Integrated Instruction Set Randomization and Control Reconfiguration for Securing Cyber-Physical Systems

Bradley Potteiger, Zhenkai Zhang, Xenofon Koutsoukos

Vanderbilt University
Institute for Software Integrated Systems
1025 16th Avenue South, Nashville TN 37212

Motivation

- CPS are vulnerable to code injection attacks through software vulnerabilities
- Zero day exploits make it important to consider multiple levels of defense mechanisms
- System crashing can have devastating effects on safety critical CPS
- Performance overhead can lead to real time constraint violations

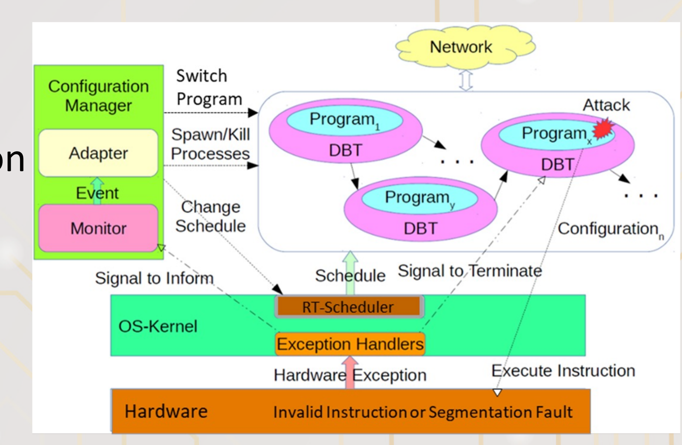
Problem Formulation

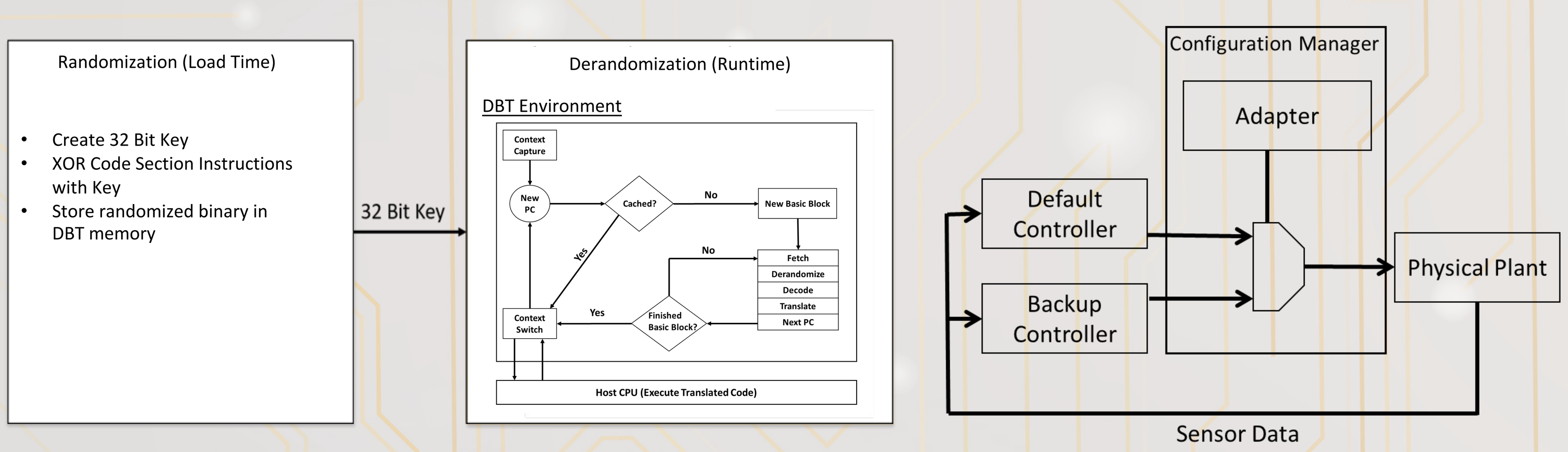
- Remote hijacking of vehicles can occur through code injection attacks
- Instruction set randomization (ISR) has been proven to be effective against code injection attacks, but leads to system crashing
- Reconfiguration is necessary to maintain safe CPS operation

Hypothesis: We can integrate ISR with control reconfiguration to detect and recover from attacks fast enough to maintain safe and stable CPS behavior

Approach

- Control Software ISR
  - Machine code encoded with 32 bit key
- Runtime Derandomization
  - Code is executed through dynamic binary translation layer (DBT)
  - Context switch between DBT and host CPU
  - Derandomization after instructions fetched in DBT pipeline
- Detection
  - Signal handler for invalid instruction exception
- Recovery
  - Switch to non-compromised controller





