Fall 2021 CAE Tech Talk
November 18, 2021

Securing Cyber-Physical Systems by Platform Reboot

Monowar Hasan

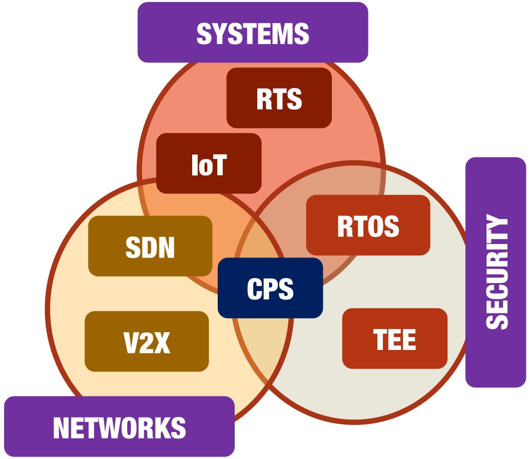
Assistant Professor, School of Computing
Wichita State University, Wichita, KS
monowar.hasan@wichita.edu

Wichita State University
College of Engineering
School of Computing

About Me

- Assistant Professor
  - School of Computing, Wichita State University (WSU)
  - Cyber-Physical Systems Security Research Lab (CPS2RL) [https://cps2rl.github.io]
    - Current members: 3 PhD, 2 Undergraduate
  - Past: UIUC (PhD, 2020), UM (MSc, 2015)

- Research: Systems, Security, Networking
  - Security for real-time, IoT, and cyber-physical systems
  - Resilient real-time networks using SDNs
  - Security and resource management for vehicular communication networks



Today's Talk

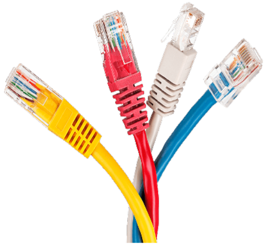
Security for Cyber-Physical Systems

Cyber-Physical Systems (CPS)

CYBER

```
    return map;
  }, {});

  for (const method of focusMethods) {
    wrapper.find('GitTabItem').instance()[method]();
    assert.isTrue(spies[method].called);
  }
});

describe('destroy and terminatePendingState should be overridden as no-ops', function() {
  it('does not destroy', async function() {
    mount(buildApp());
    const item = await atomEnv.workspace.open(GitTabItem.buildURI());
    const callback = sinon.spy();
    const sub = item.onDidDestroy(callback);

    assert.strictEqual(callback.callCount, 0);
    item.destroy();
    assert.strictEqual(callback.callCount, 0);

    sub.dispose();
  });

  it('does not terminate pending state', async function() {
    mount(buildApp());

    const item = await atomEnv.workspace.open(GitTabItem.buildURI());
    const callback = sinon.spy();
    const sub = item.onDidTerminatePendingState(callback);

    assert.strictEqual(callback.callCount, 0);
    item.terminatePendingState();
    assert.strictEqual(callback.callCount, 0);
```

Software, Control Algorithms, Code

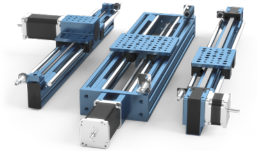


Networking, Communication

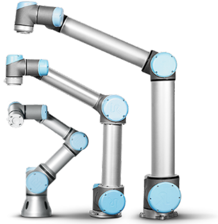
Microcontrollers, ECU, PLC

PHYSICAL

Sensors

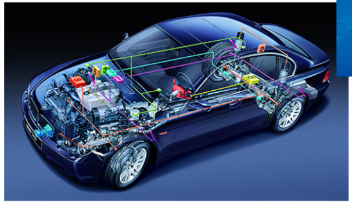


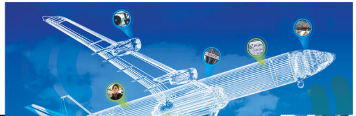
Actuators



Plant

Automobiles





Avionics

Control Systems

CPS Applications

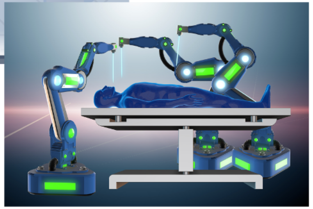
Unmanned Vehicles

Autonomous Driving

Surveillance

Manufacturing

Healthcare



* Image courtesy: Google Image Search

CPS Security

Traditional CPS

- Custom Hardware
- Proprietary Operating System
- Proprietary Software
- Limited Network Connection

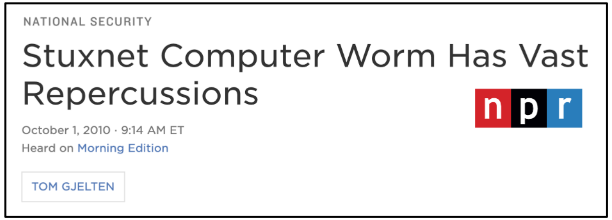
Modern CPS

- COTS Hardware
- Open Source Operating System
- Open Source Software
- More Connectivity → Internet!

