SAIN: Improving ICS Attack Detection Sensitivity via State-Aware Invariants

Syed Ghazanfar Abbas, Muslum Ozgur Ozmen,
Abdulellah Alsaheel, Arslan Khan, Z. Berkay Celik, Dongyan Xu

USENIX Security 2024

PURDUE UNIVERSITY
CERIAS
PurSecLab

Programmable Logic Controllers (PLCs)

- Real-time rugged computer
- Embedded with a PLC program
- Control and monitor a physical process
- Commonly used in critical infrastructures

Programmable Logic Controllers (PLCs)

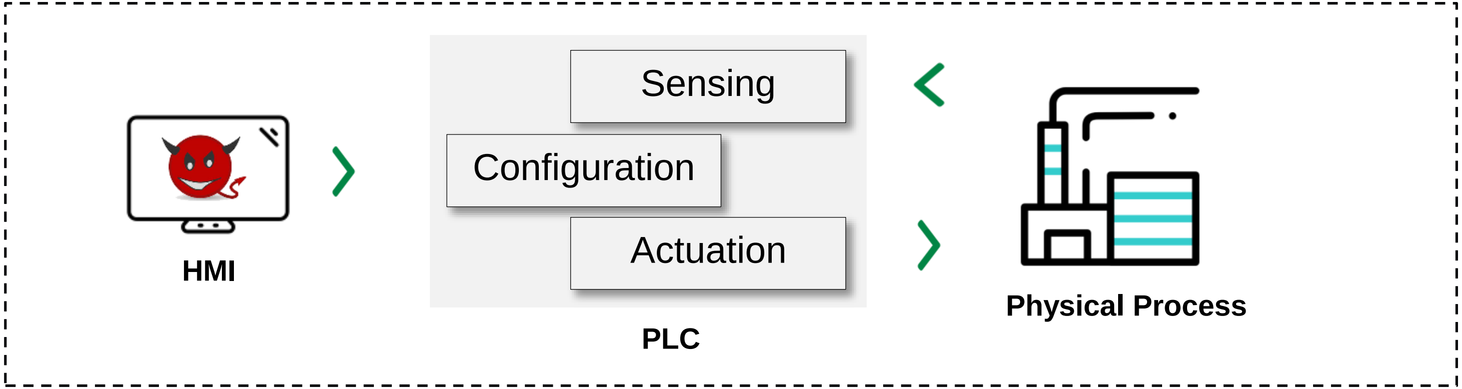
- The PLC program involves three types of variables

Sensing Variables
(Real-time sensing data)

Configuration Variables
(Process settings)

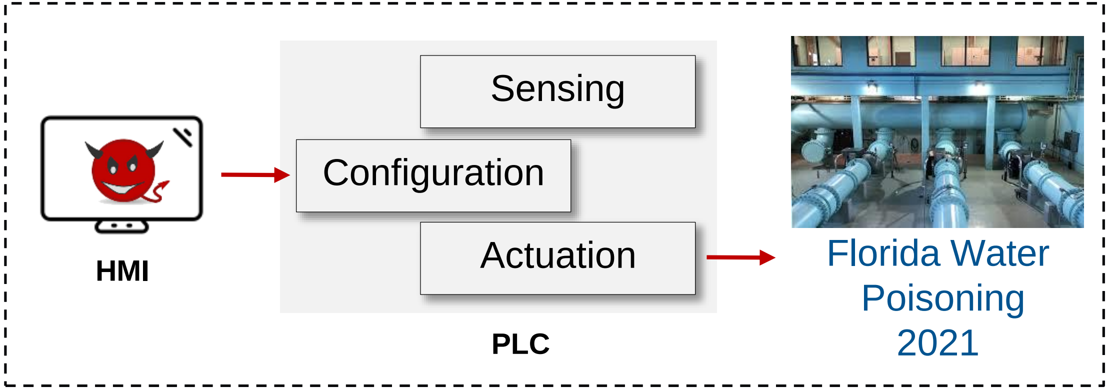
Actuation Variables
(Control signals for actuators)

