# Improving Security in RTEMS and EPICS IOC Application

By

**Vijay Banerjee, Uchenna Ezeobi**

**Advisor: Gedare Bloom**

Department of Computer Science

University of Colorado Colorado Springs

# Outline



- Motivation
- Modularization of the Software Stack
- Static Analysis
- Fuzzing
  - Model inference with Fuzzing
  - Machine Learning Approach
- Improving Security for RTEMS-EPICS integration
- Conclusion

# Why security?



- Industrial control systems (ICS) are an integral part of *Critical Infrastructure.*

# Why security?



- Industrial control systems (ICS) are an integral part of *Critical Infrastructure.*

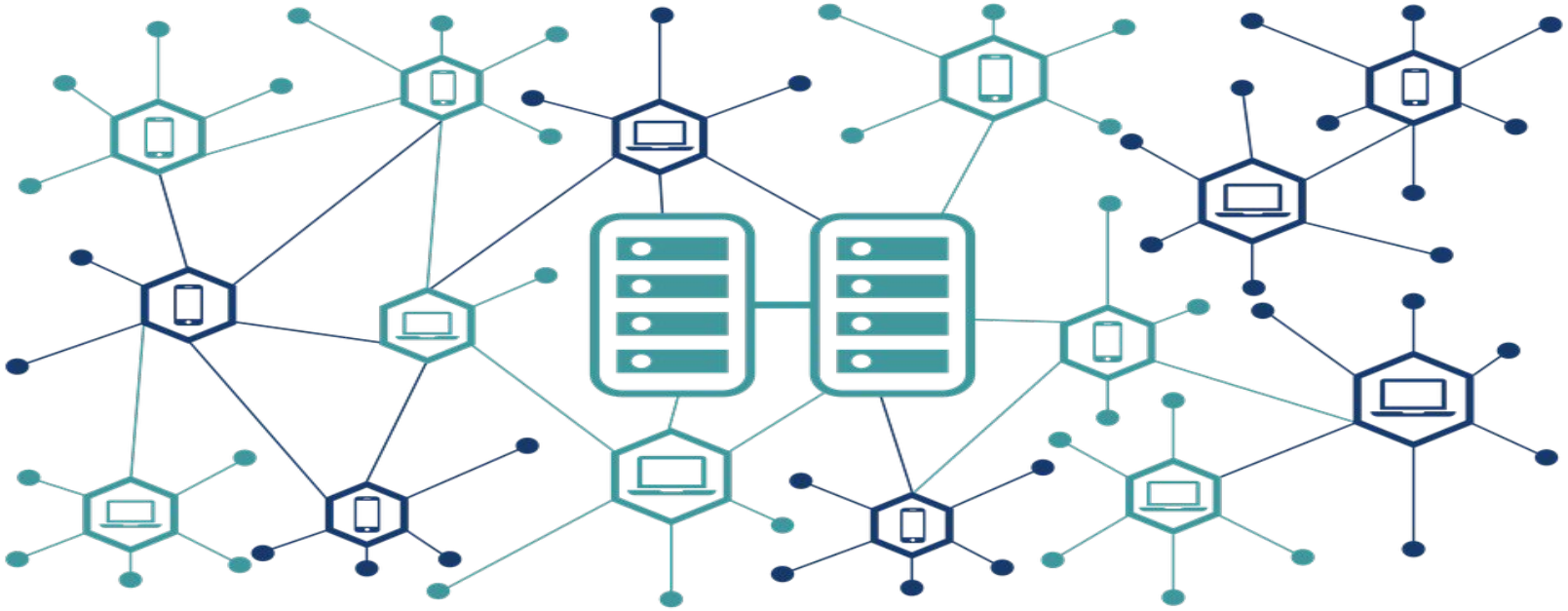
- High attack motivation

**source**: Google Images

# Why security?

- Industrial control systems (ICS) are an integral part of *Critical Infrastructure.*

- High attack motivation
- Large attack surface



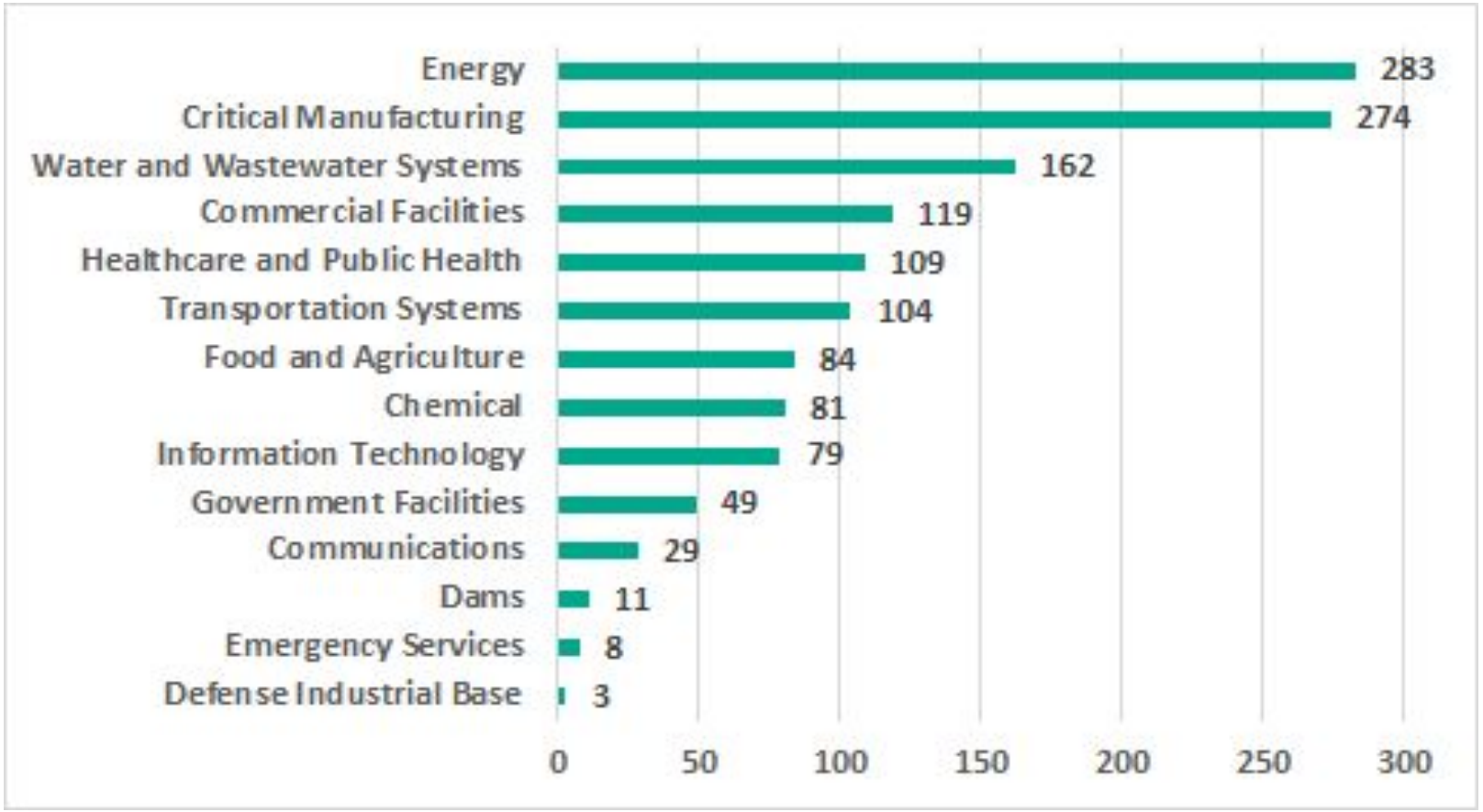
**source**: Google Images

# Why security?

- Industrial control systems (ICS) are an integral part of *Critical Infrastructure.*

- High attack motivation
- Large attack surface
- Insecure design

**Number of vulnerable components used in different industries**




**source**: Kaspersky, I. C. S. (2020). Threat landscape for industrial automation systems H2 2019

Real-Time Executive for Multiprocessor Systems (**RTEMS)** is a POSIX compliant hard Real-Time Operating System (RTOS).