Larger Attack Surface!

Modern CPS are vulnerable to security threats!

Increased Security Risks

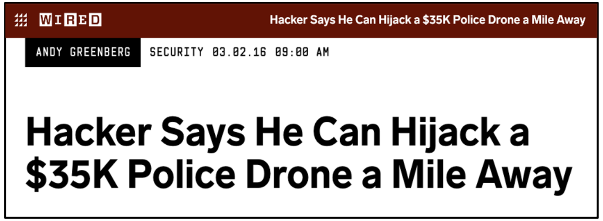

NATIONAL SECURITY

Stuxnet Computer Worm Has Vast Repercussions

npr

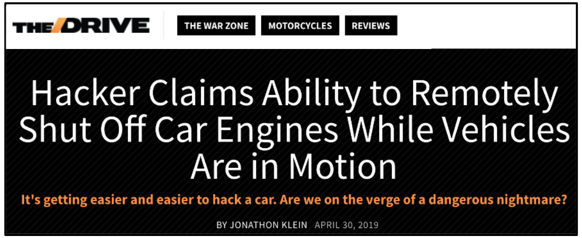
October 1, 2010 · 9:14 AM ET
Heard on Morning Edition

TOM GJELTEN


WIRED Hacker Says He Can Hijack a $35K Police Drone a Mile Away

ANDY GREENBERG SECURITY 03.02.16 09:00 AM

Hacker Says He Can Hijack a $35K Police Drone a Mile Away


THE DRIVE THE WAR ZONE MOTORCYCLES REVIEWS

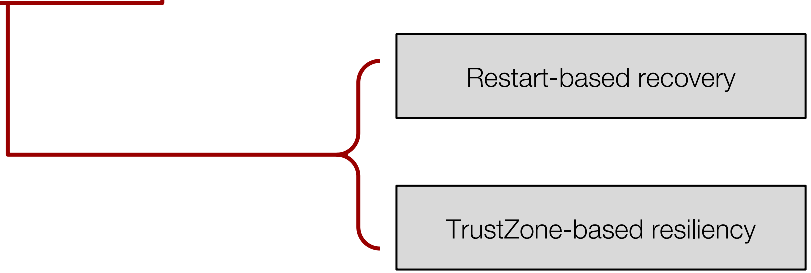
Hacker Claims Ability to Remotely Shut Off Car Engines While Vehicles Are in Motion

It's getting easier and easier to hack a car. Are we on the verge of a dangerous nightmare?

BY JONATHON KLEIN APRIL 30, 2019

Attack Resilient CPS Platforms

- Security issues → leads to safety issues
  - Difficult to ensure system won't be compromised
- Goal:
  - Provide guaranteed safety → under attack
- Proposed idea:
  - Proactive mechanism → prevents attack from progressing



Restart-based recovery

TrustZone-based resiliency

The Rest of Today’s Talk

ReSecure [IoT’18, ICCPS’18]

Preserving Physical Safety under Cyber Attacks

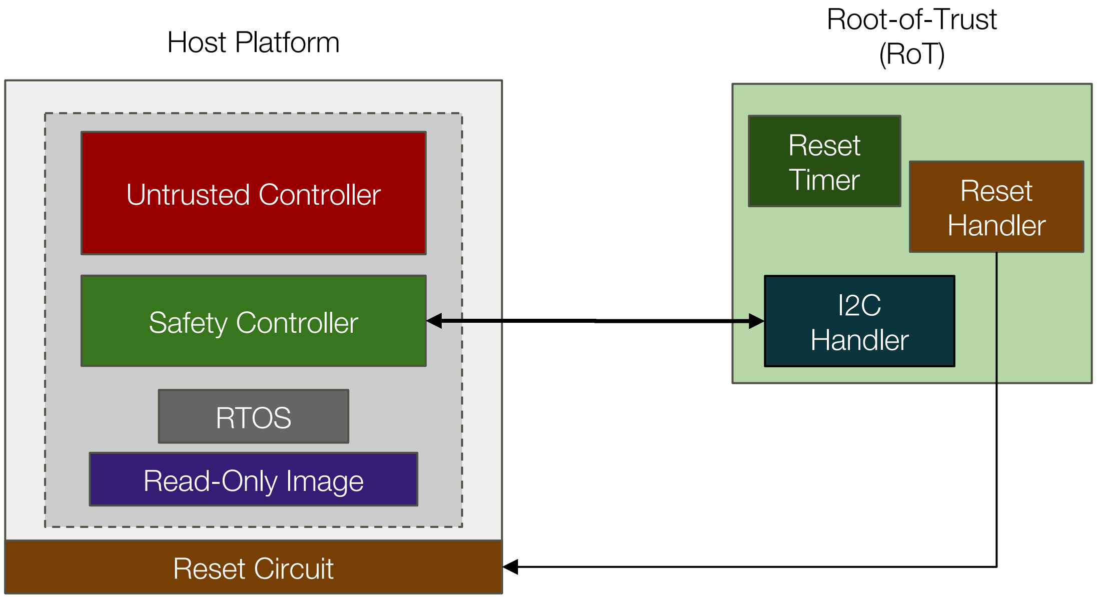
[IoT’18]	F. Abdi, C. Chen, M. Hasan, S. Liu, S. Mohan and M. Caccamo, "Preserving Physical Safety Under Cyber Attacks," iEEE Internet of Things Journal, Aug. 2019.
[ICCPS’18]	F. Abdi, C. Chen, M. Hasan, S. Liu, S. Mohan and M. Caccamo, "Guaranteed Physical Security with Restart-Based Design for Cyber-Physical Systems," ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS), 2018.

