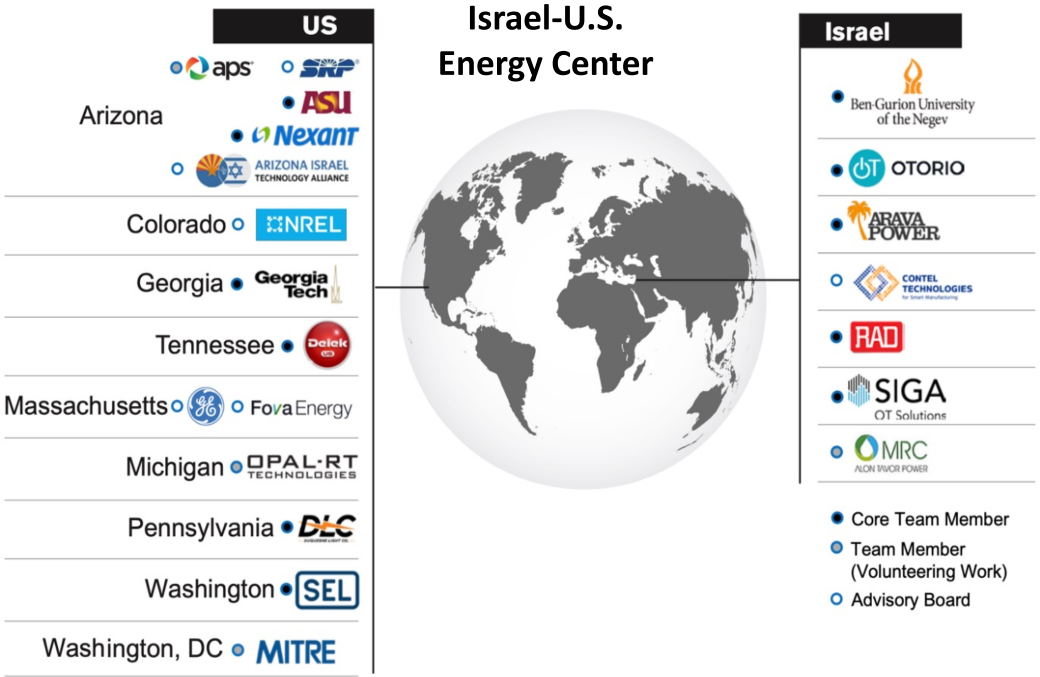
# Cybersecurity Technology for Critical Power Infrastructure AI-Based Centralized Defense and Edge Resilience




BIRD Workshop at ASU
October 9 – 11, 2023

## Task 7:
## Practical Deployment Architecture for Alerting Malware Activity in ICS/SCADA Networks

Dr. Wenke Lee, Moses Ike
(Presented by **Bo Feng**)
Georgia Institute of Technology

# Background

Industrial Control Systems (ICS)
Operational Technology (OT)
Cyber-Physical Systems (CPS)

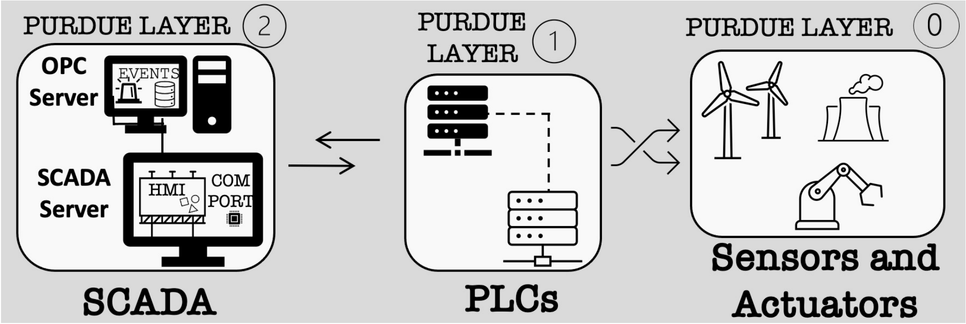
Power Substations

Water Treatment Plants

Nuclear Plants

Supervisory Control and Data Acquisition (SCADA) Systems

Programmable Logic Controllers (PLC)

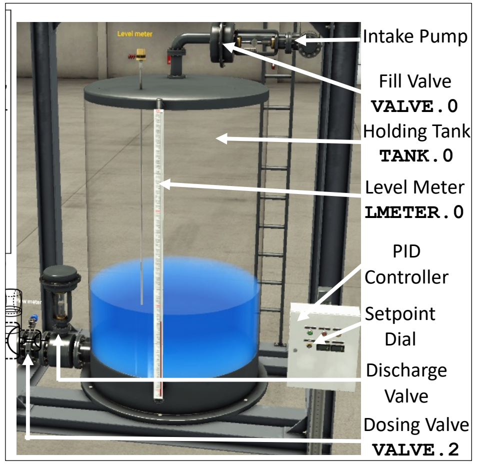


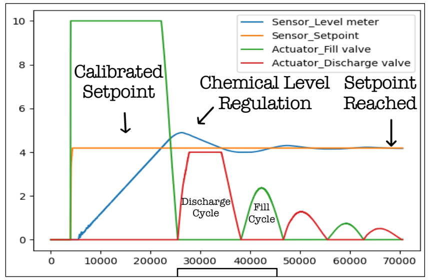

The Purdue ICS Network Architecture

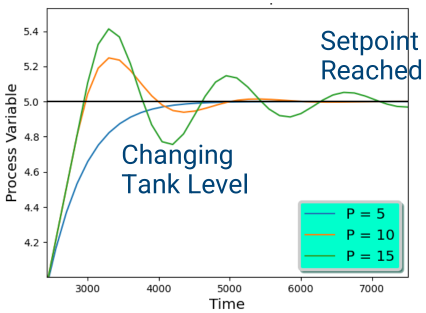
# (Control) Cause and (Physical) Effect

Unlike IT, OT has Cause and Physical Effect

- Control inputs result in physical actuation
  - Control inputs are generated by a software logic
  - Physical actuation are governed by the ”Physics” of devices (physical process), and measured by sensors



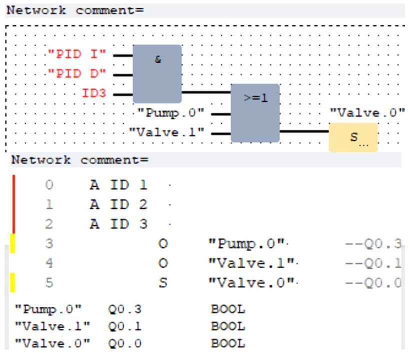




## Two (2) Types of Control

### Basic Control

~1 - 10ms (done by PLCs)

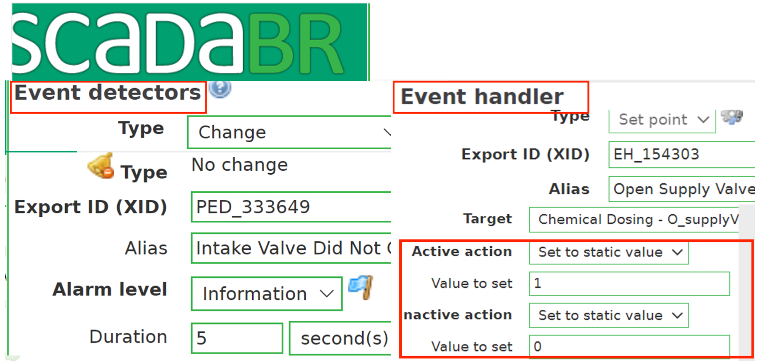
Programmable Logic Controllers (PLC)




### Supervisory Control

~1000ms (done by SCADA)