RTEMS is used in the RTOS layer for EPICS systems. The security of EPICS-based systems also depend on the security of the base OS.

RTEMS consists of over **1 million**[1] SLOC. Over **91%** of the total code is in **C**.



Distributed control systems using EPICS can have **hundreds** of Interconnected nodes.

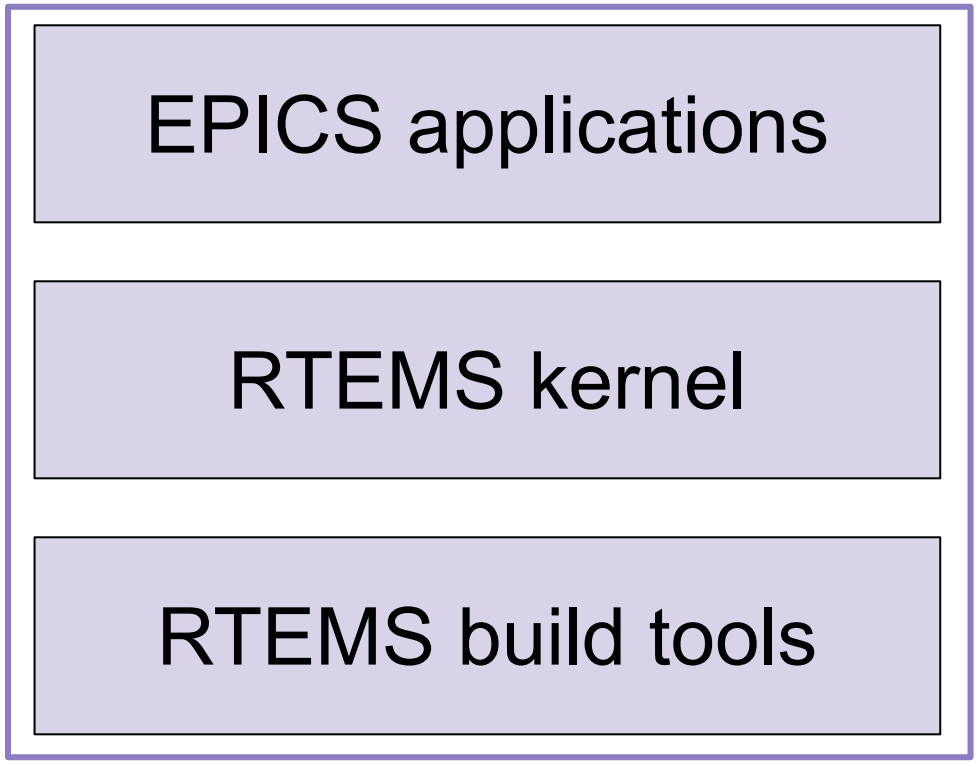
**1:** Generated using David A. Wheeler's 'SLOCCount'.