Our Approach: ReSecure [ICCPS'18]

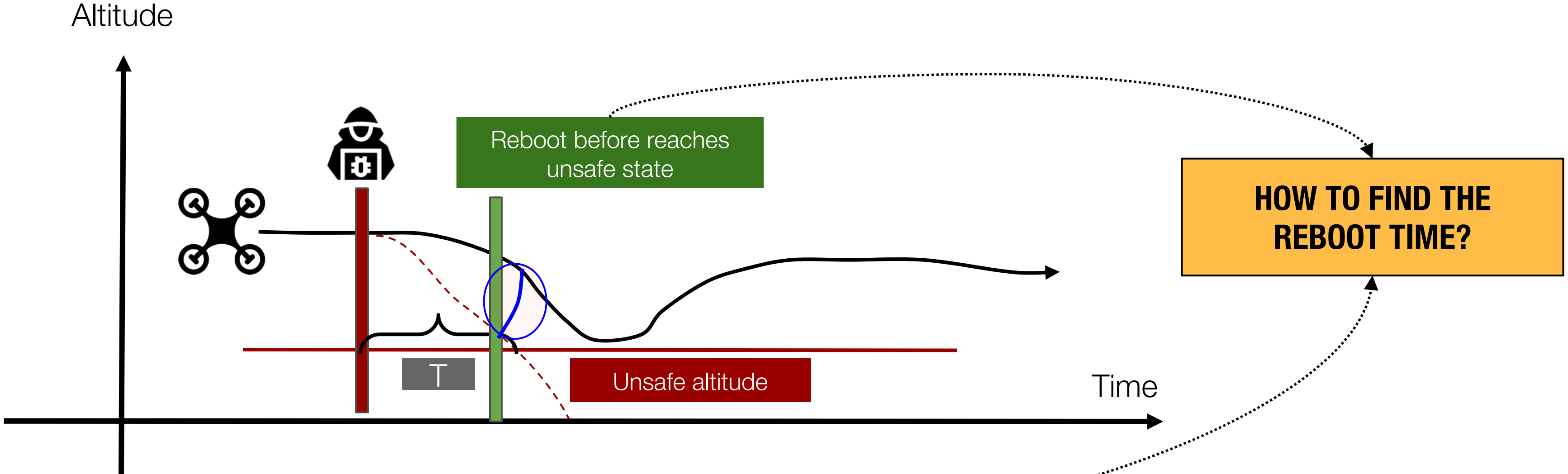
- Restart the system once a while to reset any attack progress
- Employ a Safety Controller (SC) and a Root-of-Trust (RoT) module

ReSecure: Design

- Host platform
  - Untrusted controller
  - Safety controller
- Root-of-Trust
  - Enforces restart