Supervisory Control and Data Acquisition (SCADA) Systems

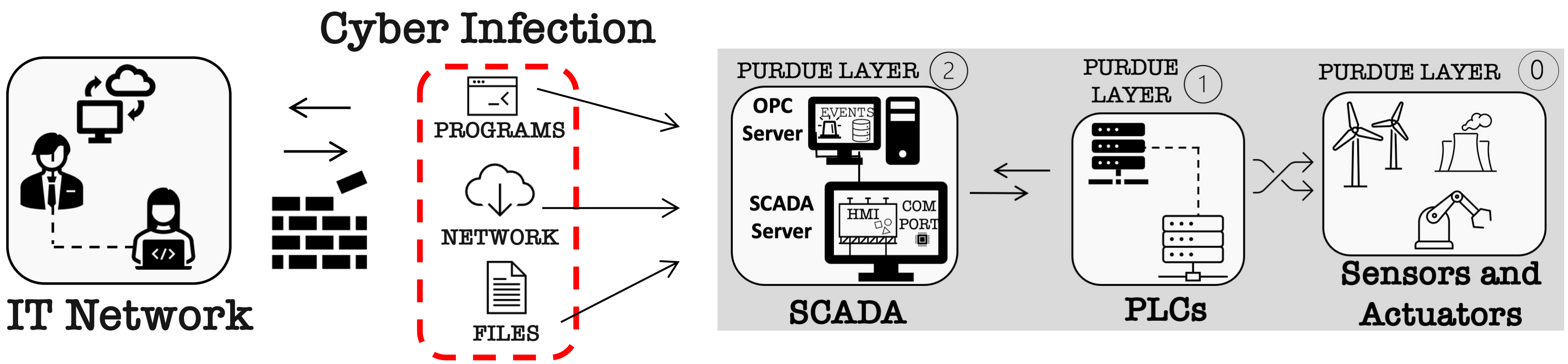



Georgia Tech

# Threat Model (We assume SCADA is compromised)

## Modern ICS is no longer "air-gapped"…. prone to cyber infection

- SCADA connects to the IT network and internet, utilizes cyber resources
- Modern attacks infect SCADA to disrupt physical processes



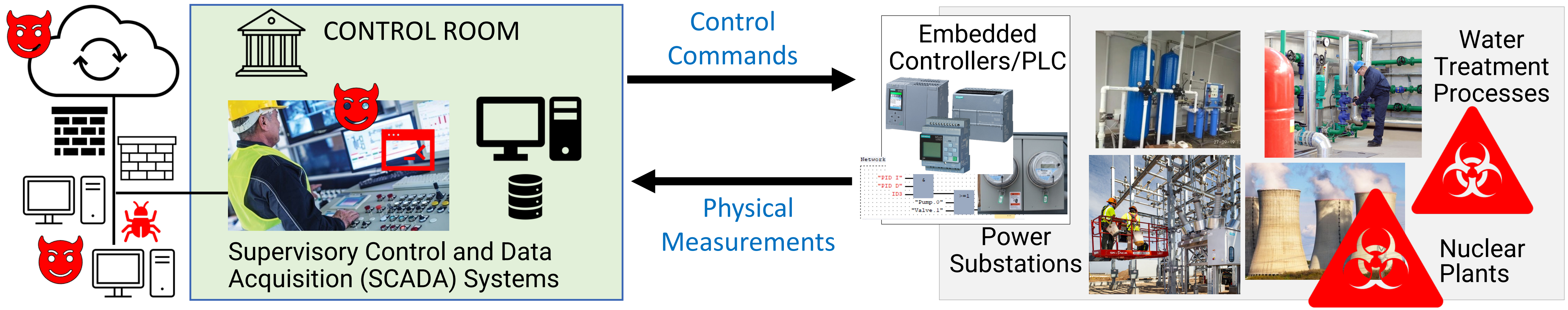

## A note on SCADA Analysis

- Network Traffic vs. Host Execution

**THREAT MODEL**
SCADA is compromised or Attacker resides in the SCADA Systems, and aims to cause physical disruption or damages

Georgia Tech

# Malware Attacks are a big problem in SCADA/ICS Networks




Software-based systems execute physical domain operations ⟺ SEMANTIC GAP ⟺ Physical processes, actuators, sensors exhibit continuous “physics” behavior

# Modern ICS/SCADA Host Attack Challenges

## INFECT SCADA -> STAGE PAYLOAD -> ATTACK PROCESSES

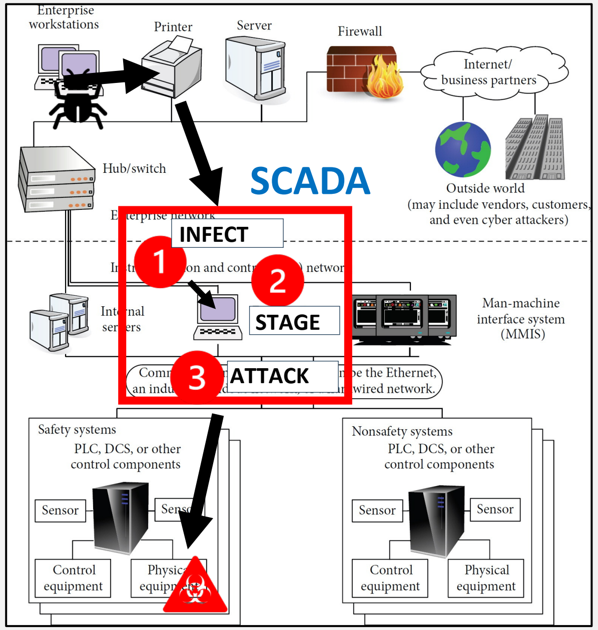
### PAST MALWARE ATTACKS

**(LAUNCHED VIA THE SCADA HOST)**

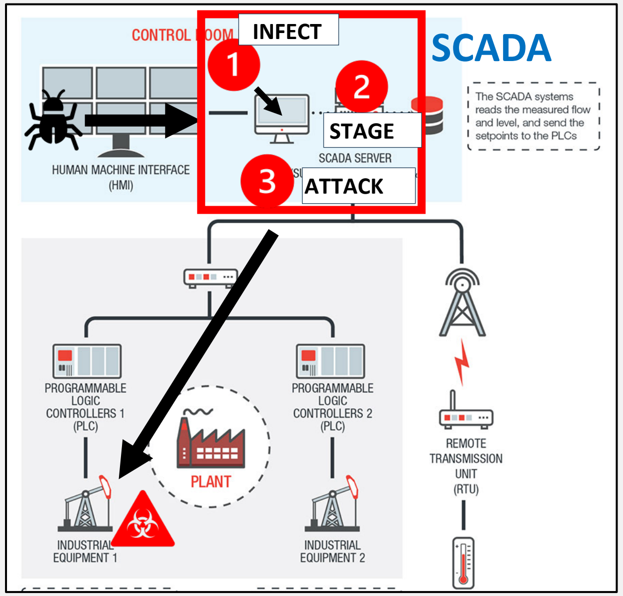
- **2010 Stuxnet:** Iran centrifuge system
- **2014 Havex** (various organizations)
- **2016/2022 Industroyer:** Ukrainian Power
- **2021 Oldsmar:** Water Treatment Plant
- **2021 Colonial Attack:** Oil and gas Pipeline

Examples

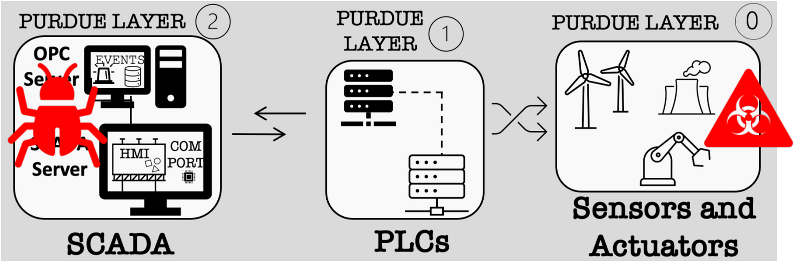
**STUXNET ATTACK**




**INDUSTROYER ATTACK**



# Limitations of Existing ICS Defenses




- **Physical Measurements:** Flag anomalous sensor readings at runtime. First learns the normal "Physics" of the ICS process
    - High false alarms due to benign physical noise, faults, and errors
    - Cannot validate flagged anomalies
- **PLC Defenses:** Prevent PLC logic/code modification
    - A SCADA adversary can modify PLC I/Os without touching its logic
- **Host-based (SCADA)**: Flag anomalous host API calls at runtime
    - Attackers also use normal tools/API calls. Requires knowledge of operational events or what triggers SCADA
    - Cannot know if anomalous APIs cause negative physical effects