# Where do we start?

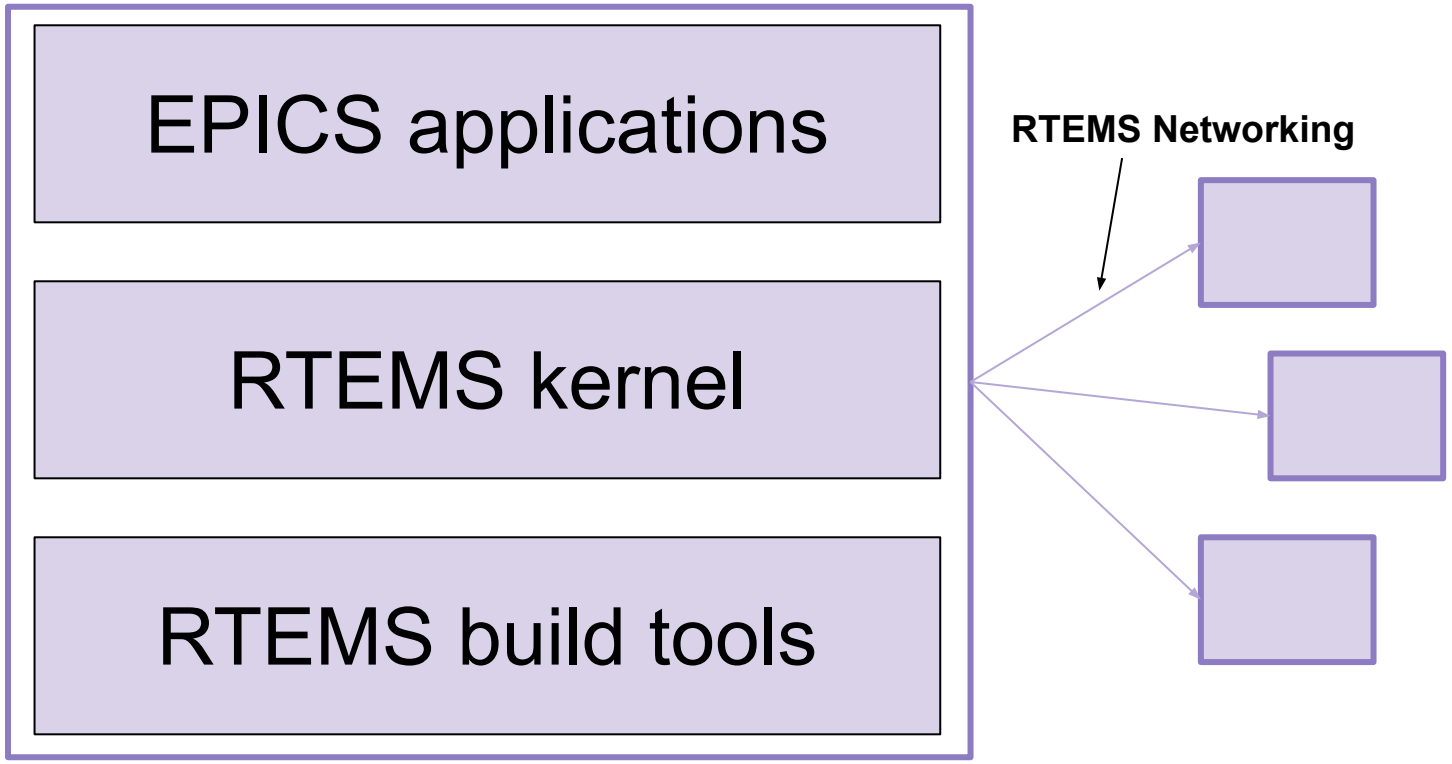
# EPICS + RTEMS: Vertical stack





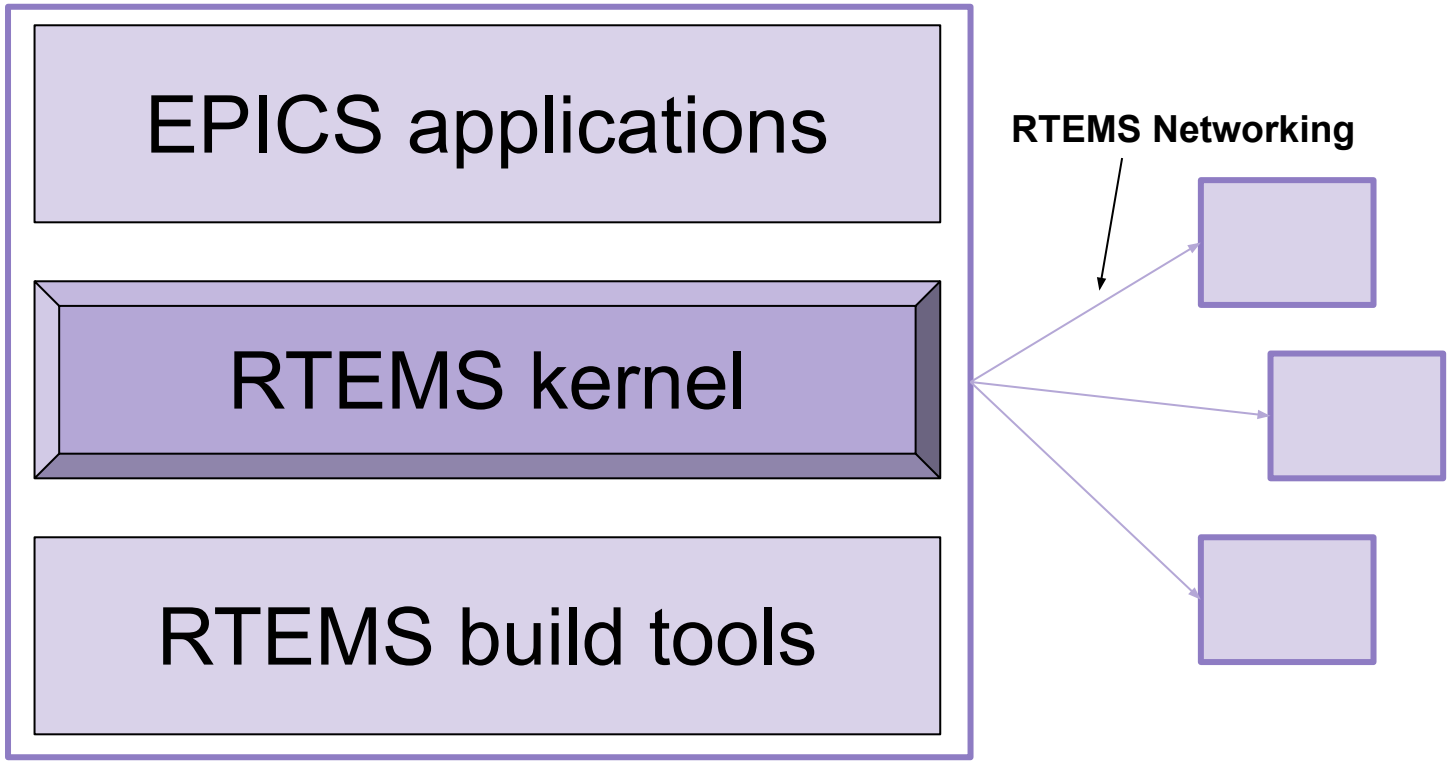


# EPICS + RTEMS: Vertical stack



EPICS


EPICS applications
RTEMS kernel
RTEMS build tools
RTEMS Networking

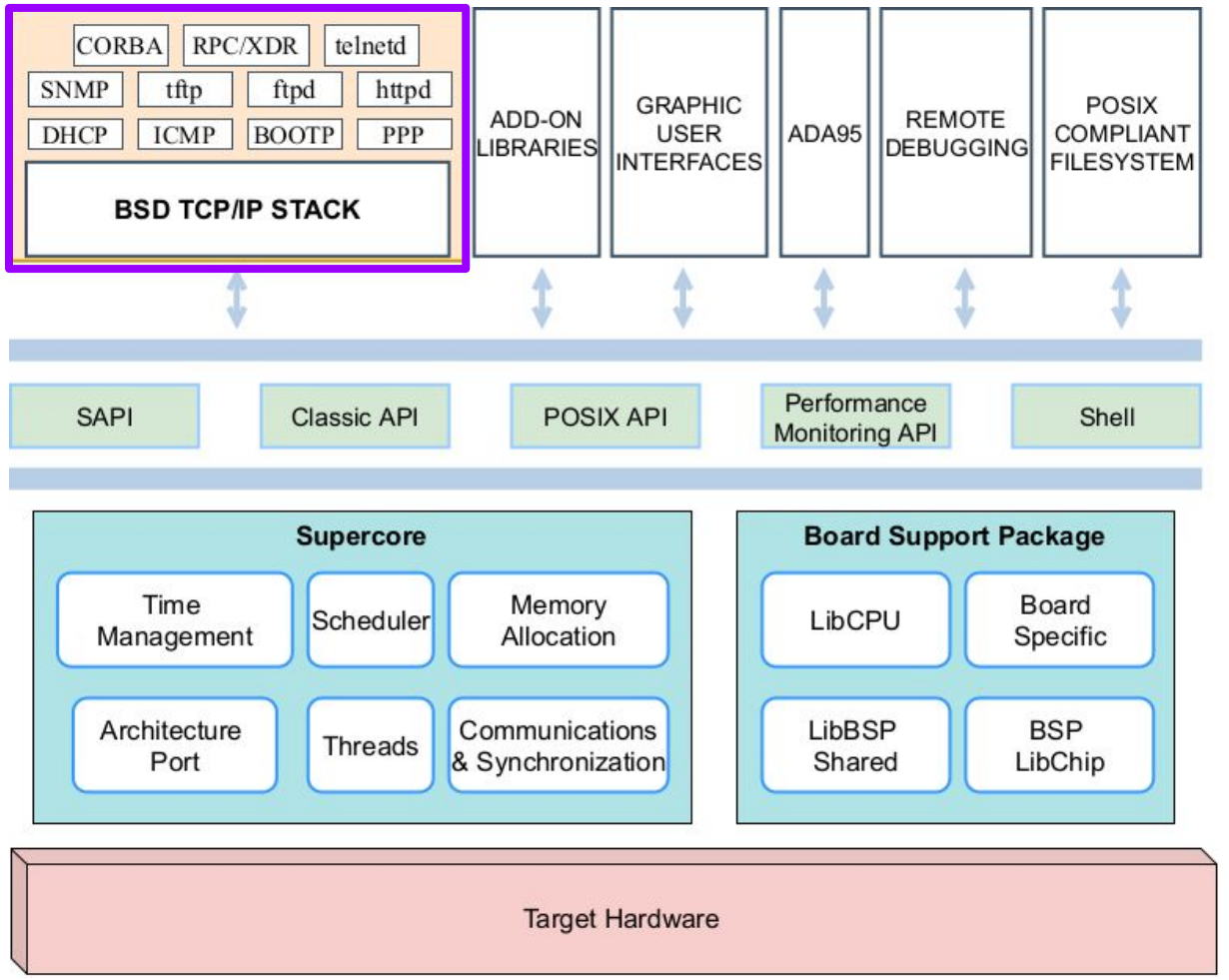

Embedded With
RTEMS
www.rtems.org

# Security in RTEMS


- Modularizing the Network stack
- Online Recovery

# RTEMS (legacy) High-level Architecture

- In RTEMS, the network implementation is derived from an old BSD network stack
  - This default network implementation is now called the legacy stack.