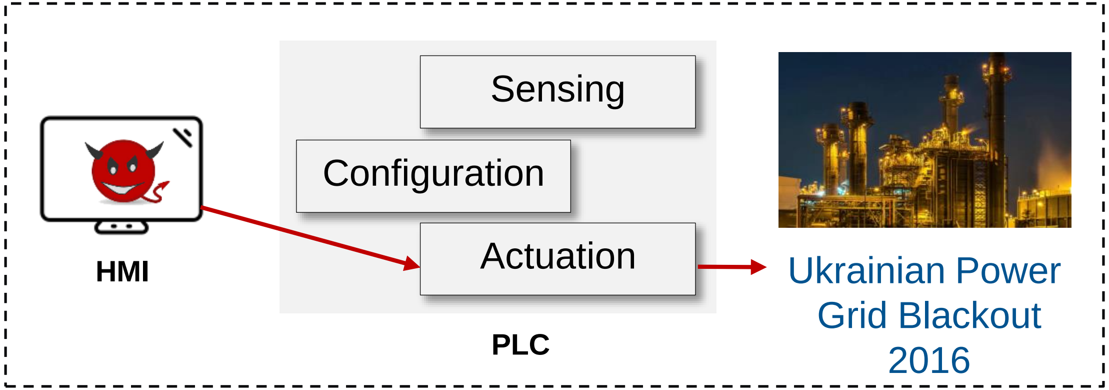
Our ICS Attack Model



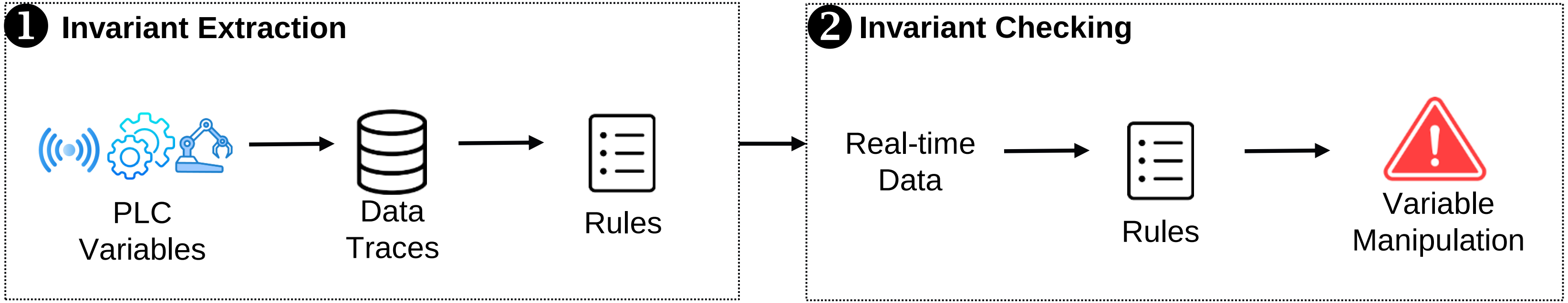
ICS Attacks

- Real-world attacks remotely manipulate sensing/configuration/ actuation PLC variables.





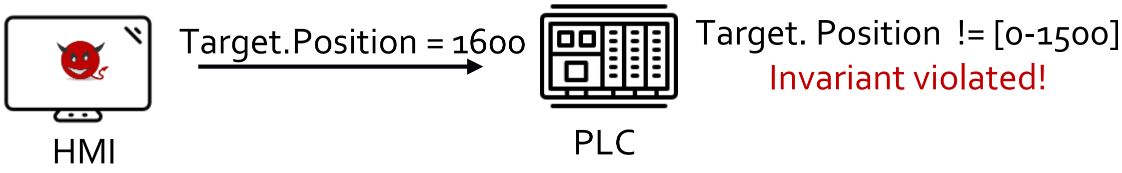
SOTA Defense: Invariant Checking



- Example:

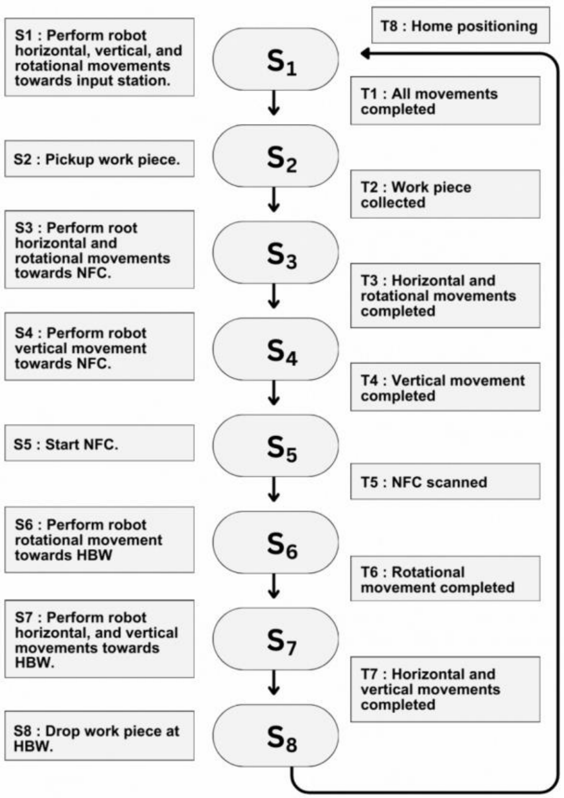
Offline: $Robot\text{-}movement\text{-}on \leftrightarrow Target.Position\ [0 - 1500]$

Online:



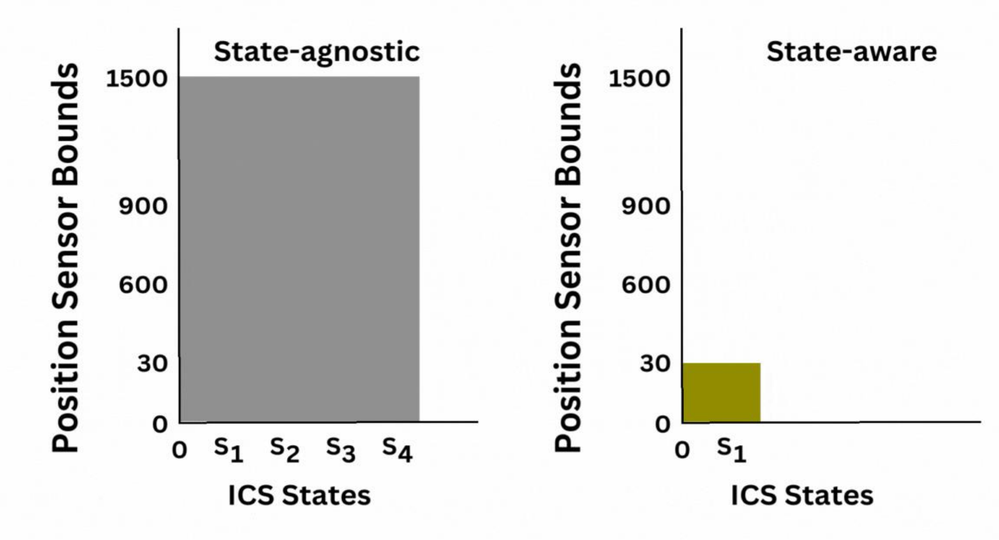
Problem w/ SOTA: State-Agnostic Invariants

- Loose bounds invariants leading to detection evasion
- ICS process is naturally reflected by states and transitions between them



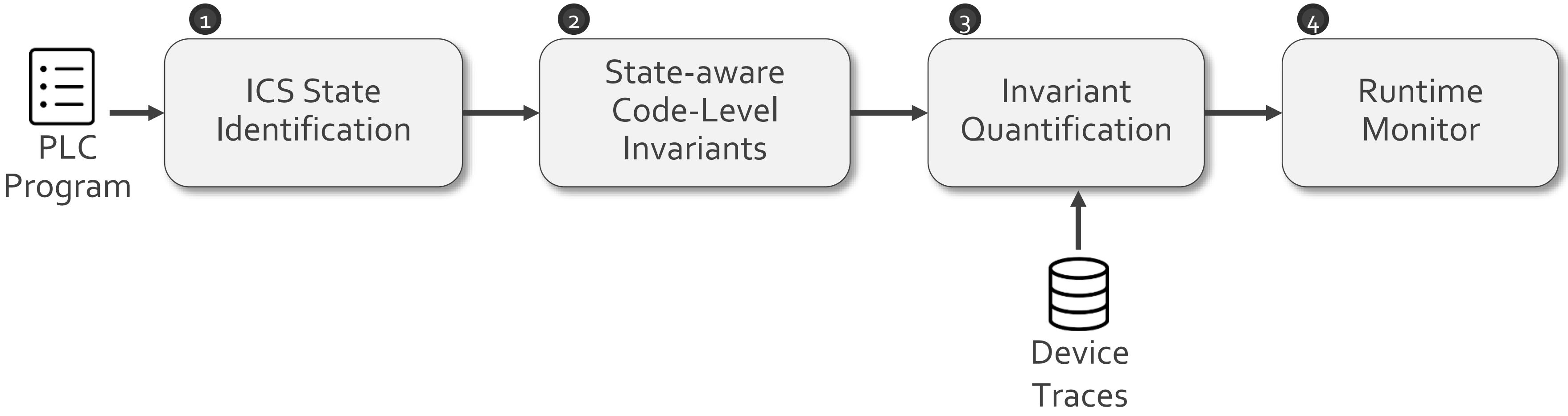
Problem w/ SOTA: State-Agnostic Invariants

