Carnegie Mellon University Software Engineering Institute

Research Review 2021

Towards Incremental and Compositionally Verifiable Security for *CHIC-centric* Cyber Physical Systems

Amit Vasudevan, Ph.D.
MTS-Senior Researcher, FV-CPS/ACPS/SSD

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Document Markings

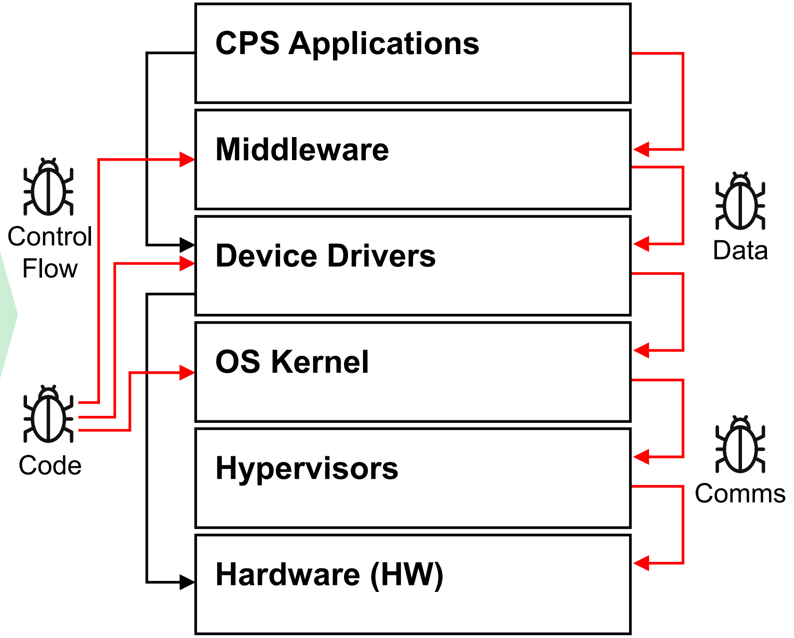
Carnegie Mellon University Software Engineering Institute

DoD Problem: Insecure CHIC-centric CPS Implementations

Carnegie Mellon University Software Engineering Institute

CHIC Stack Heterogeneity Challenge

- **Platforms**
- **Software Layers**
- **Configuration & Interactions**
- **Ownership**
- **Development Pedigree**



CPS = mission critical,
CHIC = criticality agnostic

Effective solution must meet these goals

INNOCUOUS + **PROVABLE** + **COST-EFFECTIVE**

State of the Art and Shortcomings

CHIC Stack Heterogeneity Challenge

CHIC Stack Implementation Security via Incremental, Composable, and Development Compatible Verification

Platforms	Software Layers	Configuration & Interactions	Ownership	Development Pedigree

Only Security, No Verification:
Micro-kernels, Separation kernels, MILS, isolation kernels, small-TCB hypervisors

- Isolate components via HW and/or SW
- Isolated components can still be exploited
- No privileged disaggregation

Goals

~~PROVABLE~~

COST-EFFECTIVE

INNOCUOUS

Security by Verifying Everything:
seL4, certiKOS, Ironclad, Verve, Verisoft, uberXMHF, IotVisor

- Focus on verification methodology (refinement proofs, mechanized semantics,…)
- Treat CHIC stack as monolith (e.g., run as a VM) towards isolation property
- Steep learning curve and cost for extensions and other properties
- Constrained functionality