# Problems with the legacy implementation

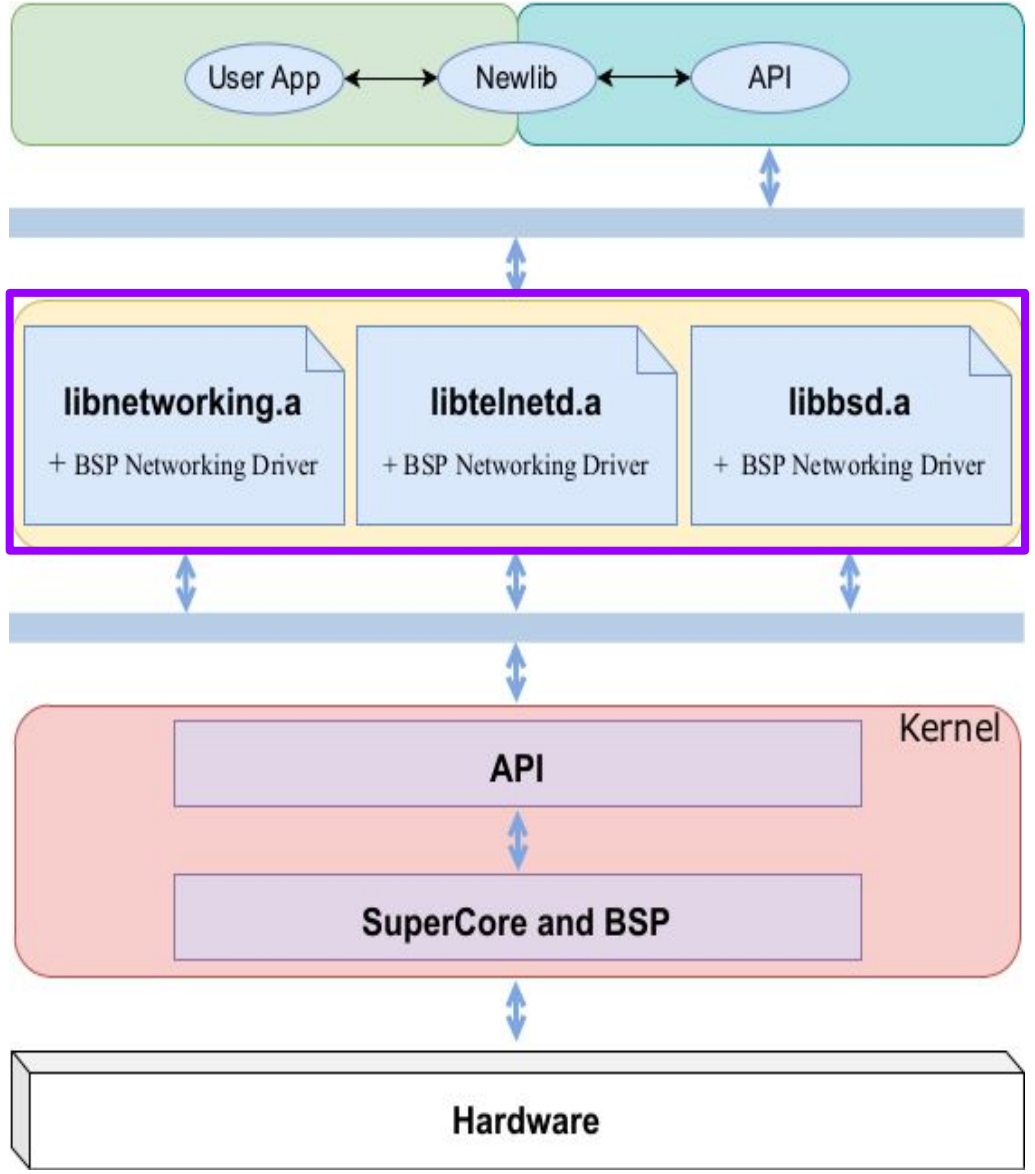


- Difficult to update the network stack
- Legacy stack is a derivative of an old BSD stack from the late 90s and doesn't support IPv6
- USA Govt. Memo #M-21-07: Phase out IPv4 and transition to IPv6

# Networking-as-a-Library: Design



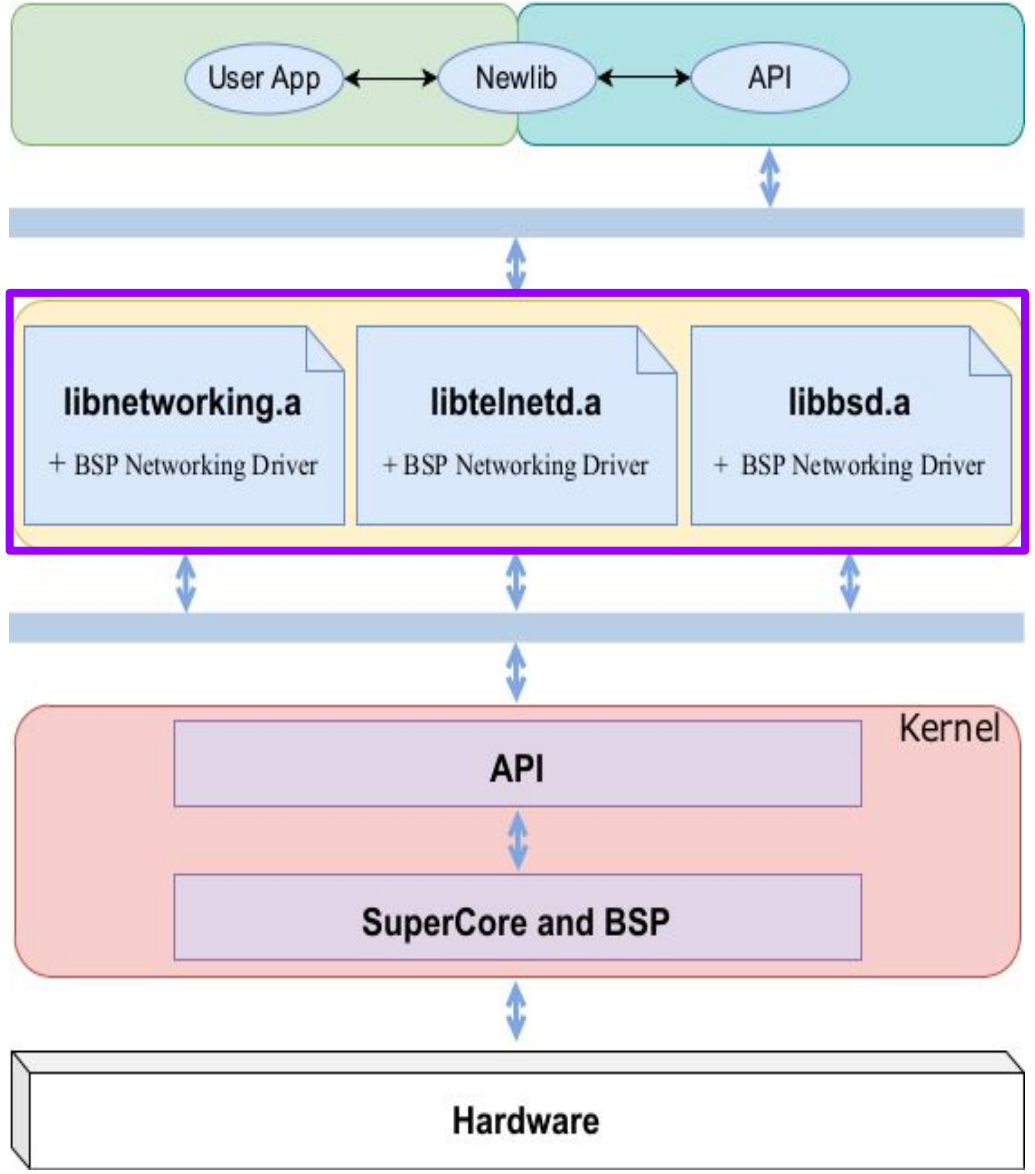
- Networking stack is built into a static library
- Testing and patching is simpler
- Targets supported by legacy stack can keep using it.