- The value bounds for an invariant vary depending on the current ICS state



Our Solution: SAIN

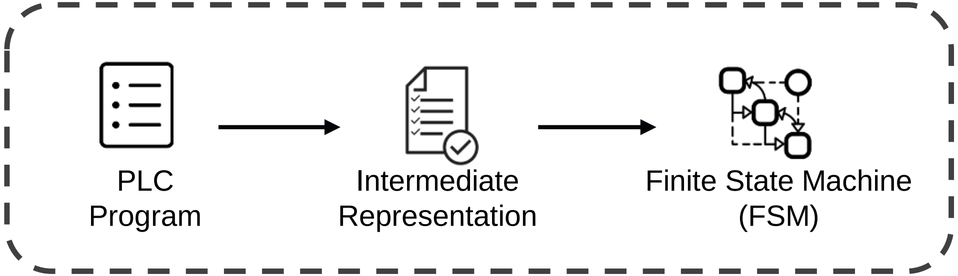
- We propose SAIN for deriving **S**tate-**A**ware **IN**variants to improve ICS attack detection sensitivity against stealthy evasion



Step 1: ICS State Identification

- PLC program executes as an FSM with distinct states in accordance with its control flow

- SAIN analyzes PLC program to identify the states and FSM



Step 2: State-aware Code-Level Invariants

- Intra-State Invariants

  Single-Variable: $V_2 = 30$

  Multi-Variable: if V_1 = True