## Our Insight: Correlate cause and effect via SCADA-Physics Anomaly Correlation

- Start from SCADA, where the attack is launched from, and find anomalies
- Then use Physics to corroborate (or filter) detected SCADA anomalies

Georgia Tech

# SCADA-Physics Anomaly Correlation:
## Summary/Roadmap

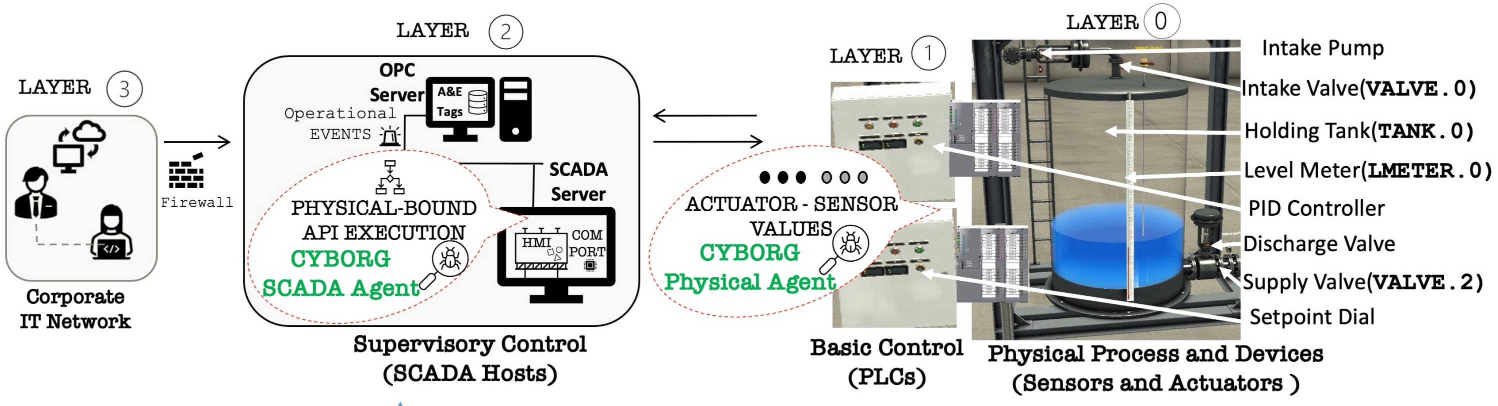
- **SCADA Host Analysis**
  - Use SCADA operational events to Induce SCADA “Physical-bound” API Calls
  - Analyze their statistical (Frequency and Timing) dependencies. Model SCADA operational semantics

- **Physics Analysis**
  - Use sequence-based neural network to learn normal sensor/actuator time series
  - Apply Transformer-based Autoencoders to rank important physics relationships
  - Leverage “Inertia” to inform effect neural network sequence length.

- **SCADA-Physics Correlation**
  - Align the anomaly correlation time-window (i.e., Physics lag behind SCADA due to inertia)
  - Anomalies that show up at both behaviors, is a strong indicator of an actual attack
    - In practice, we use Physics anomalies to filter or corroborate SCADA anomalies
  - False positives: Physics anomalies that permeate from previous time windows before SCADA anomaly

Georgia Tech

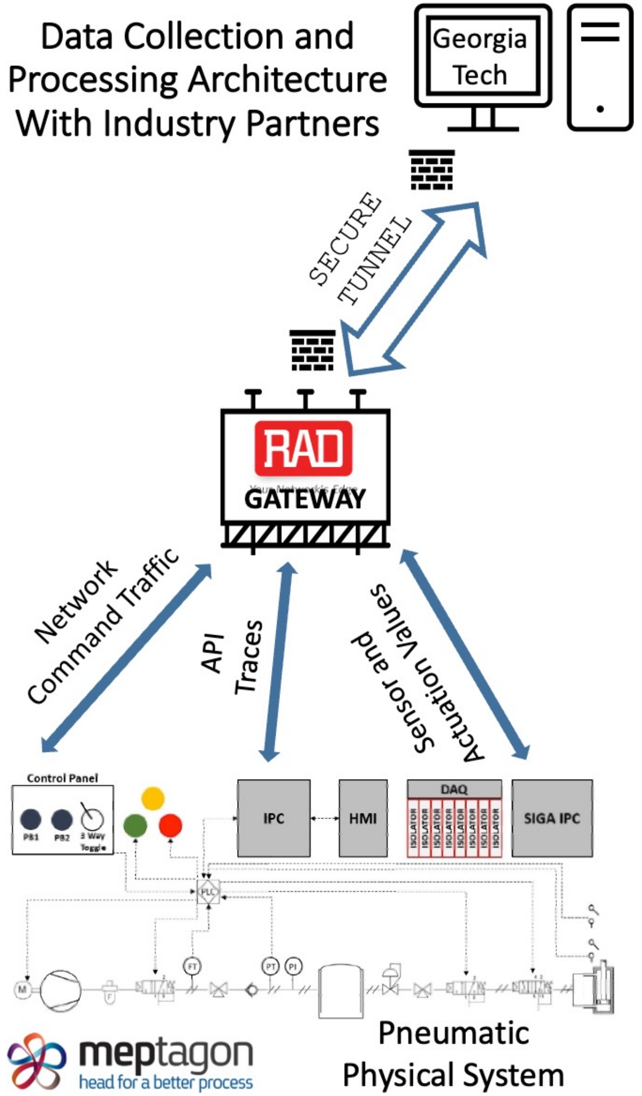
# SCADA-Physics Anomaly Correlation: Example Deployment