# Networking-as-a-Library: Effort



- Migrated about **270,000 SLOC**
- Created an independently hosted repository for legacy network stack.
- Added new LWIP network stack through “rtems-lwip”



# Networking-as-a-Library: Ongoing works



- Developing standalone “Networking services” repository. (Kinsey Moore and Chris Johns)
- Shift to modern libbsd stack.
- More supported targets are being added to LWIP repository

# Online Security: Challenges



- Overhead
- Difficult to implement
- Needs to meet real-time guarantees

# Online security through secure reboots



- Secure boots are popular in non-real-time domains
- Periodic integrity checking prevents persistent attacks
- Can be implemented with popular bootloaders

# Implementation using UBoot



- Wide array of supported targets
- Tested with RTEMS on BeagleboneBlack.
- Open Source!

# Secure Reboot efforts

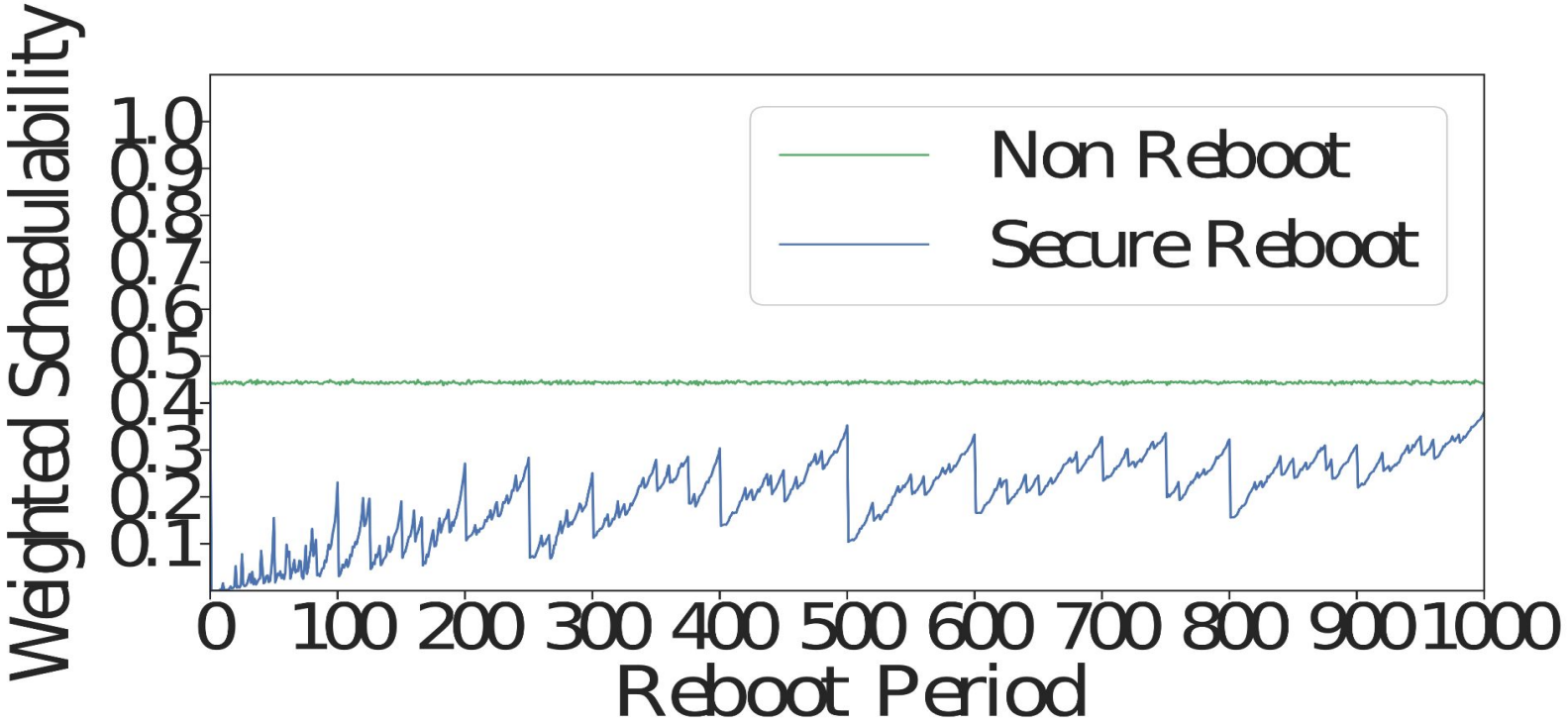