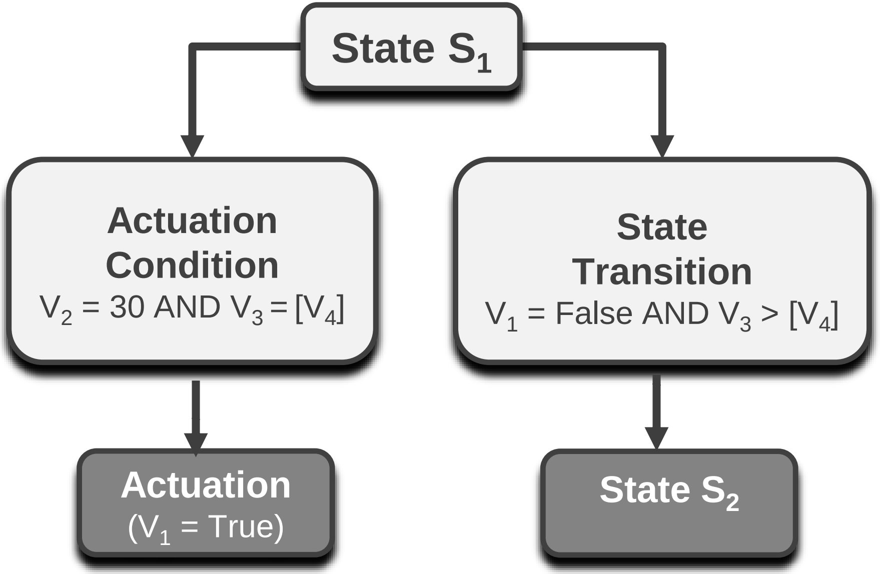
  then $V_2 = 30$

  and $V_3 = [V_4]$

- Inter-State Invariants

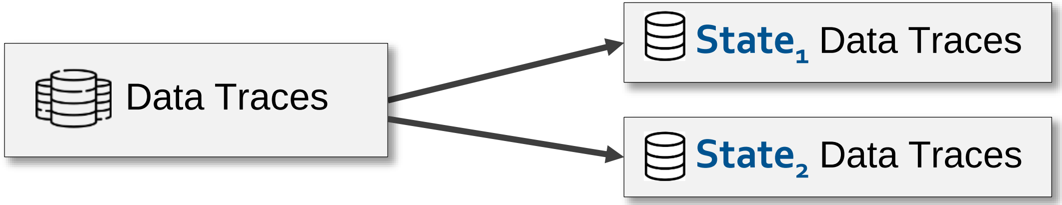
  if State2 then

  V_1 = False and $V_2 > [V_4]$

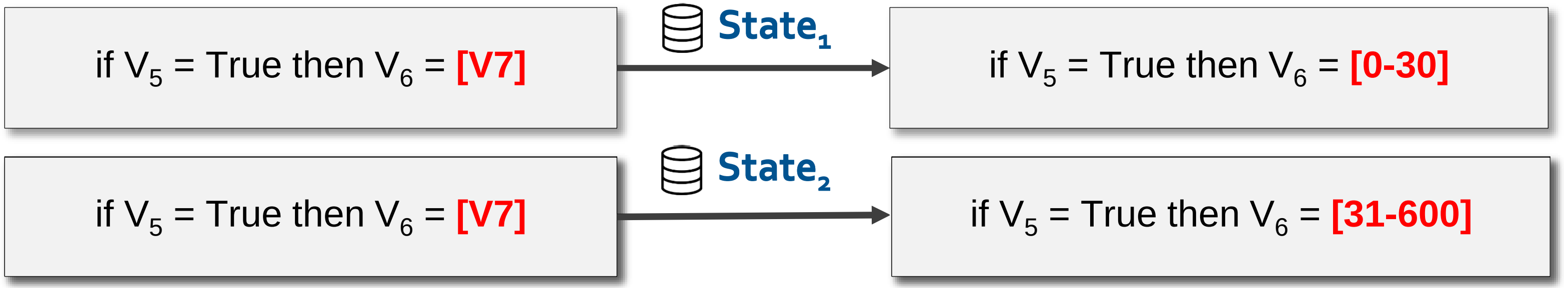


Step 3: Invariant Quantification

- Extracts state-specific sub-traces

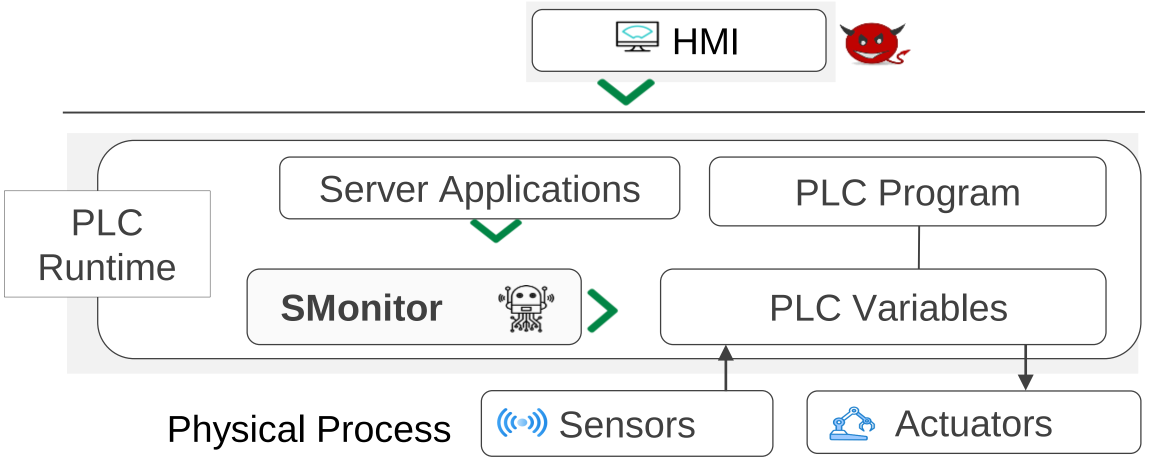


- Quantifies the "unresolved" invariant values



Step 4: Runtime Monitoring

- SAIN enforces state-aware invariants via a **SMonitor Agent**



Evaluation

- Two ICS simulation/emulation platforms
  - Manufacturing plant
  - Chemical plant