**ScadaBR** **OpenPLC** **FactoryIO**

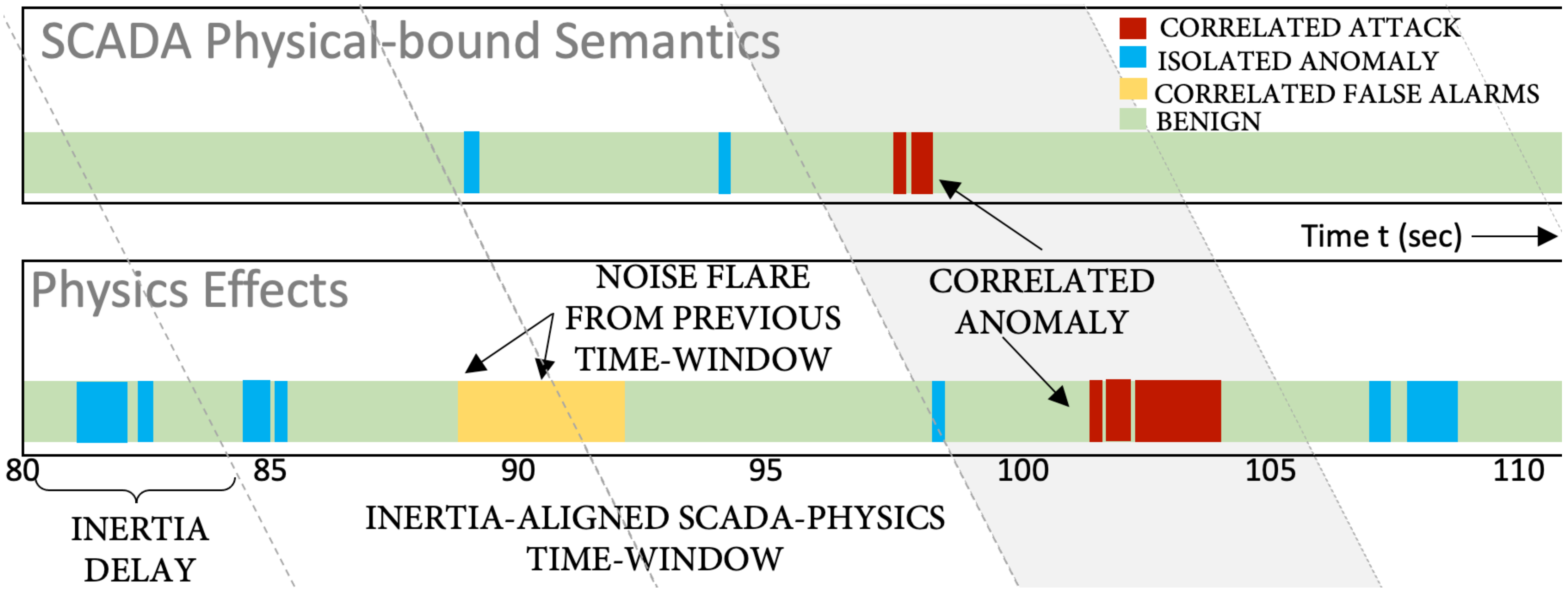
Tools used in our test environment

Data Collection and Processing Architecture With Industry Partners



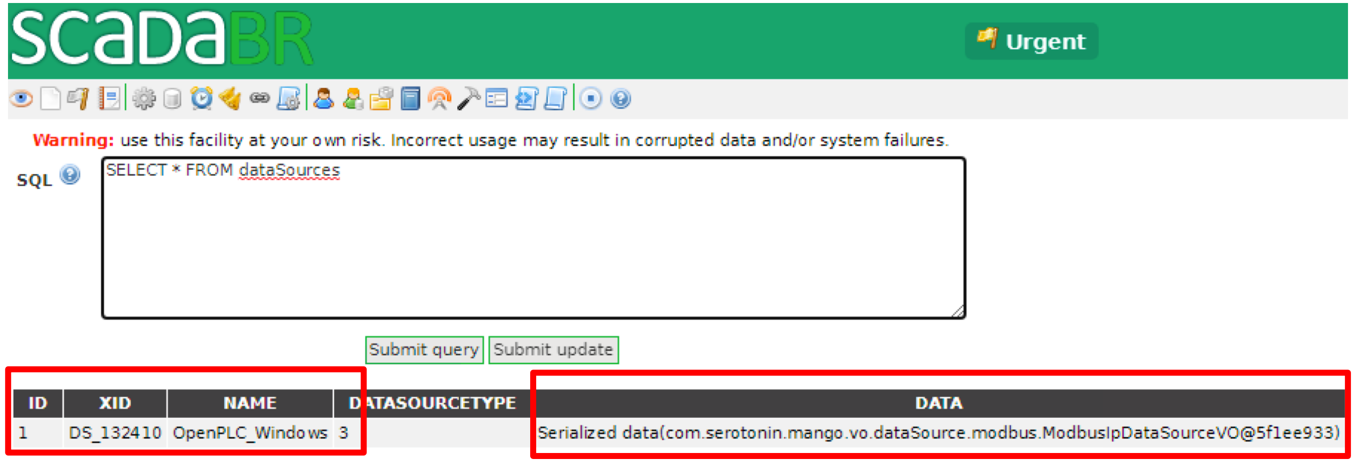

Georgia Tech

# SCADA-Physics Anomaly Correlation

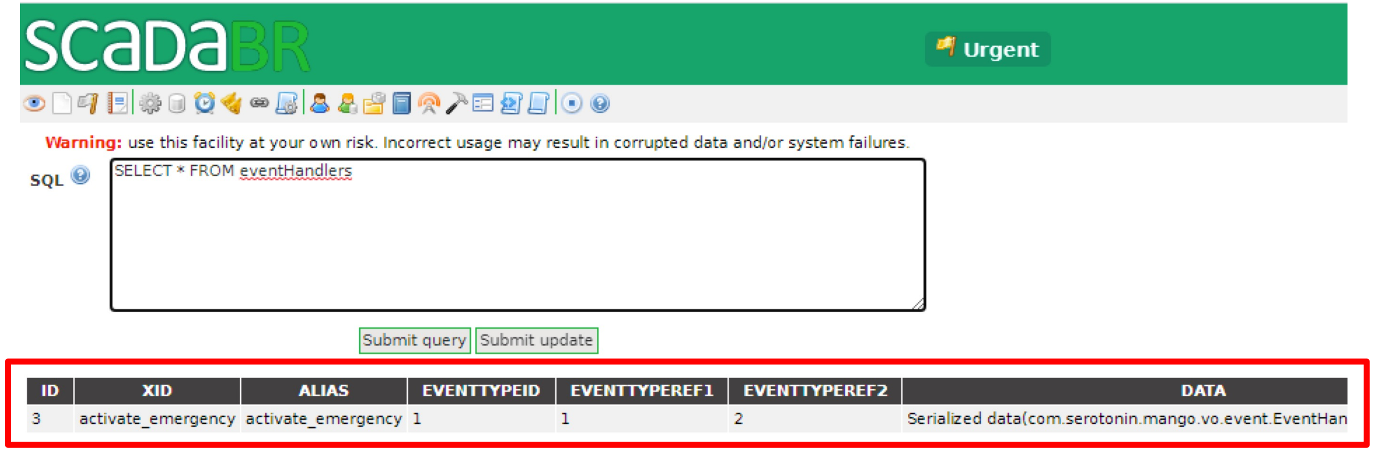



Georgia Tech

# SCADA Operation in ScadaBR




View Connected PLCs

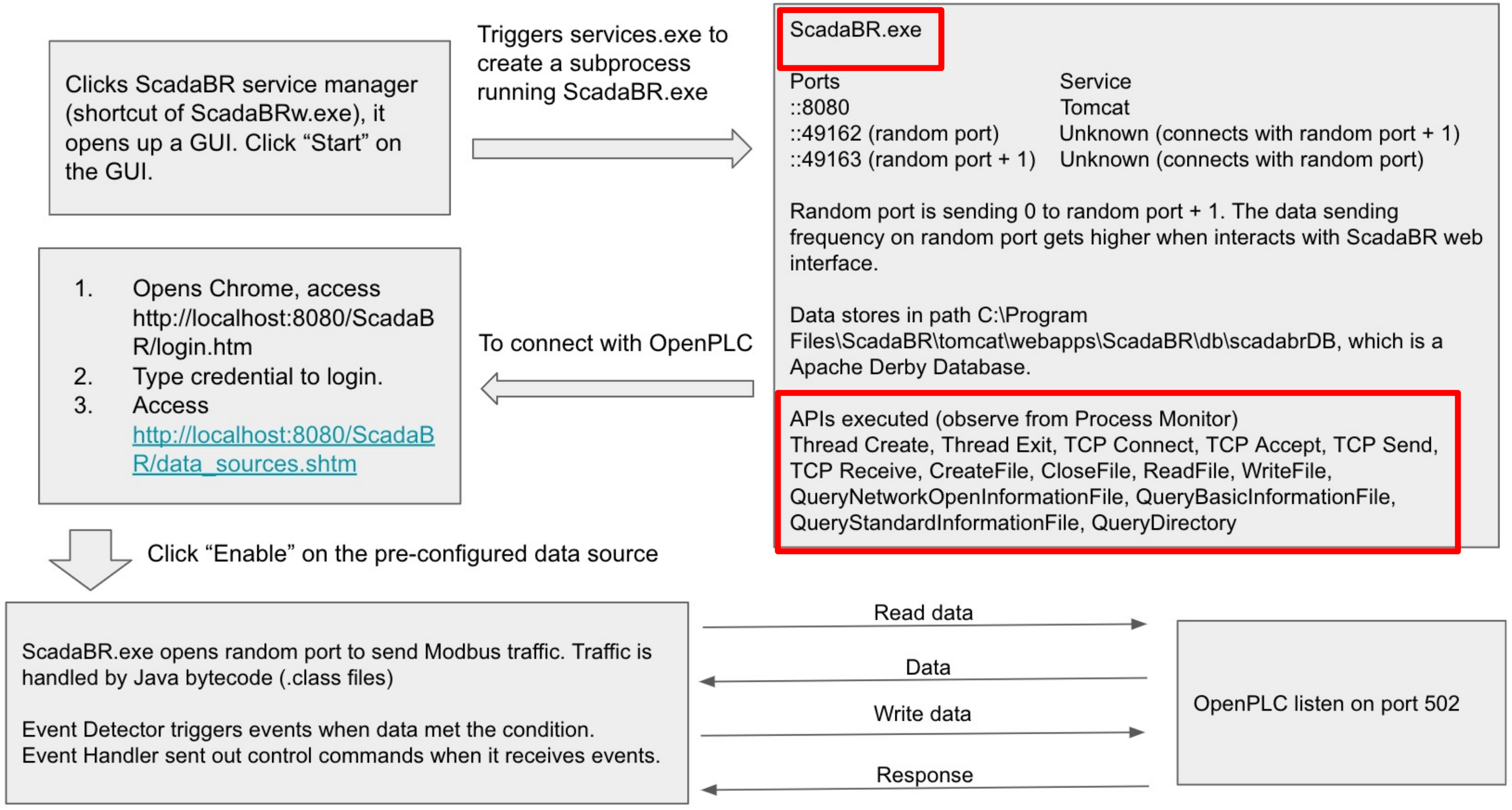



View Configured Events and Event Handlers