Experimental Setup

Hardware Testbed

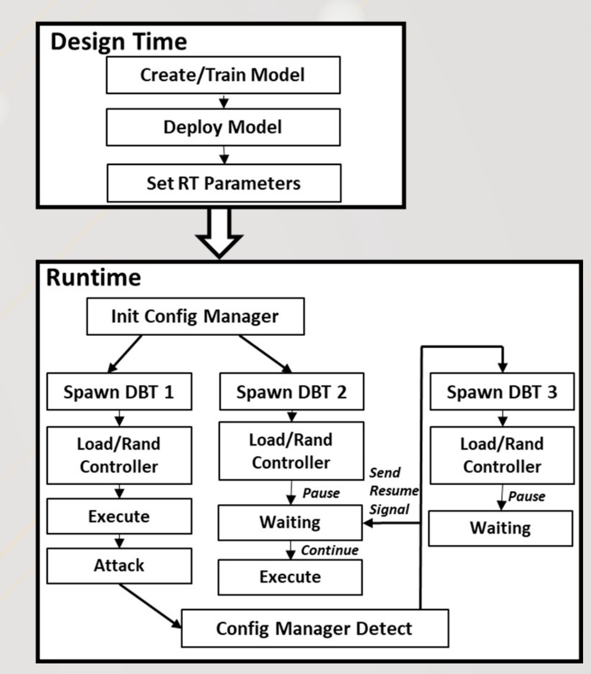
- ECU Cluster – 2 Beaglebone Black
- Controller Board – NVIDIA Jetson TK1
- Two Network Interfaces
  - Ethernet
  - CAN Bus

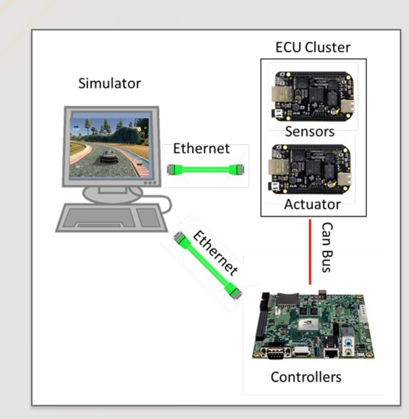
Software Environment

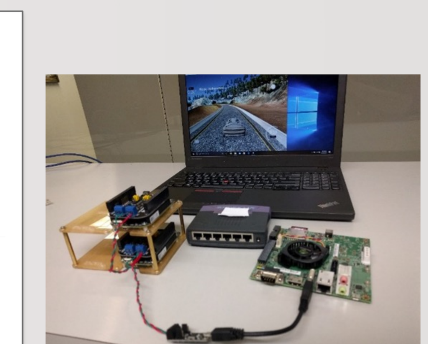
- OS – Linux4Tegra 21.5
- DBT Environment – MAMBO
- Communication
  - ZMQ – Ethernet
  - SocketCAN – Can Bus

Simulation Environment

- Udacity Simulator – Open source autonomous vehicle simulator (Physical Domain)
- Sensors and Actuators – Beaglebone Black
- Vehicle Controllers – NVIDIA Jetson
- MTD Framework – Encapsulates vulnerable controllers on NVIDIA Jetson





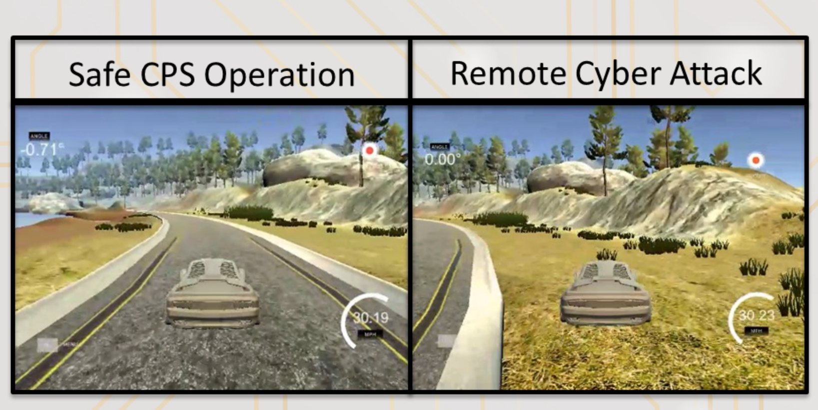


Autonomous Vehicle Case Study

Sensors
- Camera
- GPS
- Gyroscope

Actuators
- Steering
- Throttle

Controllers
- Convolutional Neural Network
  - 9 layers
  - Input: Camera Images
  - Output: Steering