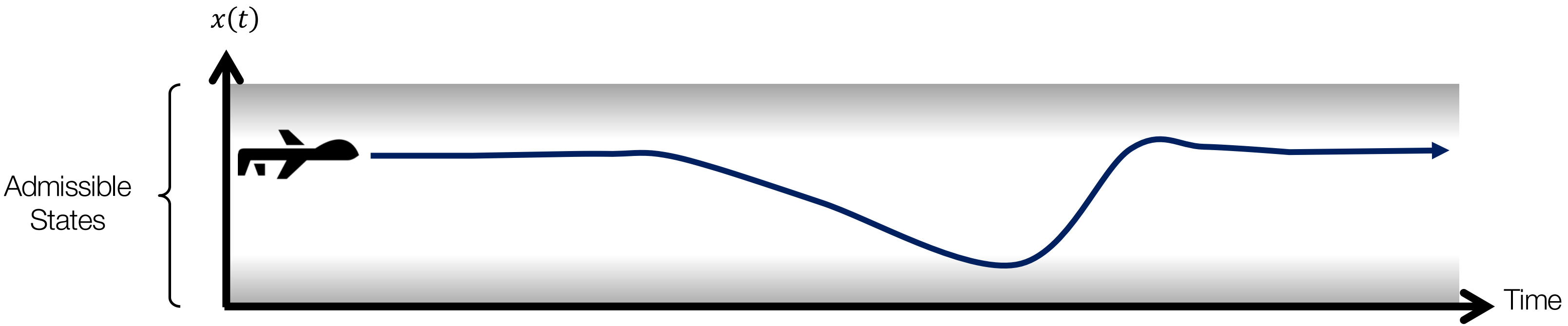
ReSecure: Overview


Altitude
Reboot before reaches unsafe state
T
Unsafe altitude
Time
HOW TO FIND THE REBOOT TIME?

- For a given state:
  - Calculate the shortest time T → system goes to “unsafe” state
- Reboot and reload the system before T

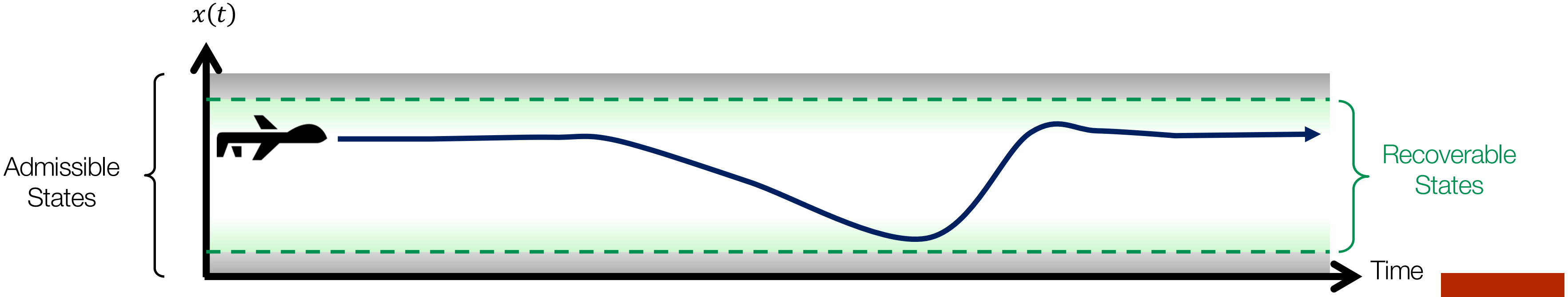
CPS States

- Admissible States S
  - States that do not violate any of the operational constraints of the physical plant
  - Safety invariant: system must always remain inside admissible states: $\forall t: x(t) \in S$



CPS States

- Admissible States S
  - States that do not violate any of the operational constraints of the physical plant
  - Safety invariant: system must always remain inside admissible states: $\forall t: x(t) \in S$
- Recoverable States R
  - Defined with regards to a given safety controller (SC)
  - A subset of admissible states ($R \subseteq S$) such that
    - if the given SC starts controlling system from $\mathrm{x} \in \mathrm{R}$, all future states will remain admissible



Determine Recoverable States

Reachability Analysis

- True Recoverable States:
  - All the states from which safety controller can stabilize the plant within α time.

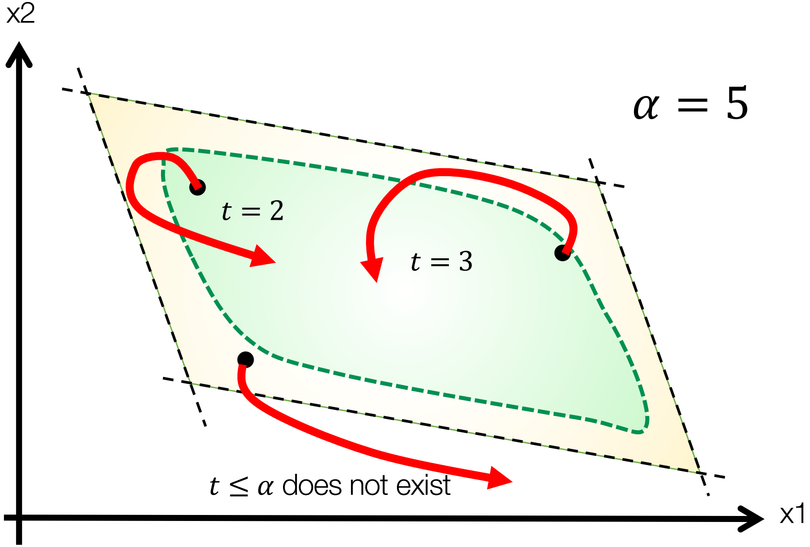
$\Gamma_\alpha = \{\, x \mid$

$$Reach_{\leq\alpha}(x, SC) \subseteq S\ \&$$

During recovering, the system should remain in admissible states.