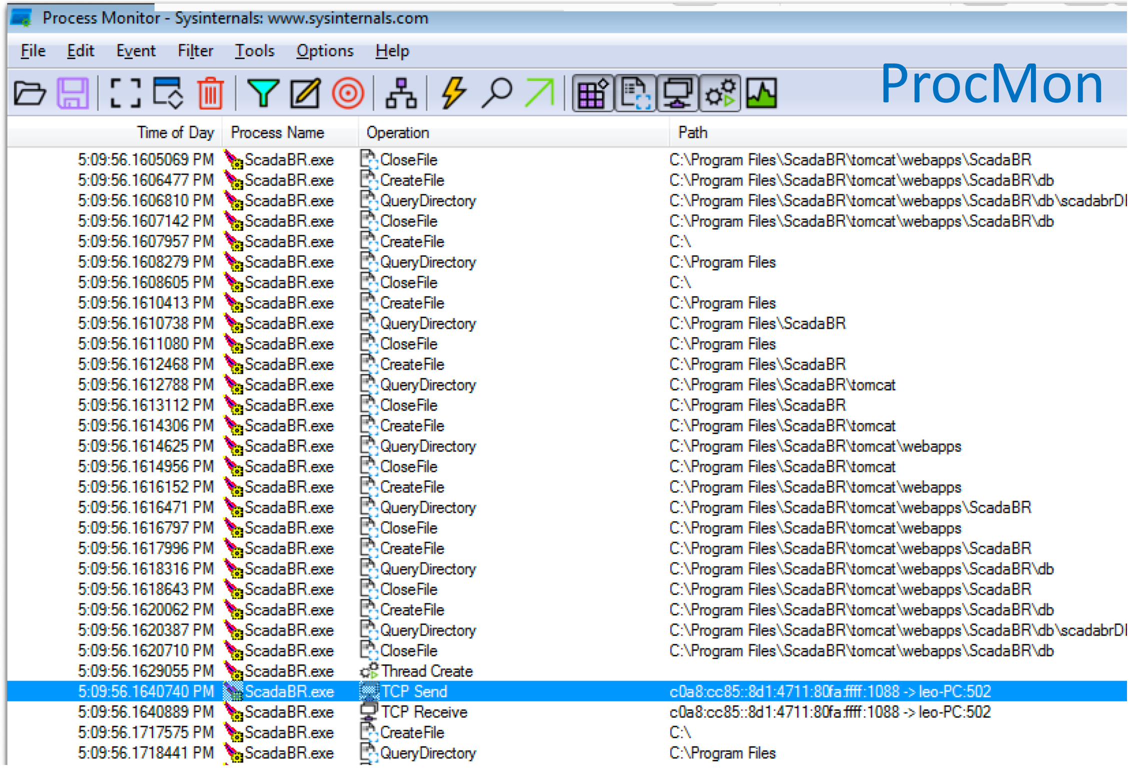
# Analysis of SCADA Host Execution

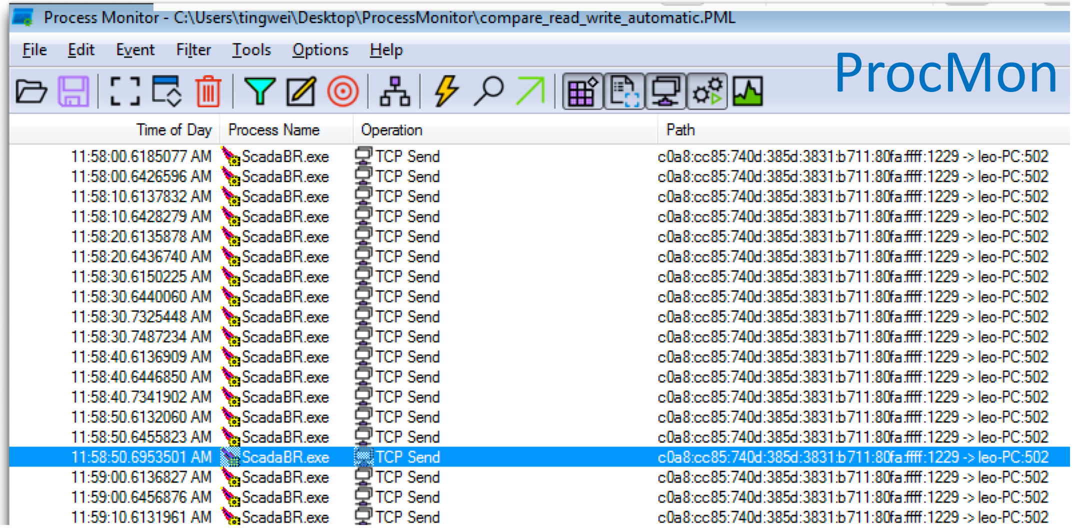
- Analyze “Physical-bound” API calls (not all API calls)
  - API call execution used to control the physical world
  - Must first identify the appropriate SCADA software process (done once)

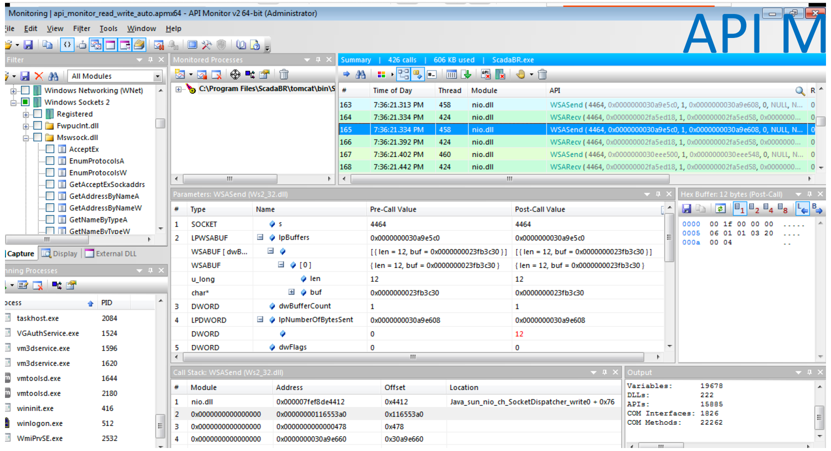
## [Dissecting SCADA Internal Host Interactions]
(Note that every SCADA System will be different)