Goals

PROVABLE

~~COST-EFFECTIVE~~

~~INNOCUOUS~~

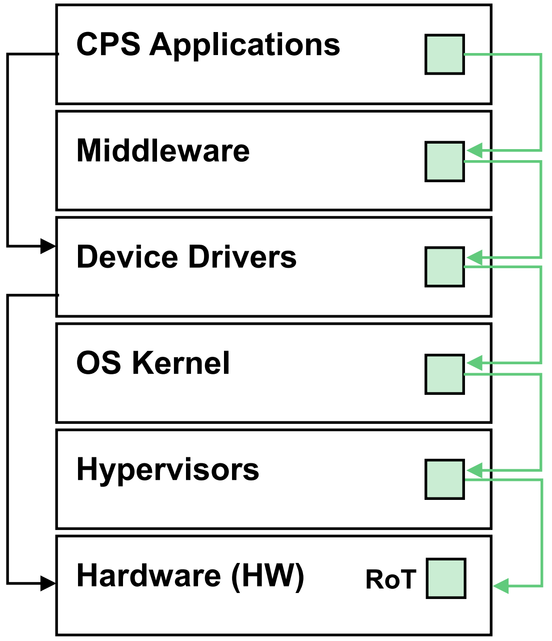
"Every time we try to do something real with solutions similar to seL4, we end up with lots of code hacks and fixes which breaks proofs when achieving isolation between critical software components. We would favor an architecture that is developer friendly and provides us with security properties for desired components in the software stack and platform of our choice. There is a need for modular, plug-and-play security solutions."

– Dr. Delbert Christman, VP R&D, Autonodyne
[USAF Skyborg + Golden Horde Awardee]

Our Solution: Incremental and Compositionally Verified Security of CHIC-centric CPS Stack Implementations

- üobjects: design time, singleton object abstraction for exclusive resource guards with secure interfaces
- üobject collection: runtime, protected group of üobjects
  - Root-of-Trust (RoT; hw, sw, hw-sw)
  - secure call routing
- AG reasoning theory on CHIC stack meshes unverified components and verified üobjects [Vasudevan et. al, USENIX Security 2016]

- Flexible implementation on platform and CHIC stack layer of choice
- Fine granularity retrofit
- CHIC-AG reasoning allows incremental, composable verification with free foundational properties + uobject specific properties
- Principled interfaces and resource closure allow state of the art verification techniques on multi-threaded uobject execution traces



INNOCUOUS + PROVABLE + COST-EFFECTIVE

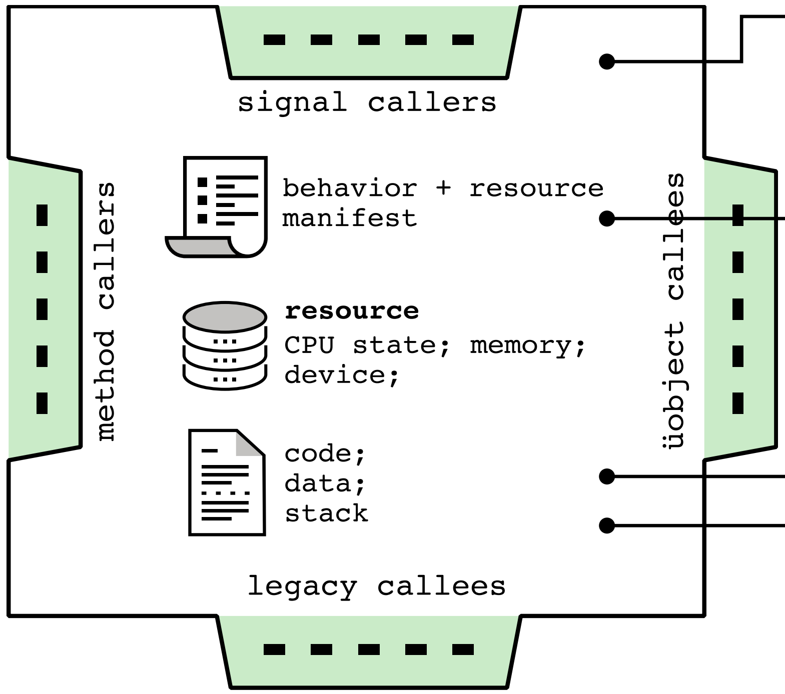
Our vision for this strategy has been published at ACM SIGOPS OSR Journal