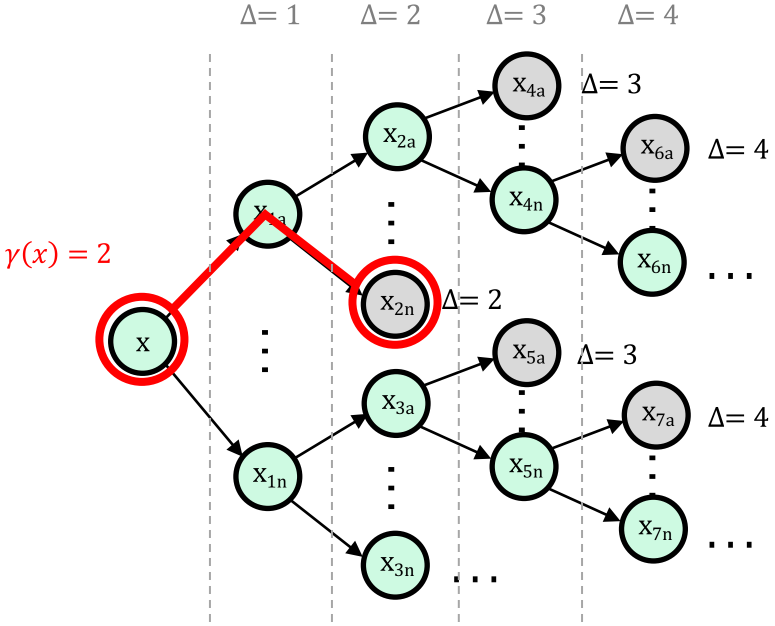
$$Reach_{=\alpha}(x, SC) \subseteq R\}$$

The destination should be a recoverable state.



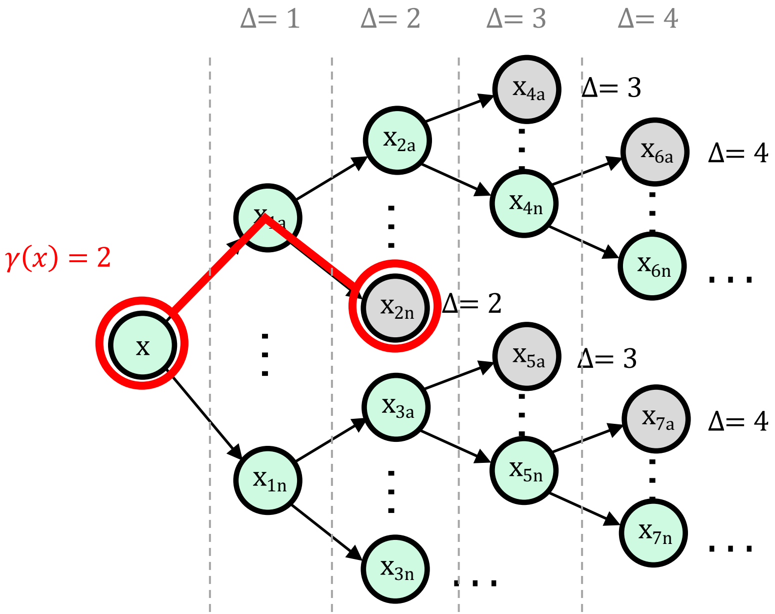
Determine Next Restart Time

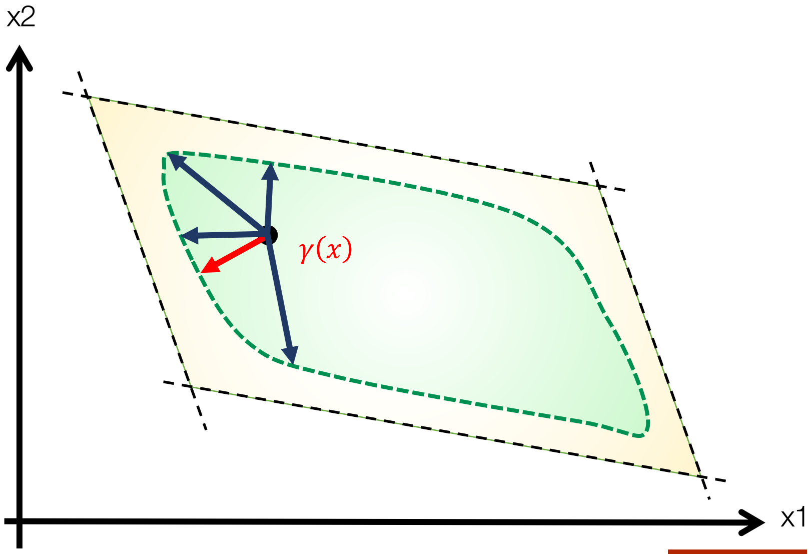
- From a given state:
  - Calculate the shortest time, $\gamma(x)$, to an unsafe state