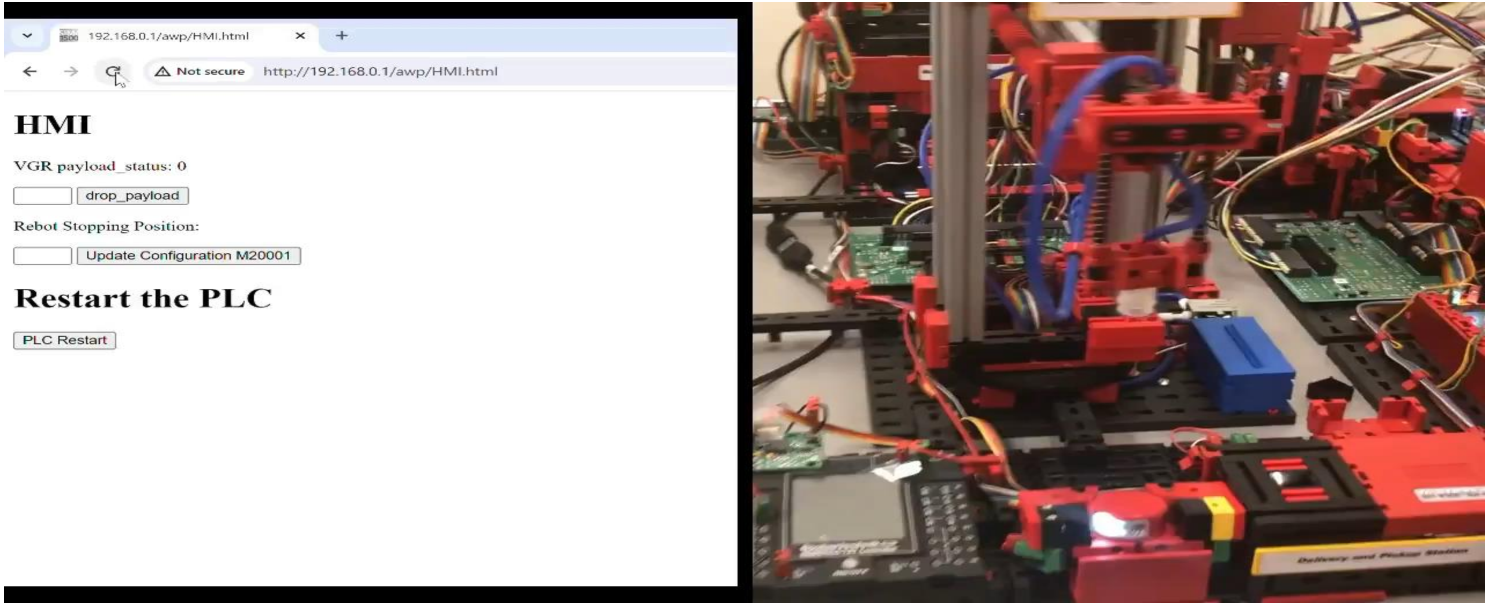
- 17 Attacks
  - Sensor variable manipulation
  - Malicious configuration
  - Malicious actuator commands

Results

- SAIN detects all 17 intra-state and inter-state attacks.
  - SAIN detects 7 attacks with single-variable state-aware invariants and 10 attacks with multi-variable invariants.
- In contrast, state-of-the-art method detects 35% of attacks.
- SAIN incurs, on average, a false positive rate of 2% and a run-time overhead of 3%.

Case Study Demo

- Attack: PLC variable manipulation to drop a workpiece.



Conclusion

- Existing ICS invariants are state-agnostic, leading to loose bounds and detection evasion
- State-aware invariants achieve tighter, state-dependent bounds and improve detection sensitivity
- SAIN enables offline generation and runtime enforcement of state-aware invariants, with high detection accuracy and low overhead

Thank you! Questions?

abbas4@purdue.edu

https://github.com/purseclab/SAIN

Acknowledgements

NSF, CNS, ONR, Cisco

PURDUE UNIVERSITY

CERIAS

PurSecLab