Amit Vasudevan, Petros Maniatis, Ruben Martins. **überSpark: Practical, Provable, End-to-End Guarantees on Commodity Heterogenous Interconnected Computing Platforms.** In *ACM SIGOPS Operating Systems Review Journal – Special Issue on Formal Methods & Verification 2020*

Technical Approach – Building Blocks [Design Time]

Carnegie Mellon University Software Engineering Institute

üobject



Singleton object guarding exclusive indivisible system resource

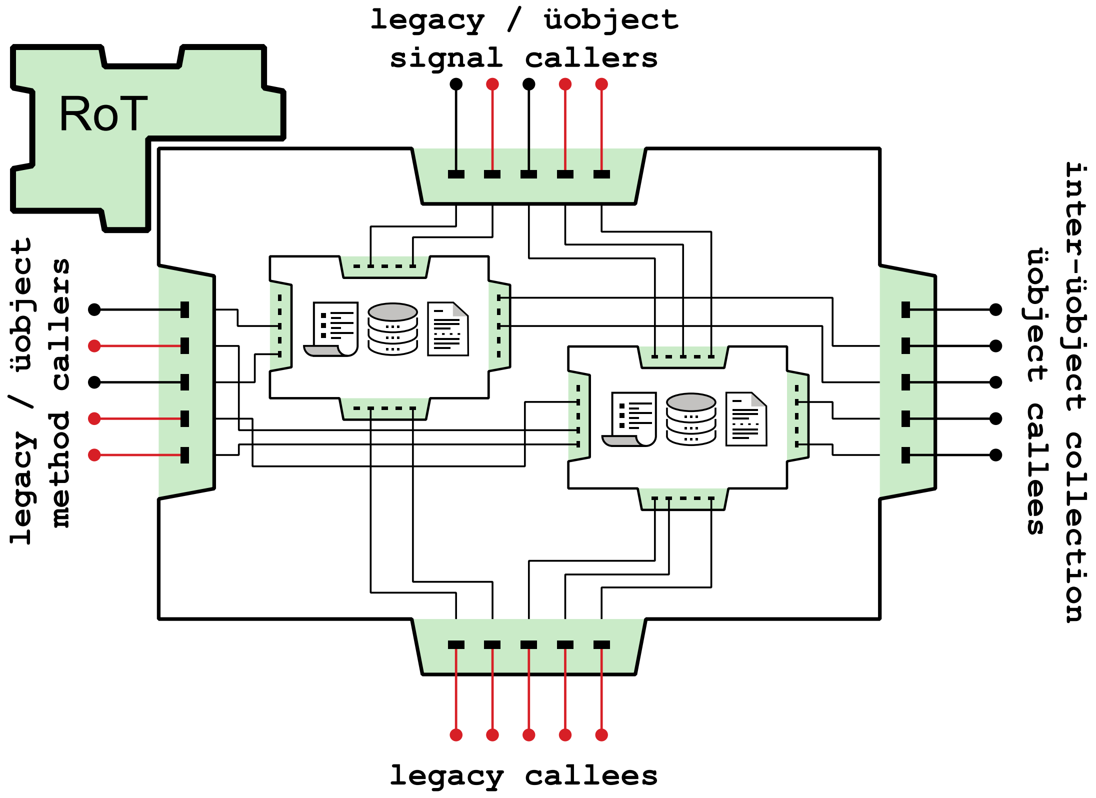
Principled entry, interruption, legacy code invocations and üobject invocations
- Facilitate AG reasoning and composition

Call-return Interfacing
- Handle CHIC programming idioms

Resource Interface Confinement
- Protection, access control, shared memory concurrency

Technical Approach – Building Blocks [Runtime]