Determine Next Restart Time

- From a given state:
  - Calculate the shortest time, $\gamma(x)$, to an unsafe state

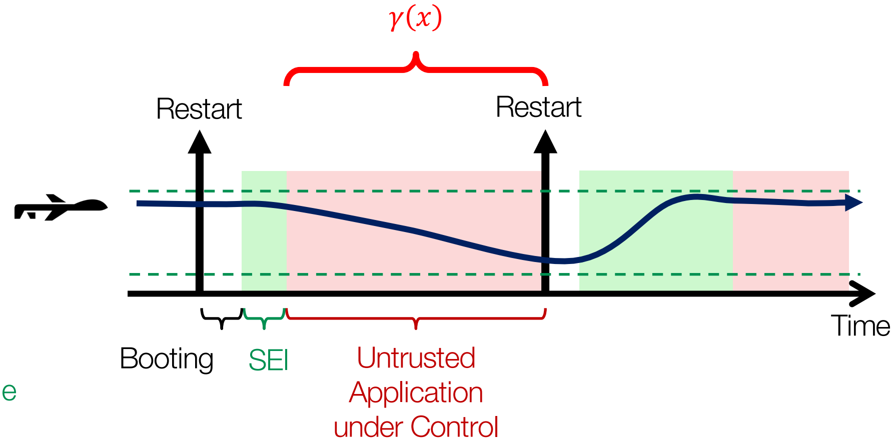




ReSecure: Workflow

- The system enters a Secure Execution Interval (SEI) during booting
  - The software is uncompromised
  - Access to RoT is enabled during SEI only

- Execution steps:
  1. Boot up (software is loaded)
  2. Enter SEI
  3. Run safety controller
  4. Check the system's state
  5. Compute next SEI time $\gamma(x)$
  6. Configure the restart timer on the RoT module (then RoT module closes I^{2}C)
  7. Exit SEI, jump to user's application (the untrusted controller)



Restart-based Recovery

Remarks

- Restarts are costly!
    - Platform specific
        - large restart time → not suitable for highly dynamic systems

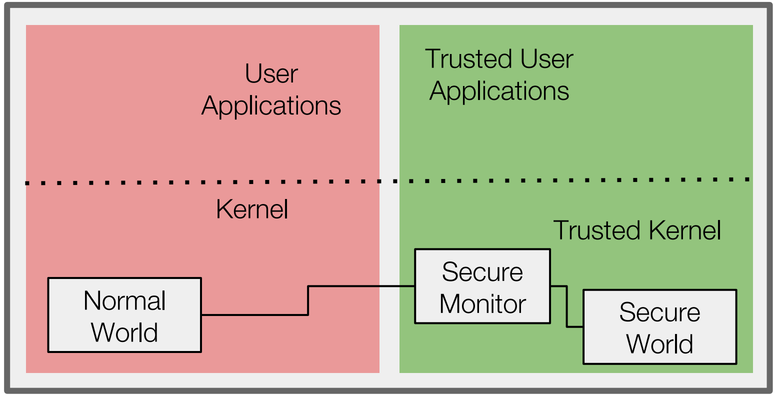
- Require custom hardware
    - Root-of-Trust

Follow-up work [IoT'18]