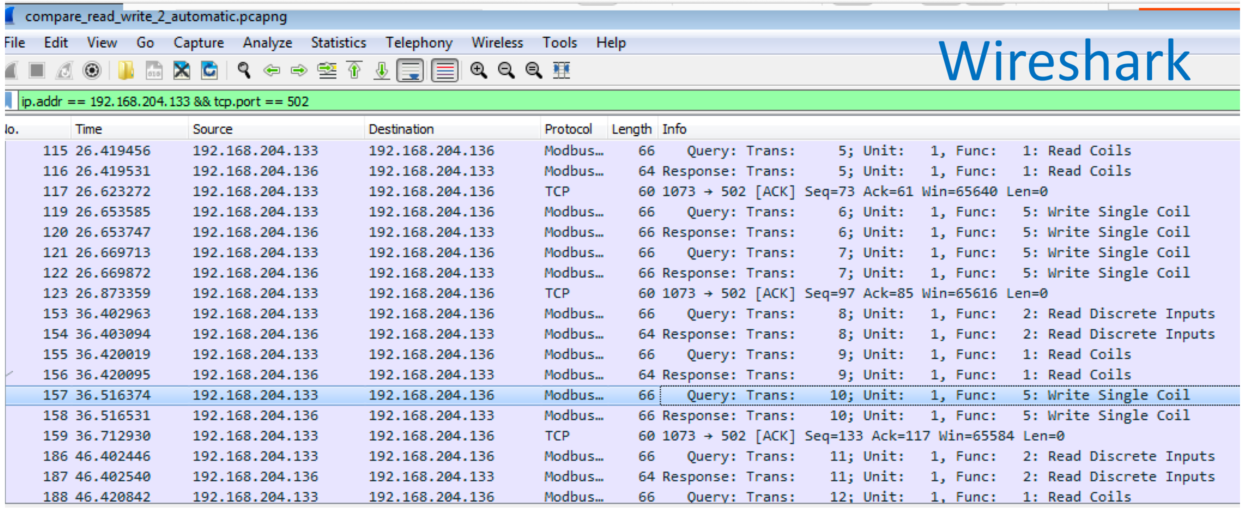


# Analysis of SCADA Host Execution



# Statistical Analysis of SCADA Physical-bound API Calls

## Formulation of Frequency and Timing Dependencies: Algorithm Development

### Control Command Dependency

$$P(V_k) := \{C_j, C_{j+1}, .....C_n\} \cup \{M_j, M_{j+1}, .....M_n\}$$

$$\forall M_i, C_j \in P(V_k) \wedge (ts(M_i) < ts(C_j)) : \quad C_j \hookleftarrow C_i$$

### Control Time-Interval

$$\forall C_i, C_j \in P(V_k) \;\; s.t. \;\; i \neq j : \quad \Delta(i,j) := ABS(ts(C_i) - ts(C_j))$$

$$R_{D\Delta}(i-1, i) = \frac{Deviation(j) + \epsilon_{(i-1,i)}}{Mean(j)}$$

### Control Burst-Interval

$$(\forall B_{C_i}, B_{P_i} \in P(V_k) : \quad \mu_j := |B_{C_i}| - |B_{P_i}|$$

$$R_{D\mu}(i) = \frac{Deviation(j) + \lambda_{(i)}}{Mean(j)}$$

### Control Frequency

$$\forall C_i \in P(V_k) \quad F(i) := |C_i|$$

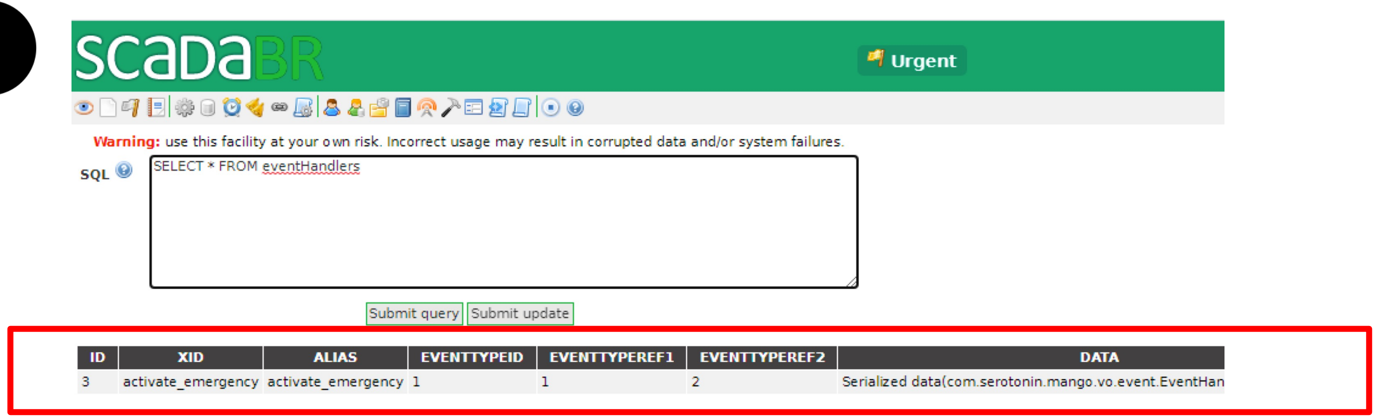
$$R_{DF}(i) = \frac{|C_i \in P(V_k)|}{|P(V_k)|}$$

Georgia Tech

# Inducing SCADA Events

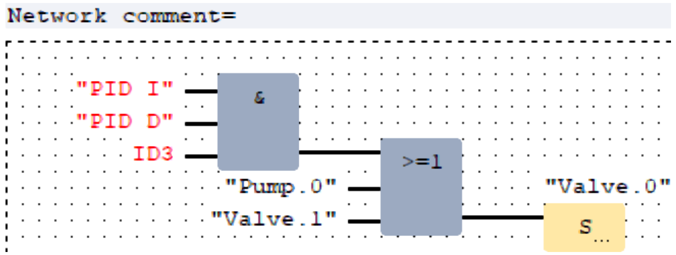
## Operational Events are configured in OPC Alarm&Event databases

1




Extract all configured events and device parameters

2




Statically parse the Function Block Diagram of the process to extract all dependent device states

3

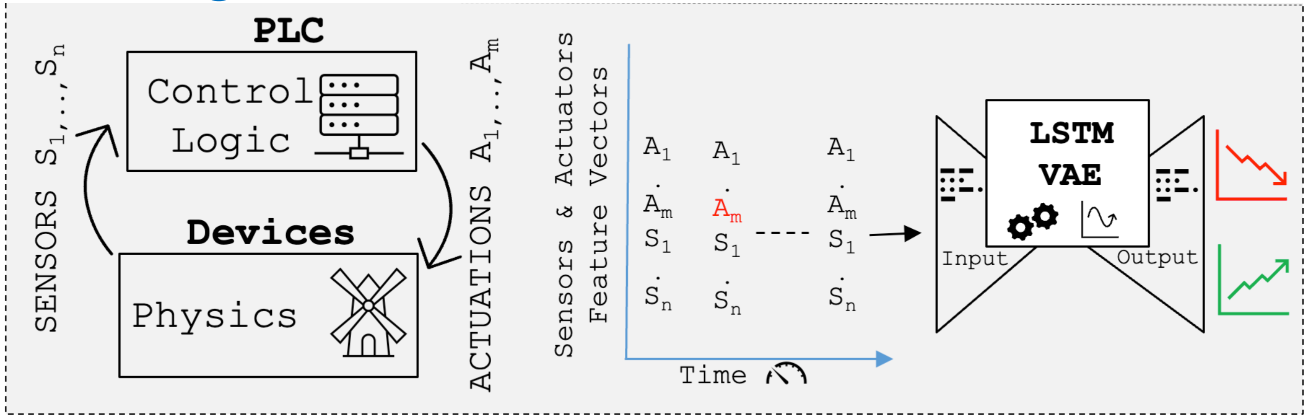
$$\forall k \in Events\_Set\ V; \quad \forall D_i \in k_D; \quad k_T := \bigcup_{i=0}^{n} \{D_i(S)\}$$