üobject collection

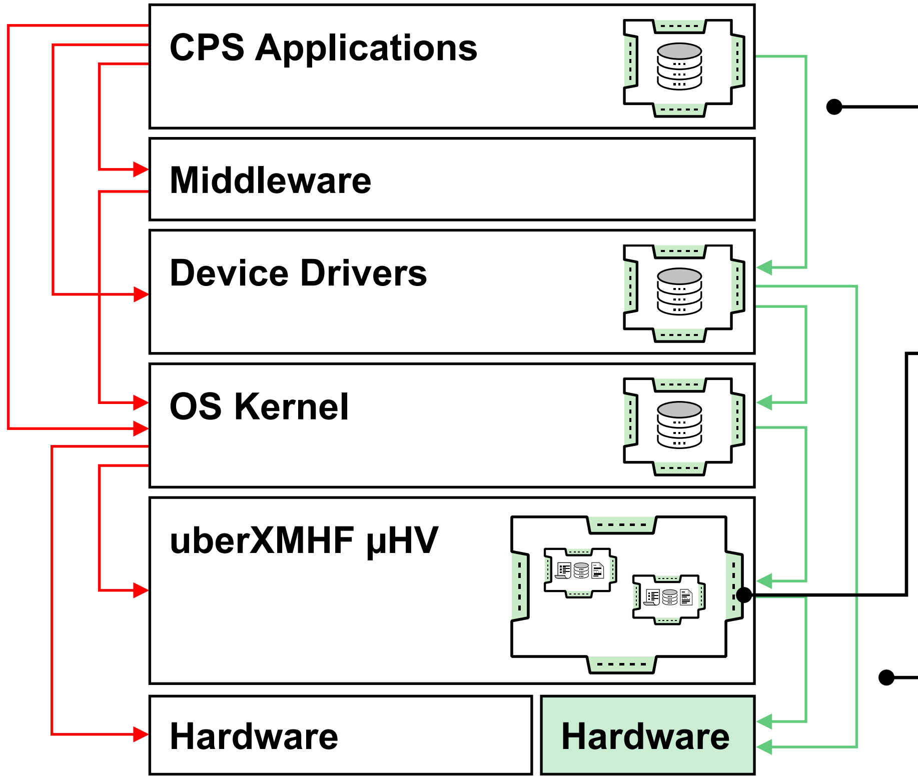

legacy / üobject signal callers
RoT
legacy / üobject method callers
inter-üobject collection üobject callees
legacy callees

Set of üobjects that share a memory address space

RoT (Root-of-Trust)
- Boot-strap and protect üobject executions

Sentinels
- Enforce call routings
- Caller/Callee mediation
- Logical privilege levels
- Flexible implementation

Technical Approach – On-Platform Secure Sensor Access



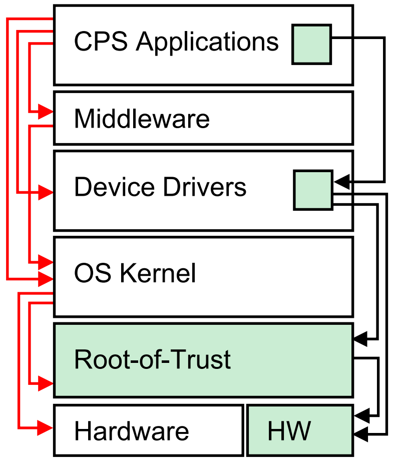
Sensor driver (e.g., GPS) and CPS application component as üobject collections

überXMHF micro-hypervisor (μHV) **[https://uberxmhf.org]** as hybrid root-of-trust

Secure control and data paths between CPS app., sensor driver μHV, and sensor hardware via sentinels (hypercalls)

Technical Approach: From Root-of-Trust to the Next Big Leap and Open Research Challenges

Scope: On-platform Secure Sensor Access



PROVABLE
+
COST-EFFECTIVE
+
INNOCUOUS [9]

(Verified) Micro-Hypervisor Root-of-Trust: überXMHF (https://uberxmhf.org)