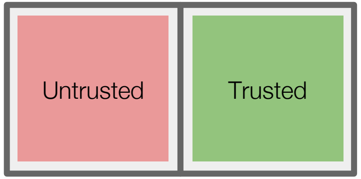
TrustZone-assisted recovery

arm TRUSTZONE

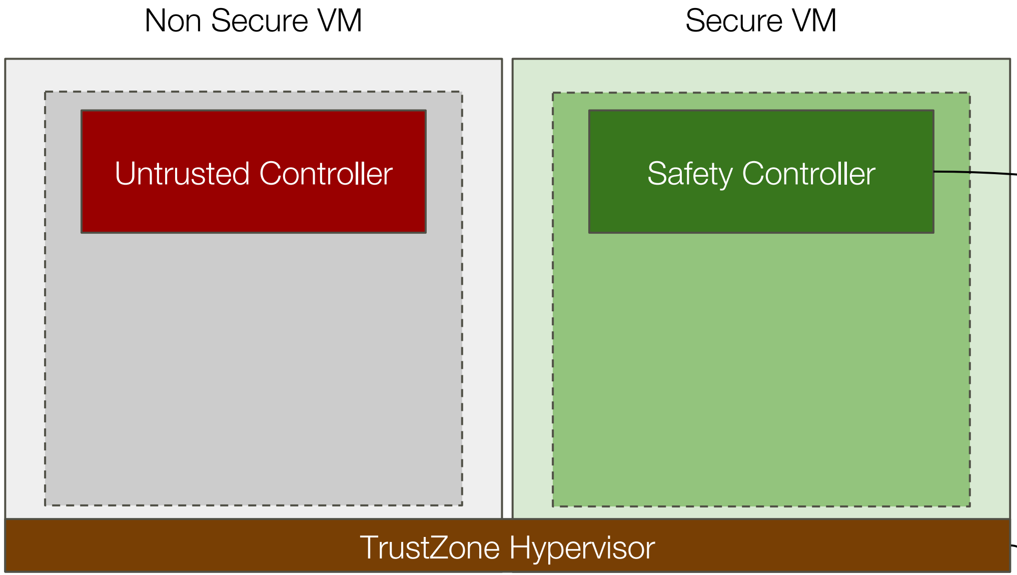
Background - ARM TrustZone



arm TRUSTZONE → isolates trusted software and data



TrustZone-based Recovery



Non Secure VM

Secure VM

Untrusted Controller

Safety Controller

TrustZone Hypervisor

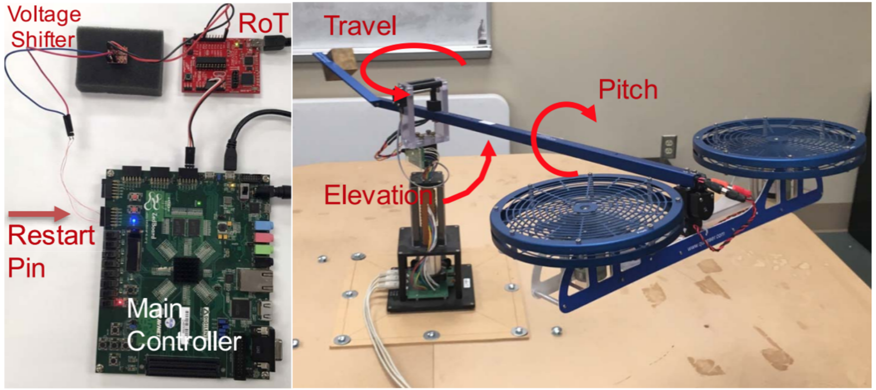
Prevents the attack from progressing!

- Recover the system to a *safe* state

- For a given state:
  - Calculate the shortest time T → system goes to “unsafe” state
- Transfer the control to the safety controller before T

Implementation & Case-Study

- Testbed:
  - 3 DoF Helicopter
- Host Platform:
  - Zedboard (Xilinx’s Zynq-7000)
  - FreeRTOS
  - ARM TrustZone (LTZVisor hypervisor)
- Root-of-Trust:
  - MSP430G2452 micro-controller
  - 160-bit internal timer