Union all event states and inject into ScadaBR iteratively

Georgia Tech

# Physics Analysis

Learning normal sensor and actuator time series

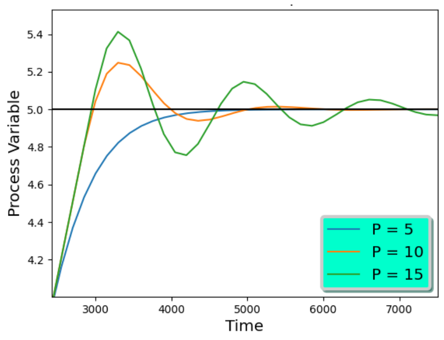


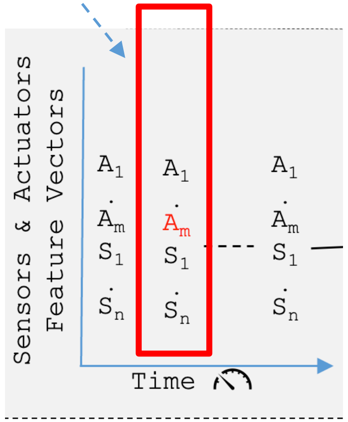

Autoencoder
Error Re-construction

$$MSE(x,\tilde{a}) = \frac{1}{m}\sum(x^i - \tilde{x}^i)^2$$

Actuator and Sensor time slice may not be causally related

Actuators are periodic in nature and not polled at same exact times, e.g., rising edge or peak values



# Physics Analysis (Cont’d)

Learning normal sensor and actuator time series


Conveyor driven by a large spinning disc experiences inertia

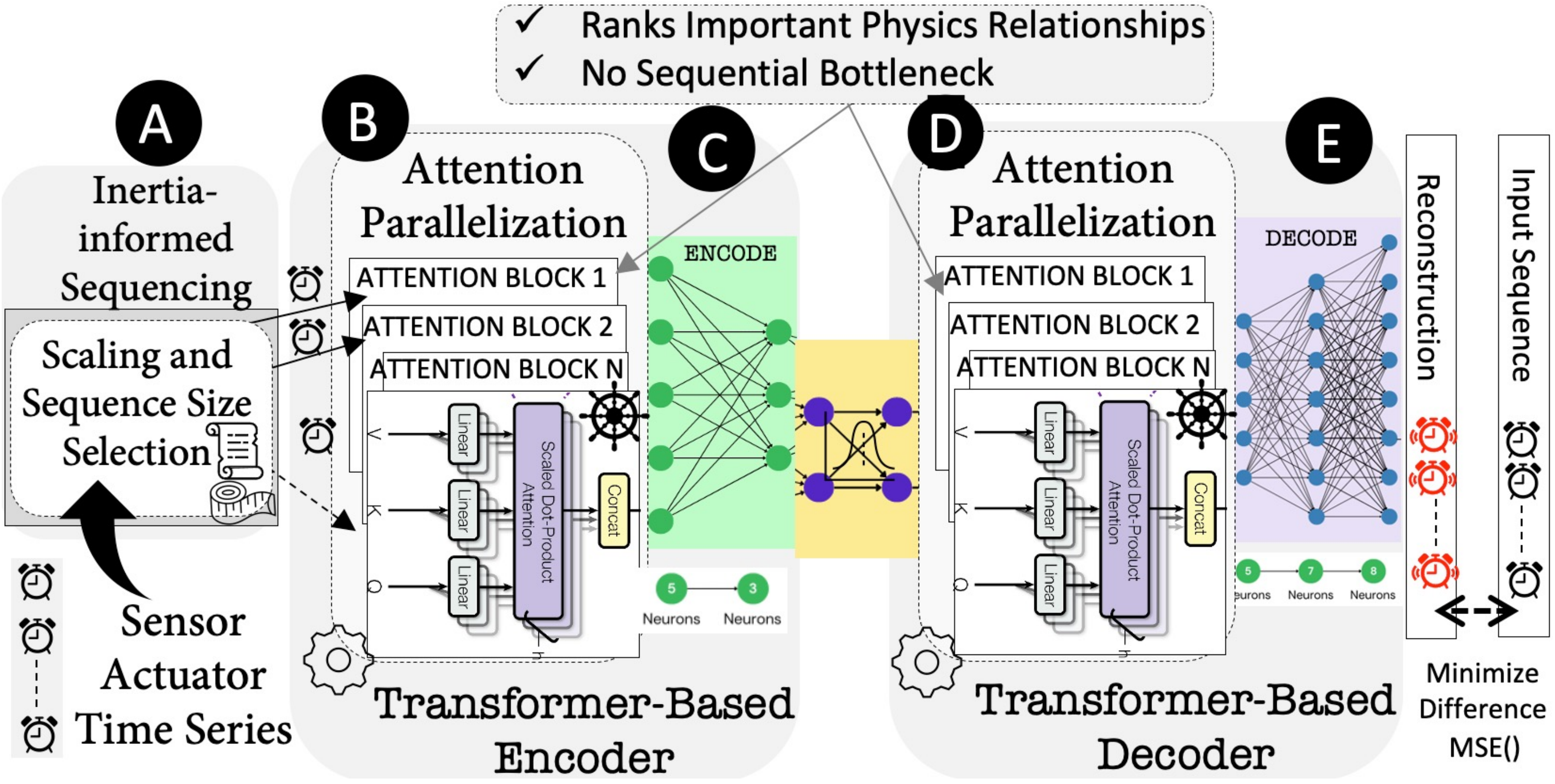
## Other challenges (for correlation with SCADA):

1. Physics have inertia (slow to respond to changes)
   - We use a data driven approach to infer inertia delays for each process
2. Neural networks have a bottle neck (more delays)
   1. We leverage Transformers

## Transformer-based Sequence Models

1. State-of-the-art sequence modelling technique
2. Uses Attention technique to rank causal relationships in time-series data
3. Can parallelize multiple Attention Blocks, eliminating the bottle neck in sequence models