2013
- x86 Automated Monolithic Verification with CBMC
- *Publication:* IEEE S&P

2016
- x86 Automated Compositional Verification with Frama-C and Compcert
- *Publication:* USENIX Security

2018
- ARMv8 version on low-cost commodity platforms (Raspberry Pi)
- *Publication:* IEEE Euro S&P **[Best Paper]**

2019
- ARMv8 hyper-scheduler extension for mixed-trust real-time computing
- *Publication:* IEEE RTCAS

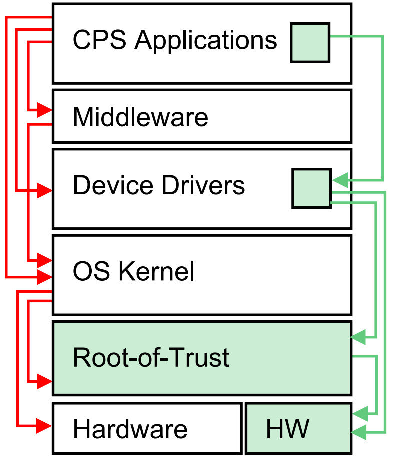
2020
- Trusted edge security gateway extensions for IoT security
- *Publication:* USENIX HotEdge

CHIC-stack Open Challenges beyond the micro-hypervisor layer:
- Multi-threading
- Hardware access (Within Scope)
- Legacy code access
- Programming idioms: deferred procedure calls, interrupts, call-backs
- Programming languages: C/C++/Assembly/Java (Within Scope)

- Challenges need to be addressed across four dimensions:
  - **S**ecurity, **V**erifiability, **P**erformance, **R**etrofit cost (**SVPR**)
- SVPR tradeoff evaluation is the foundational exploratory step towards next big leap!

Technical Approach: SVPR Tradeoff Evaluation

Collaboration with Autonodyne [Industry Partner]

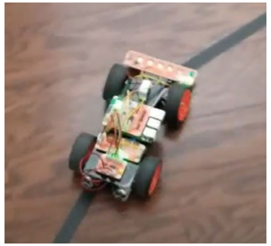


RoT: überXMHF verified micro-hypervisor (μHV); Hardware (HW) partitioning + üobject instantiation

üobjects: sensor driver, CPS application end-point

Secure calls: μHV [hypcall]

Off-the-shelf CHIC-centric Rover platform



- ARM Platform
- Linux OS
- Python CPS Application

Mission Functionality: Follow a pre-defined Way-point

Security Property: Secure On-platform Sensor Access

Research Question	Success Criteria
Security: Can we achieve security property?	Rover with üobjects completes mission in the presence of an attack while mission fails on base system (w/o üobjects)
Verifiability: Can we achieve composable, verifiable properties towards security?	Automatically discharge specifications directly on the code (memory integrity sub property) and compose with RoT
Performance: Can we achieve acceptable performance towards security?	Rover with üobjects completes mission (w/o attack) within time window comparable to base system (w/o objects).
Retrofit: Can we achieve acceptable retrofitting cost towards security?	Prototype tool-chain for developers to interact with üobjects similar to interfacing with existing OS APIs

Security Objective

Security Scope: On-Platform Secure Sensor Access
Integrity protection of the CPS app., sensor driver with authentic sensor data flow between them