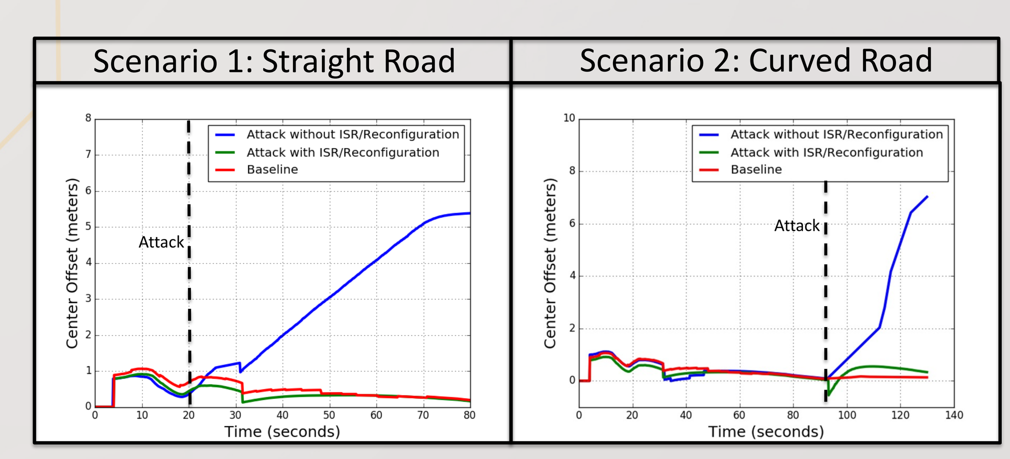


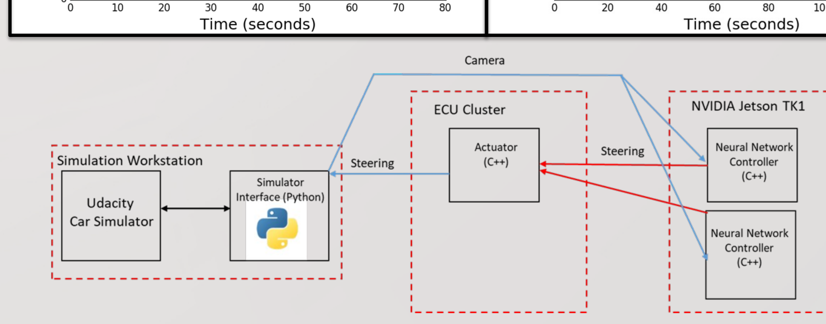
Case Study Results

Vulnerability
- Buffer overflow vulnerability in NN controller camera input processing function

Attack Process
- Spoof malicious camera packet to exploit buffer overflow in NN controller
- Divert control flow and execute malicious instruction payload
- Drive vehicle off of the road at maximum speed

Defense Mechanism
- ISR – Results in invalid instruction exception
- Recover to backup NN controller



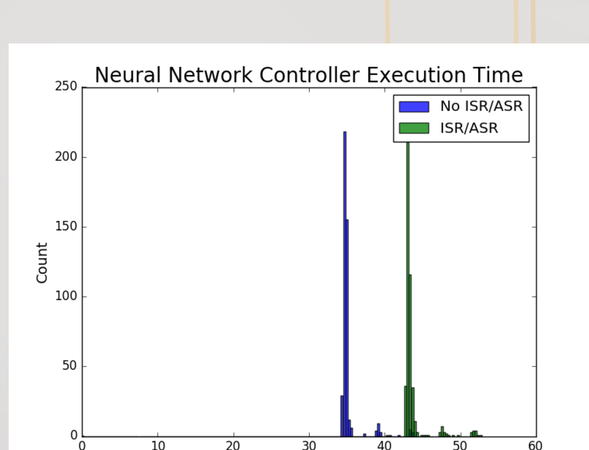


Performance Overhead

Experiment Setup
- 1000 iterations of controller under varying inputs

Measured Time Difference
- Sensor input received -> controller output computed

Neural Network Controller
- Sampling Period – 100 ms
- Worse Case Execution Times
  - No ISR Framework – 41.74 ms
  - ISR Framework – 54.32 ms
- Average Execution Times
  - No ISR Framework – 38.5 ms
  - ISR Framework – 42.9 ms

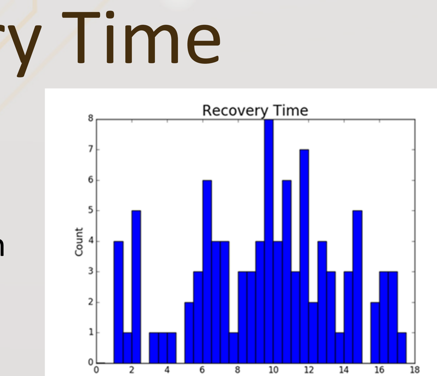


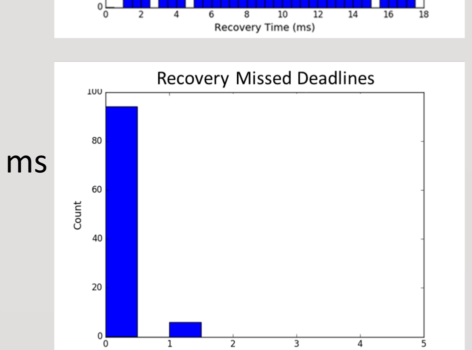
Recovery Time

Experiment Setup
- 100 experiments with ISR/Reconfiguration resulting in controller recovery

Measured Time Difference
- Attack detection -> backup controller resumes execution

Neural Network Controller
- Average Recovery Time – 10.21 ms





Future Work

- Implementing diverse controllers
- Exploration of address space and data space randomization
- Integration of dynamic reconfiguration

This work was supported by the AFRL SURE and NIST ARICS Programs. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of AFRL or NIST.

THE EIGHTEENTH ANNUAL
HIGH CONFIDENCE SOFTWARE AND SYSTEMS CONFERENCE
Annapolis, MD | May 7-9, 2018