- Derived inequalities to find formal bounds on feasible overheads and periods

# Secure Reboot efforts



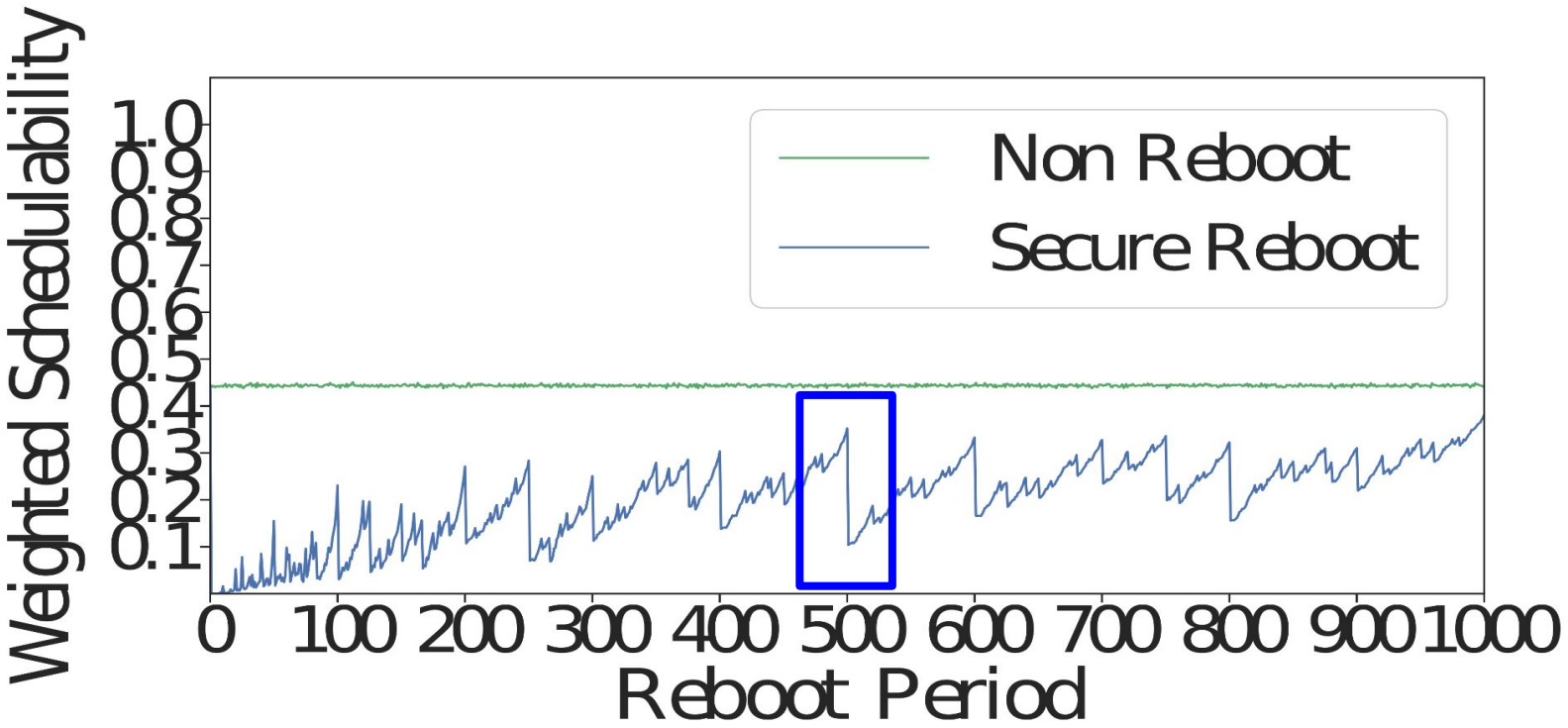
- Derived inequalities to find formal bounds on feasible overheads and periods
- Designed synthetic experiments to assess the impact.



# Secure Reboot efforts



- Derived inequalities to find formal bounds on feasible overheads and periods
- Designed synthetic experiments to assess the impact.



# Secure Software Development Lifecycle

- Apply Static Analysis to EPICS
- Apply Security Fuzz Testing To EPICS
- Testing EPICS-RTEMS integration

# Static Analysis For EPICS

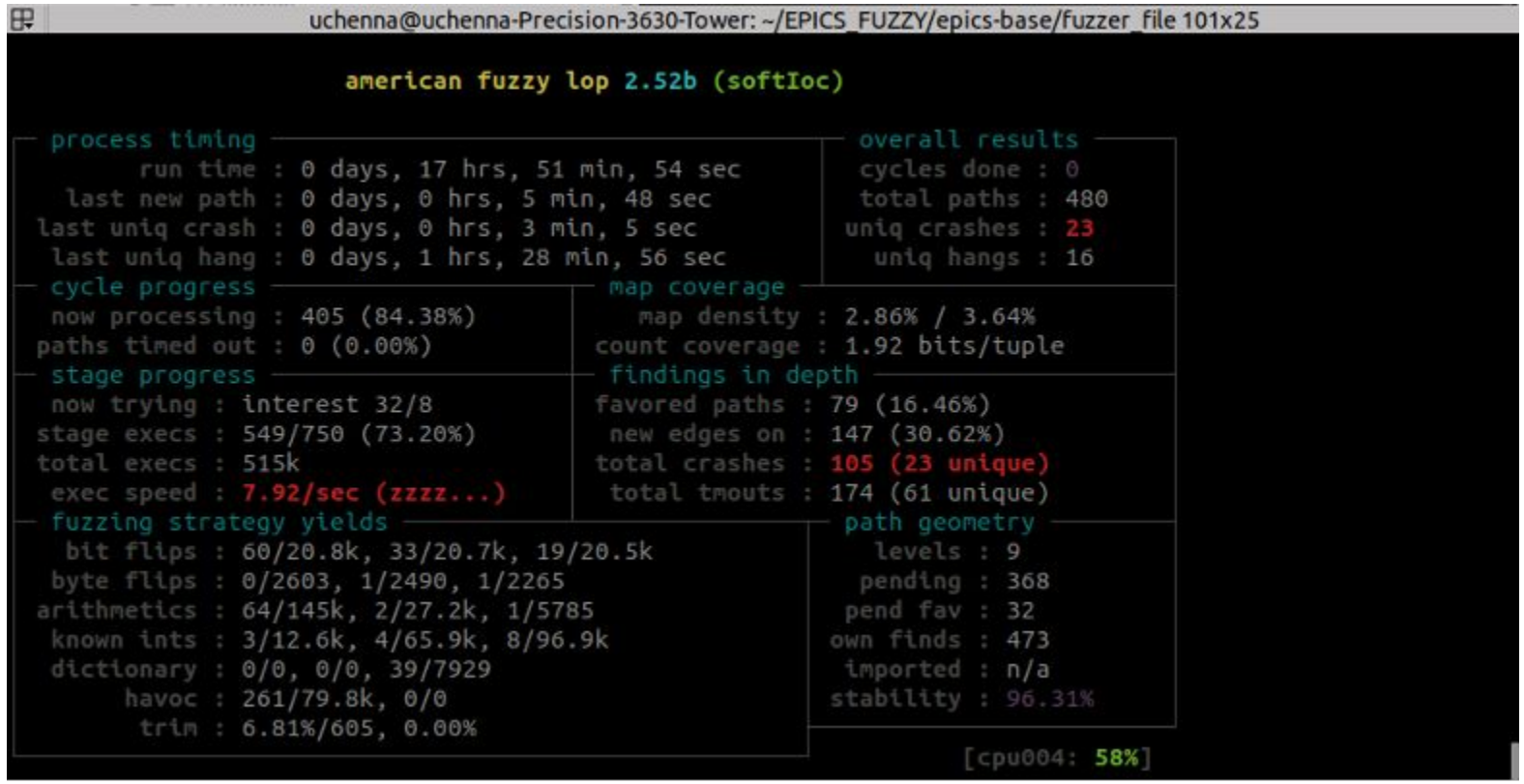


- Scanned Selected Software RTEMS and EPICS component
  - Coverity Scan, Codiga, Clang
- Suffers a lot from False positive
- Had Issues from integrating Coverity Scan into the CI development Lifecycle