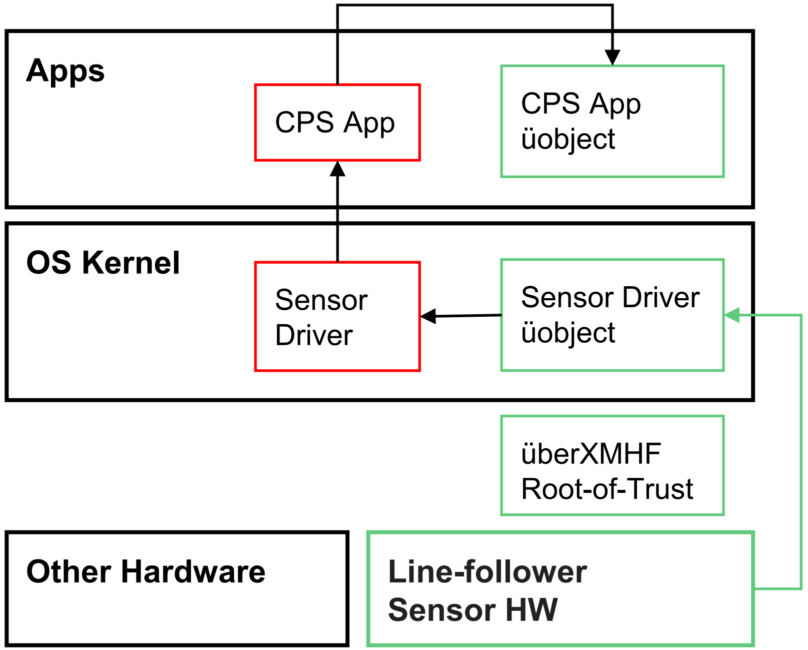
Research Question	Evaluation Metrics	Success Criteria
Can we achieve security property?	Simulated Memory Integrity Attack	Rover with üobjects completes mission while mission fails on base system (without üobjects)

Security Objective

Security Scope: On-Platform Secure Sensor Access

Integrity protection of the CPS app, sensor driver with authentic sensor data flow between them

- Designed and implemented secure sensor access mechanism using RoT-backed üobjects
- üobjects memory protection via RoT so they cannot be directly manipulated from any other system components
- HMAC used for sensor data integrity and authenticity between sensor driver uobject and CPS application üobject
- HMAC keys are boot-strapped into üobjects by RoT upon instantiation
- Our approach prevents sensor data integrity attacks and provides on-platform secure sensor access



Demo! What You See is What You Get!

Verifiability Objective

Security Scope: On-Platform Secure Sensor Access
Integrity protection of the CPS app., sensor driver with authentic sensor data flow between them

Research Question

Can we achieve composable verifiable properties towards security?

Evaluation Metrics

TLA+/TLAPS model level specifications and proofs.

ACSL code-level specifications using Frama-C

Success Criteria

Automatically discharge specifications directly on the code (memory safety sub-property) and compose with RoT

Verifiability Objective: Overview of Approach

CHIC-Centric CPS System Model and Invariant Definitions

Encode CHIC-Centric CPS System Model in TLA+ ➔

Mechanized Proofs of Invariant in Distributed Setting)

Hierarchical, Mechanized, Proofs of Concurrent Memory Safety Invariants via TLAPS ➔

Show invariants are inductive

Discharge Invariants on C and Assembly code (CPS app and drivers) separately, and using sequential verification tools (e.g., Frama-C)

Demo! What You See is What You Get!

Performance Objective

Carnegie Mellon University Software Engineering Institute

Security Scope: On-Platform Secure Sensor Access
Integrity protection of the CPS app, sensor driver with authentic sensor data flow between them

Research Question

Can we achieve acceptable performance towards security?

Evaluation Metrics

Benchmarks for CPS application and sensor I/O

Success Criteria

Rover with üobjects completes mission (without attack) within time window comparable to base system (without objects)

Anticipated 8-15% CPU/Memory/Overhead

Performance Objective

- Collected results over 10 laps of the rover on the line-following circuit

- No micro-hypervisor, no üobject no attacker
  - RMS error 1.577 with average time per lap 19.79 secs
- Micro-hypervisor, üobject no attacker
  - RMS error 1.619 with average time per lap 19.81 secs
- Micro-hypervisor, üobject with attacker
  - RMS error 1.611 with average time per lap 24.55 secs

- CPU utilization is ~3%

Retrofit Objective

Security Scope: On-Platform Secure Sensor Access

Integrity protection of the CPS app, sensor driver with authentic sensor data flow between them

Research Question

Can we achieve acceptable retrofitting cost towards security?