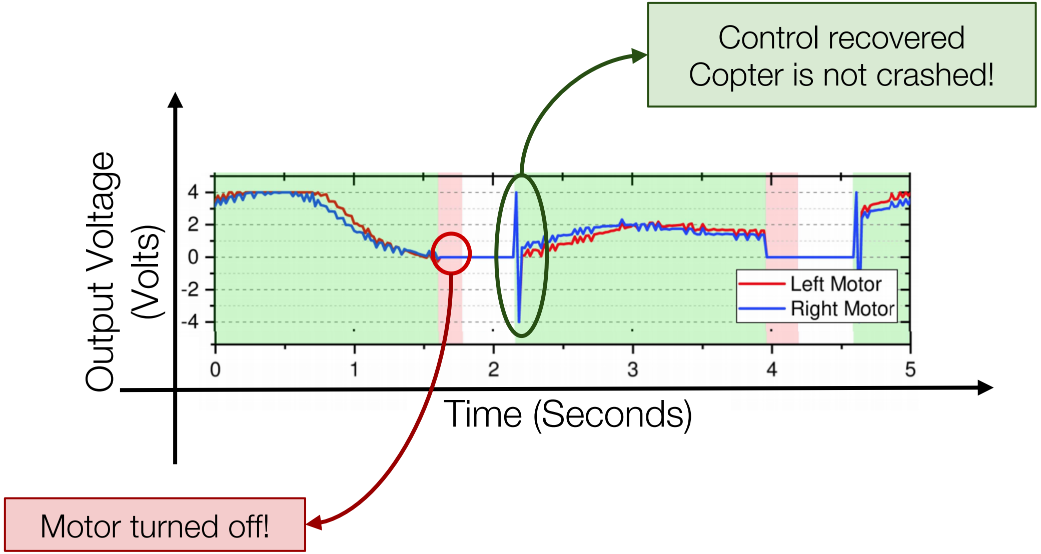


Safety Goal:
not to hit the surface of table

Results

- DoS Attack → turn off motors
  - Extreme case

- Green → Safety controller
- Red → Untrusted controller
- White → Reboot

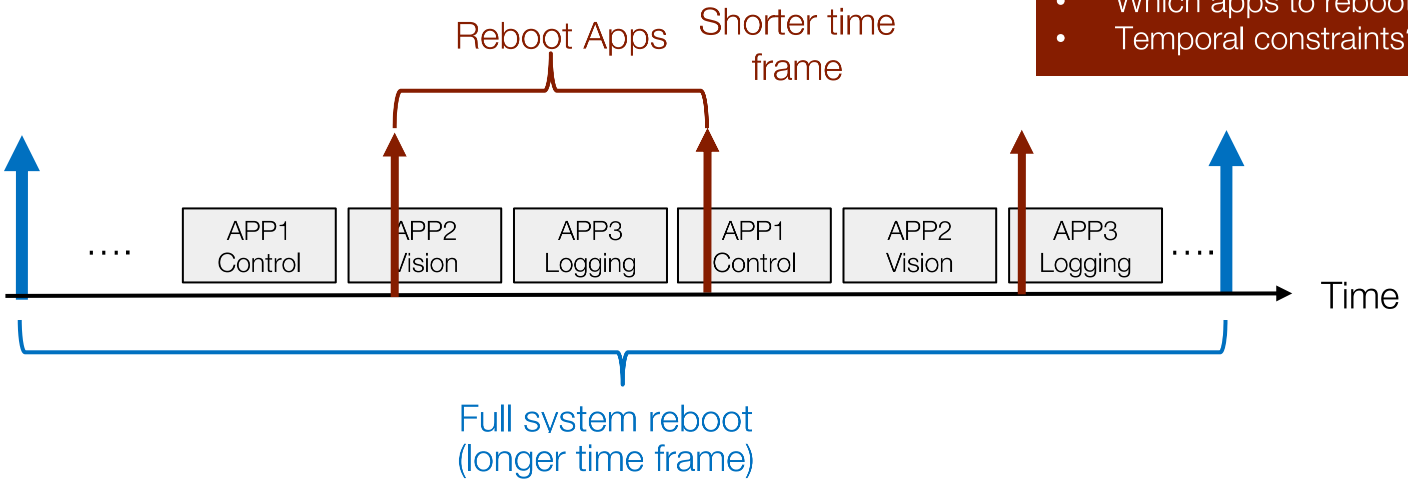


Ongoing Work

- <u>Proactive</u> → Application-level reboot

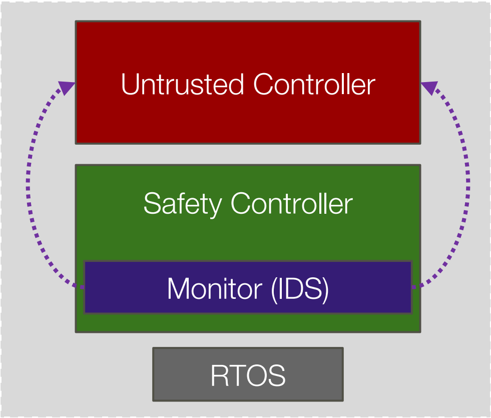
Challenges:

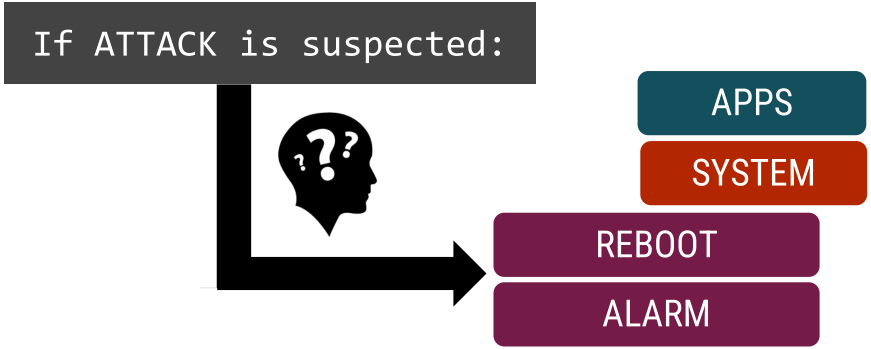
- Reboot frequency?
- Which apps to reboot?
- Temporal constraints?



Ongoing Work

- ○ <u>Proactive & Reactive</u> → Application & System-level reboot





Remarks

- Threats to critical systems are increasing
  - Requires layered defense mechanisms
- ReSecure: one way to secure critical CPS → active restart mechanism
  - Ensures physical safety
  - Prevents the attacks from progressing

Questions?

https://monowarhasan.info/
monowar.hasan@wichita.edu