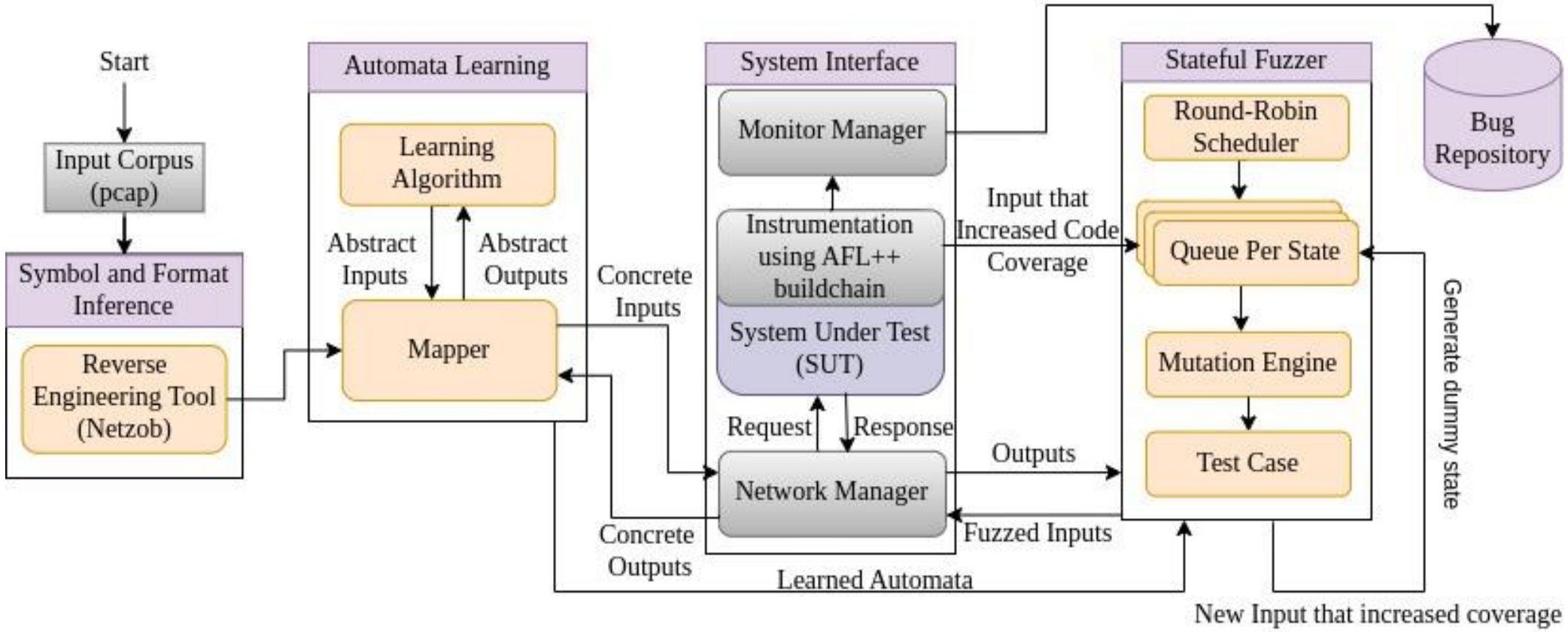
# Security Fuzz Testing to EPICS



- Fuzzing is an automated software testing that randomly feeds invalid and unexpected input into a program while monitoring for errors.
- Types of Fuzzing
  - Blackbox
  - Whitebox
  - Greybox
- Initial Effort
  - Applied AFL to EPICS 7 SoftIoC
- AFL not efficient to finding Bugs for Stateful Protocols.



- MICFICS: Model Inference Coverage-Guided Fuzzing for Industrial Control System Protocol Implementations
  - Benchmark Against stateful fuzzers like AFLNET, STATEAFL



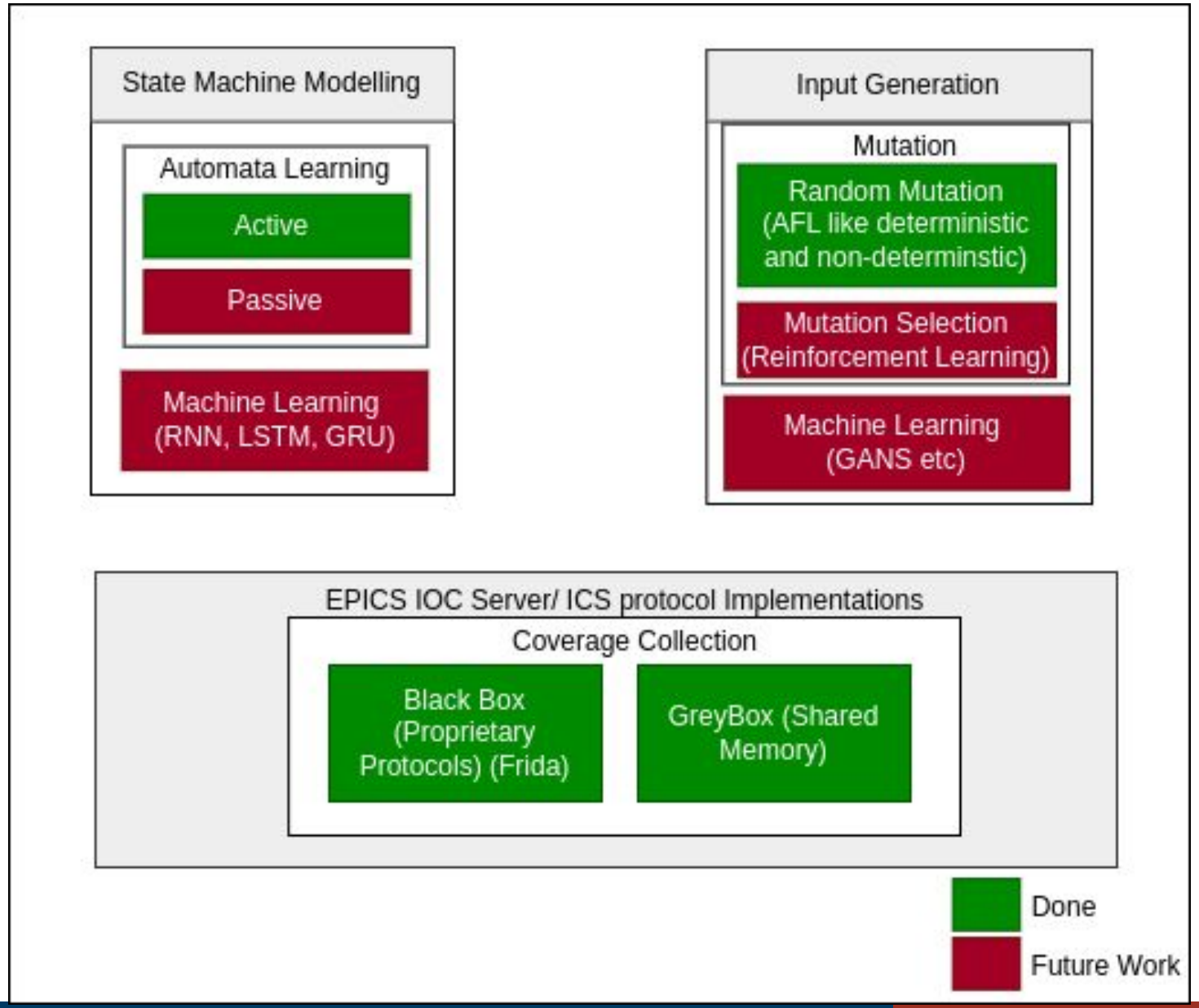
- Application of MICFICS to Portable Channel Access Server
  - Fuzz EPICS portable channel access server
  - Fuzz EPICS pvAcess Server (TCP) based example implementation

- Fuzz IOC initialization modules with and without Access Security Modules.
  - The module has multiple state in the “initHookState”

- Fuzz other EPICS components like the GUI etc

# Fuzzing Framework (In-progress)





- Application of Fuzzing Framework to fuzz multiple EPICS stack on pull request and git commits.