Evaluation Metrics

SLoC (person-yr. effort) and function-variable metrics differential on driver and CPS app

Success Criteria

Developers interact with üobjects similar to interfacing with existing OS APIs

Retrofit Objective

- **SLoC**

  25 lines of Python code to cope with smoother turns on wood floor

  ~200 lines of C code for CPS application and sensor driver

- 4 Person Weeks

  Developer who was new to the rover code-base

- Refactored sensor-driver code to adhere to üobject abstraction and perform HMAC functionality
  - C code ➔ C code with HMAC

- Refactored CPS application code to adhere to üobject abstraction and perform HMAC functionality
  - Python code ➔ C code with HMAC

- Interfacing to Root-of-Trust (RoT)
  - library to invoke üobjects with RoT-backed memory protections

Publications and Open Source

Carnegie Mellon University Software Engineering Institute

- Amit Vasudevan, Petros Maniatis, Ruben Martins. überSpark: *Practical, Provable, End-to-End Guarantees on Commodity Heterogenous Interconnected Computing Platforms*. ACM SIGOPS Operating Systems Review Journal 2020.
- *Towards Practical and Provable Guarantees on Commodity Heterogenous Interconnected Computing Platforms*. NSA Workshop on Hot Topics in Science of Security 2021.
- *Towards Practical Security on Commodity Cyber Physical Systems*. To be submitted to ACM Transactions on Cyber Physical Systems (TCPS)
- Open Source Artifacts
  - https://github.com/uberspark/uberxmhf
  - https://github.com/uberspark/uapp-SunFounder_PiCar-S
  - https://github.com/uberspark/uobjcoll-SunFounder_Line_Follower
  - https://github.com/uberspark/uobjcoll-raspberrypi-linux-i2c-bcm2835
  - https://github.com/uberspark/tests-and-evaluation

Summary: Research Successes and Future Work

Carnegie Mellon University Software Engineering Institute

Research Successes

- We were able to realize RoT-backed security with üobjects on an existing CPS ecosystem with minimal performance and developer retrofit to protect against memory-integrity violation “class” of attacks
- We were able to successfully model the CHIC-centric CPS system in TLA+ and prove concurrent memory safety properties in a composable manner
- Our technical progress is illustrated by our demo, open-source artifacts, and papers
- DoD stakeholders continue to be very interested in this technology, including DoD industry collaborators (e.g., Autonodyne).

Future Work

- Scaling in the presence of multiple sensors/actuators.
- Discovering and addressing control algorithm structure and/or complexity
- Investigate TLA+ proof engineering to maintain mechanized proofs in addition to the already development compatible code-level verification

Why this work is important

DoD CPS are becoming key to all the modernization priorities in DoD.

Integrating provable cyber protection into these systems will be critical to the success and much better than layering patches on later.

Team

Amit Vasudevan, Ph.D.
Principal Investigator
MTS–Senior Researcher
SEI/CMU

Anton Dimov Hristozov
MTS–Software Engineer
SEI/CMU

Bruce Krogh, Ph.D.
Professor Emeritus
SEI/CMU

Michael J McCall
Associate Software Security Engineer
SEI/CMU

Ruben Martins, Ph.D.
System Scientist
CMU/CSD

Delbert Christman, Ph.D.
VP R&D
Autonodyne

Raffaele Romagnoli, Ph.D.
Researcher
CMU/ECE

Jeff Boleng, Ph.D.
Technical Advisor
SEI

Contact Information

Presenter / Point of Contact match to Information Sheets

Name Dr. Amit Vasudevan

Title MTS – Senior Researcher SSD/ACPS/FVCPS

Email: info@sei.cmu.edu