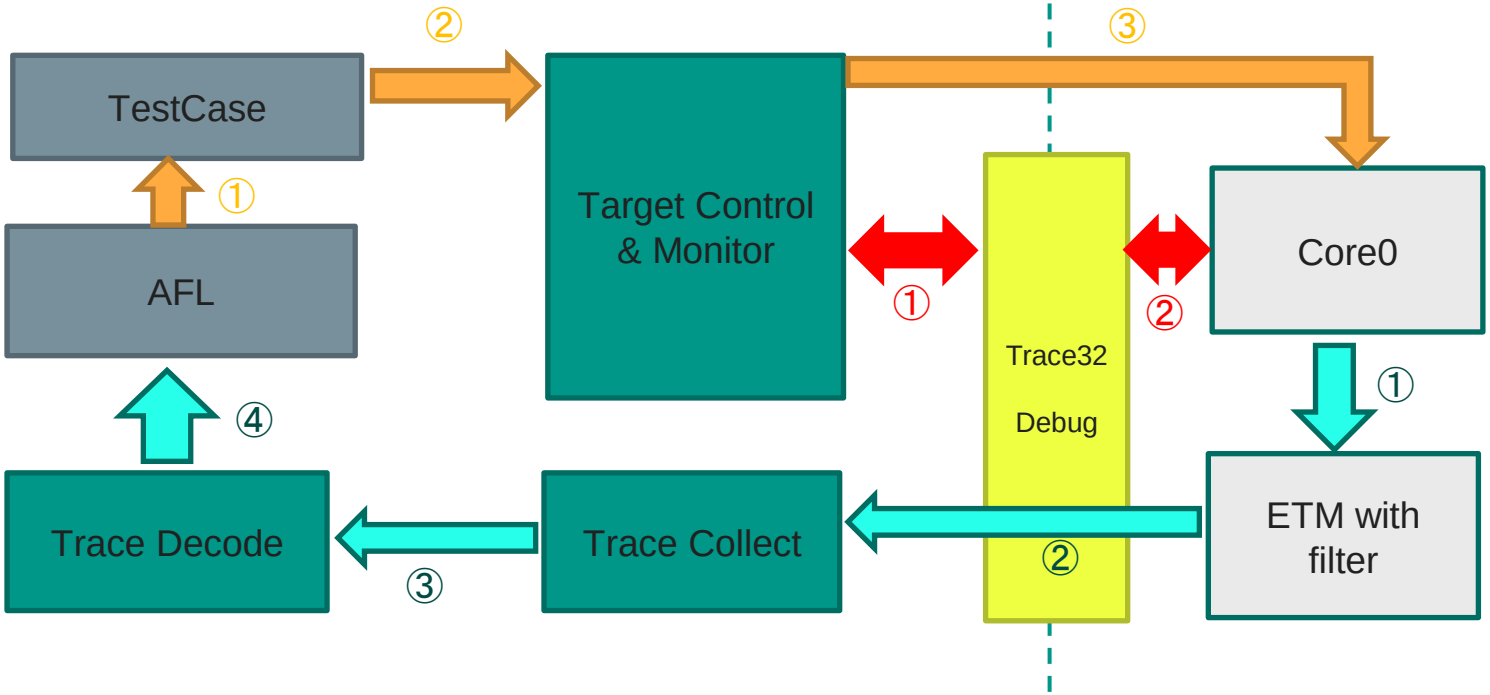
McuFuzz – Trace32 API

Application ---> TRACE32 API ---> TRACE32 application --> TRACE32
(C Functions) (sockets) (HW interface)



https://www2.lauterbach.com/pdf/api_remote_c.pdf

McuFuzz– The mcu fuzzing framework



McuFuzz – The advantages

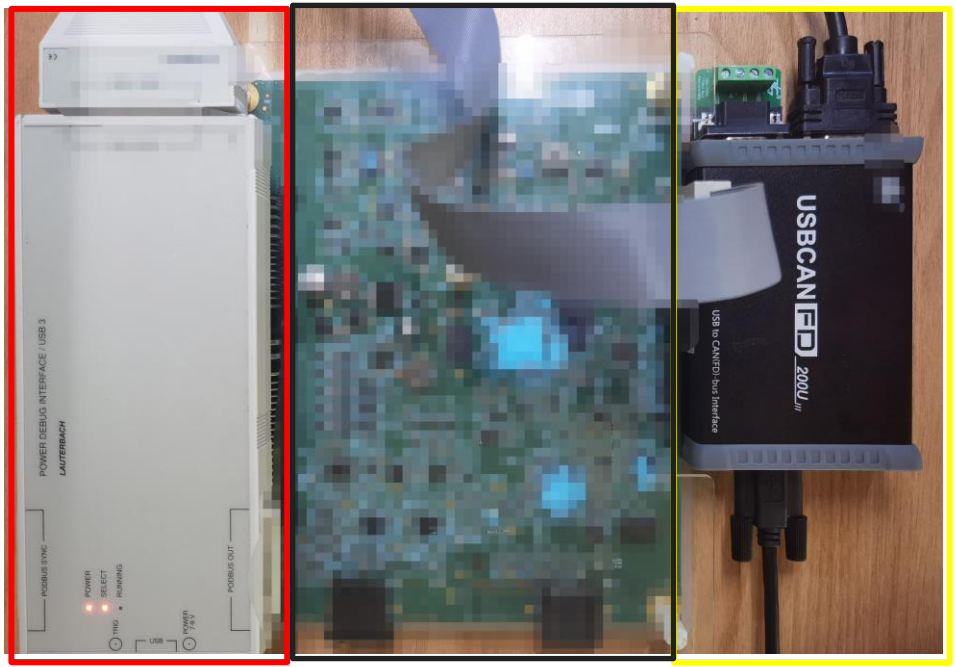
- Compiler independent
- No need to recompile code
- No code instrumentation required
- Coverage-guided