- Test multiple Coverage guided architecture in CI
  - Heterogeneous Fuzzers
  - Homogenous Fuzzers

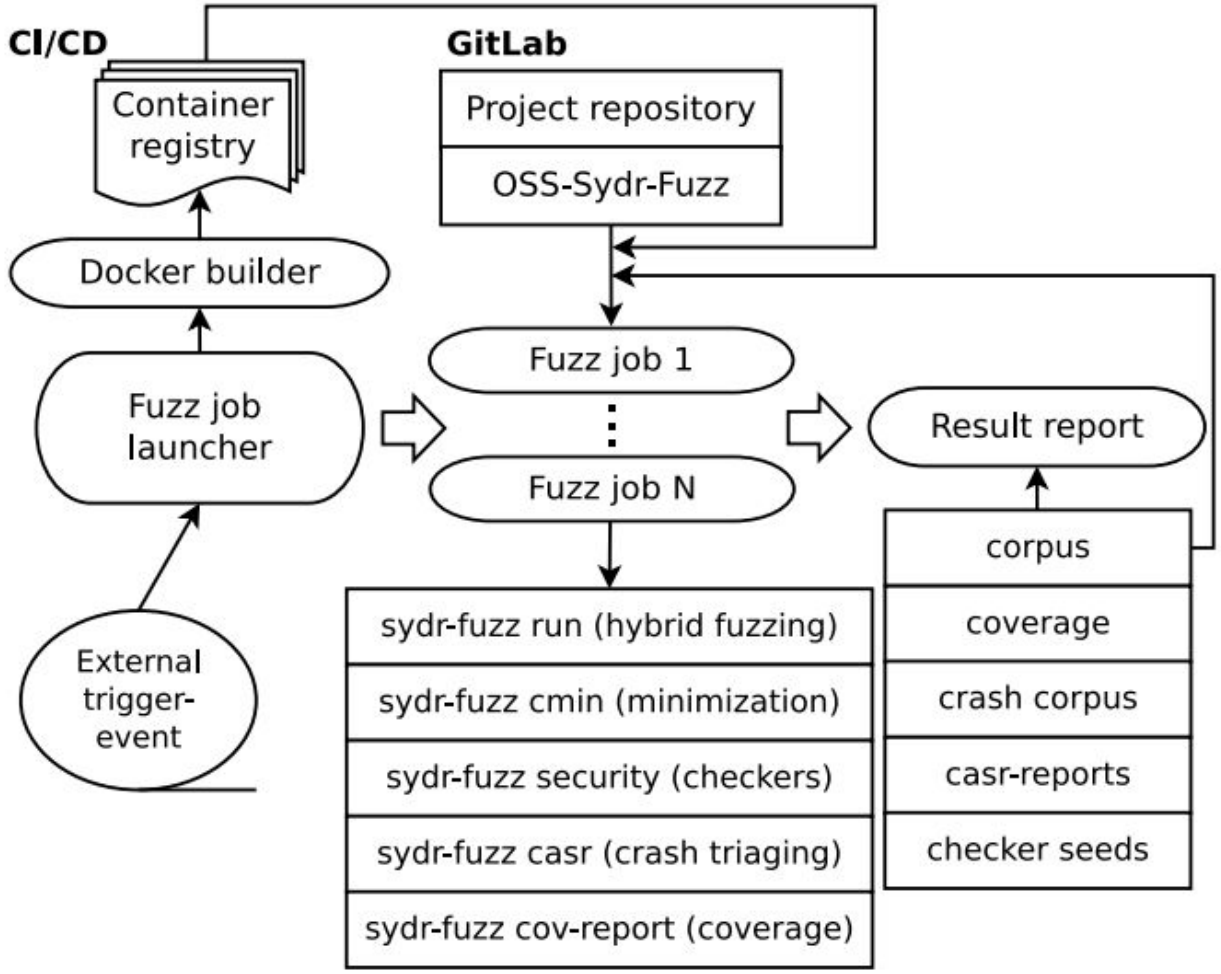



Fig. 1: OSS-Sydr-Fuzz CI Architecture.

# Testing EPICS-RTEMS integration



- Fuzz EPICS software running on RTEMS (RTOS) VME
  - Fuzzing on-device is non-practical due to low fuzzing speed
  - Fuzzing in an entire black-box manner results in missing feedback and limited crash detection
  - Fuzzing with hardware (device) in-the-loop leads to resource constraint because of synchronization of hardware and emulated environment.
- Automated Hardware rehosting
  - How to emulate Interrupts, DMA, and MMIO?
    - Qemu fully re-implements the behavior of all MMIO register
    - Use Fuzzing to approximate hardware peripheral and interrupts
- Develop a software only solution to fuzz test unmodified monolithic firmware (RTEMS) in a scalable way such as fuzzware.
  - Fuzzware uses DSE and fuzzer for access modeling of hardware generated values.
  - Fuzzware does not take into account of the statefulness of most peripheral modelling.

# Other ongoing and future projects



- RTEMS deployment repository being developed by Chris Johns (funded by Gemini)
- CI/CD testing infrastructure
- Upgrading to rtems-libbsd
- Adding more ports for RTEMS LWIP repository

# Conclusion



- We have Developed a modular Networking stack for RTEMS
- We use model inference and machine learning-based methodologies to improve state-of-the-art fuzzing for ICS/EPICS protocol implementations.
- We intend to build a CI/CD fuzzing infrastructure for EPICS.
- We propose to build the EPICS/RTEMS fuzzing framework by simulating the hardware generated values' MMIO, interrupts, and DMA.

# Published Works




- Banerjee, V., Hounsinou, S., Olufowobi, H., Hasan, M., & Bloom, G. (2022, November). Secure Reboots for Real-Time Cyber-Physical Systems. In Proceedings of the 4th Workshop on CPS & IoT Security and Privacy (pp. 27-33).
- Hounsinou, S., Banerjee, V., Peng, C., Hasan, M., & Bloom, G. (2021, December). Work-in-Progress: Enabling Secure Boot for Real-Time Restart-Based Cyber-Physical Systems. In 2021 IEEE Real-Time Systems Symposium (RTSS) (pp. 524-527). IEEE.
- Banerjee, V., Hounsinou, S., Gerber, H., & Bloom, G. (2021, October). Modular Network Stacks in the Real-Time Executive for Multiprocessor Systems. In 2021 Resilience Week (RWS) (pp. 1-7). IEEE.

# Acknowledgements



We would like to thank Dr. Gedre Bloom (ESSL@UCCS) and the RTEMS community for their support and feedback in these projects. Special thanks to Dr. Sena Hounsinou (MSU), Dr. Monowar Hasan (WSU), and other co-authors.

This work is supported by NSF CNS-2011620, NSF OAC-2001789, NSF-2046705, NSA H98230-21-1-0155 and Colorado State Bill 18-086. The opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of any other person or organization.


# Questions?