Demo

Demo – Can service Fuzzing

Trace32 PowerDebug

- Control Target: Run,break,configure ETM
- Get ETM analyzed data
- Monitor crash

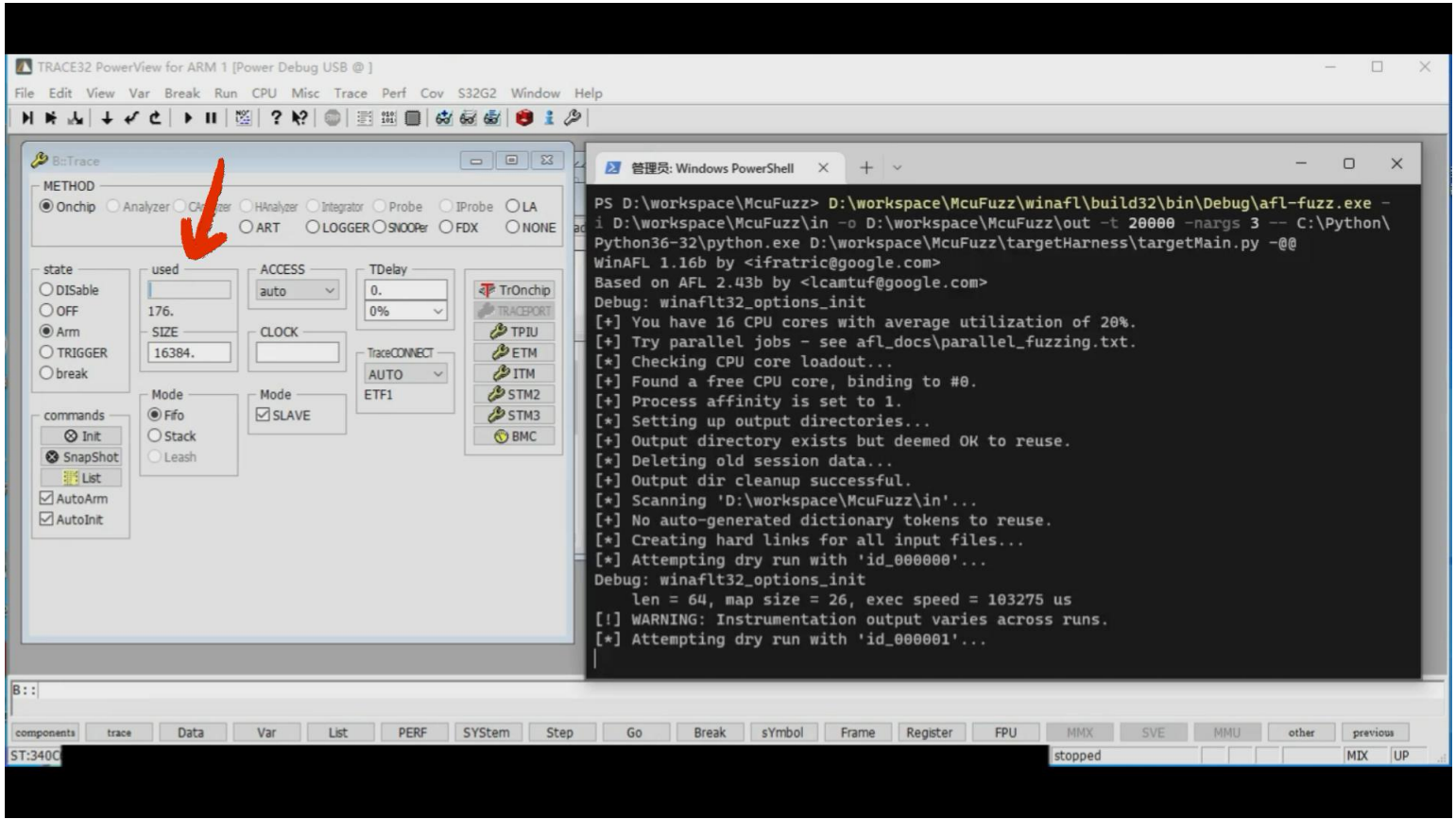


Target MCU

USBCAN Device

- Collect init seed corpus
- Send mutated data to target

Demo - Can service Fuzzing



Conclusion

Conclusion

- Coverage guided fuzzing on MCU is possible
- ETM and Trace32 is really helpfull
- This prototype is proven effective in our product

Future works:

- Improve fuzzing speed
- More target fuzzing practice
- Off-chip trace is in progress

https://github.com/flankersky/mcufuzz

flank3rsky@gmail.com

Thank You!