# Transformer-Based Autoencoder

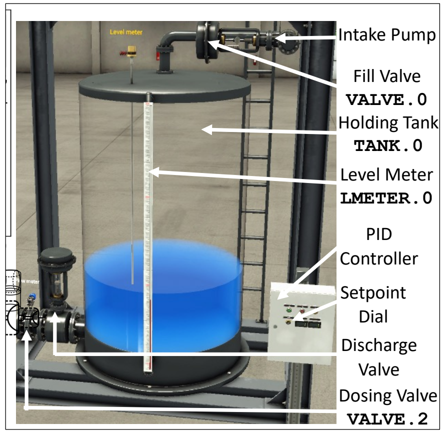


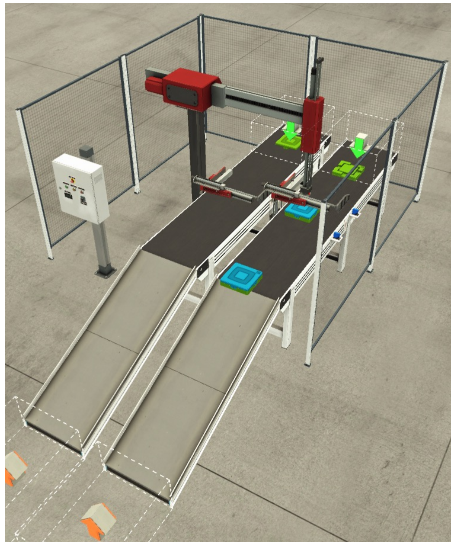

Reference Paper: Attention is All you Need
https://arxiv.org/abs/1706.03762

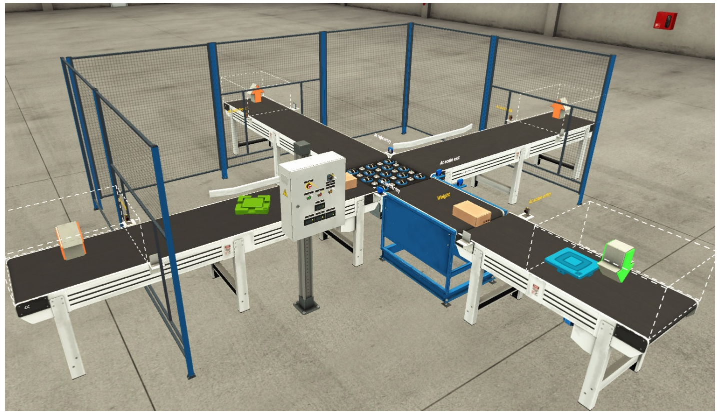
Georgia Tech

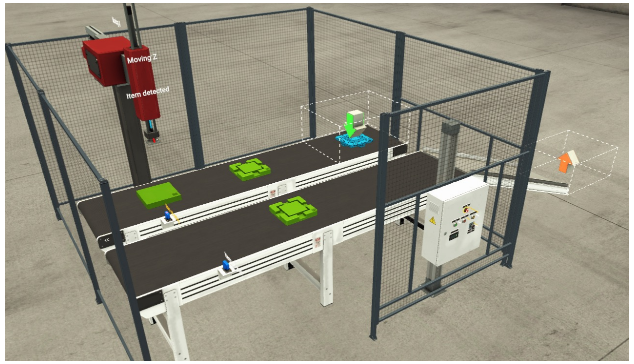
# Physical Experiments

## Emulated in FactoryIO





| ICS Processes | Domain | Events Per process | | | Event-Guided Dependency Analysis | | | | | | $R_D$ Averages | | | | Physics Analysis (Training) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | *SOD* Per Proc | | | | | | | | | | | |
| | | Scenario FileSize | States | Verify (FN) | Task/Goal | Task ID | Event Device | CMDs | Nodes | Edges | $R_{D\Delta}$ | $R_{D\mu}$ | $R_{DF}$ | Analysis Time (min) | Task Inertia | Seq Size | Precision | Recall | F1 |
| HVAC | A/C | 6K | 18 | 0 | heat setpoint | 6.1 | room temp | 9 | 4 | 5 | 0.219 | 0.053 | 0.09 | 11.6 | 11.8 | 13 | 79 | 92 | 85 |
| | | 6K | 17 | 1 | heat flow | 6.2 | vent | 23 | 5 | 5 | 0.178 | 0.410 | 0.065 | 10.2 | 11.8 | 13 | 88 | 83 | 85 |
| Chemical Dosing | Water Treatm | 11.5K | 10 | 0 | level control | 2.1 | holding tank | 24 | 4 | 6 | 0.47 | 0.125 | 0.105 | 7.6 | 4.5 | 5 | 85 | 70 | 77 |
| | | 11.5K | 14 | 2 | dosing | 2.2 | dose valve | 22 | 5 | 9 | 0.419 | 0.111 | 0.344 | 7.5 | 4.5 | 5 | 94 | 87 | 90 |
| Auto warehouse | Manufacture | 29K | 20 | 0 | pallet alignment | 3.2 | Axes X,Z | 26 | 6 | 4 | 0.41 | 0.133 | 0.088 | 12.4 | 5.1 | 5 | 95 | 93 | 94 |
| | | 29K | 36 | 1 | throughput | 3.2 | entry conveyor | 32 | 7 | 5 | 0.42 | 0.167 | 0.034 | 12.9 | 5.1 | 6 | 77 | 83 | 80 |

# SCADA Attack Injection

## STUXNET-type Infection of ScadaBR

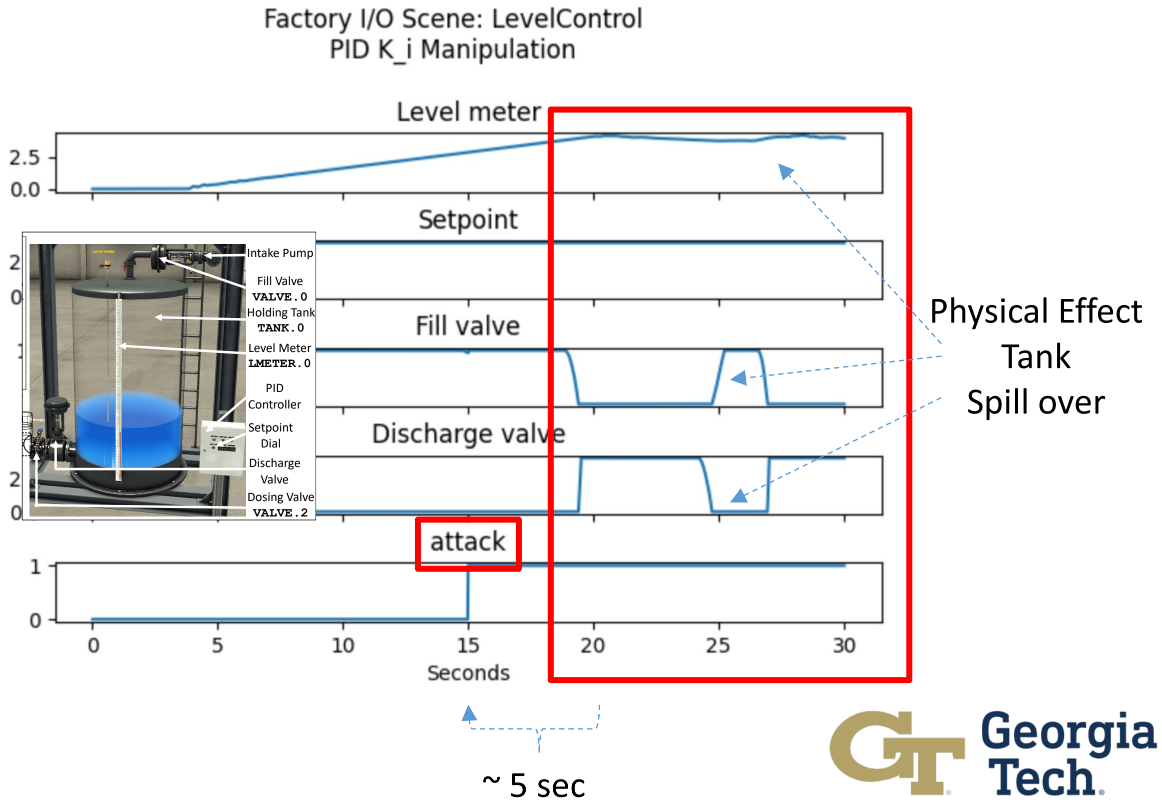
1. Clone the ScadaBR repository. (https://github.com/ScadaBR/ScadaBR)

2. Modify ModbusDataSource.java and re-compile → ModbusDataSource.class. The attack source code can be found in Attack_Scripts/ModbusDataSource.java.

3. ModbusDataSource.class will read a Stuxnet-type attack config file to perform attack in Files\ScadaBR\tomcat\conf\AttackConfig.txt

4. Replace the compiled ModbusDataSource.class with the current installed ScadaBR file C:\Program Files\ScadaBR\tomcat\webapps\ScadaBR\WEB-INF\classes\com\serotonin\mango\rt\dataSource\modbus\ModbusDataSource.class.

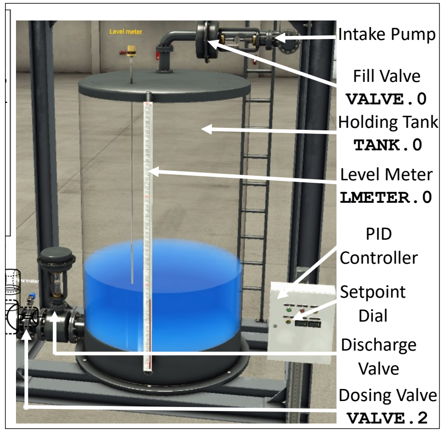
5. Restart ScadaBR. The attack will be activated when the right physical conditions exist. The attack log will be created in C:\Program Files\ScadaBR\tomcat\logs\AttackLog.txt.

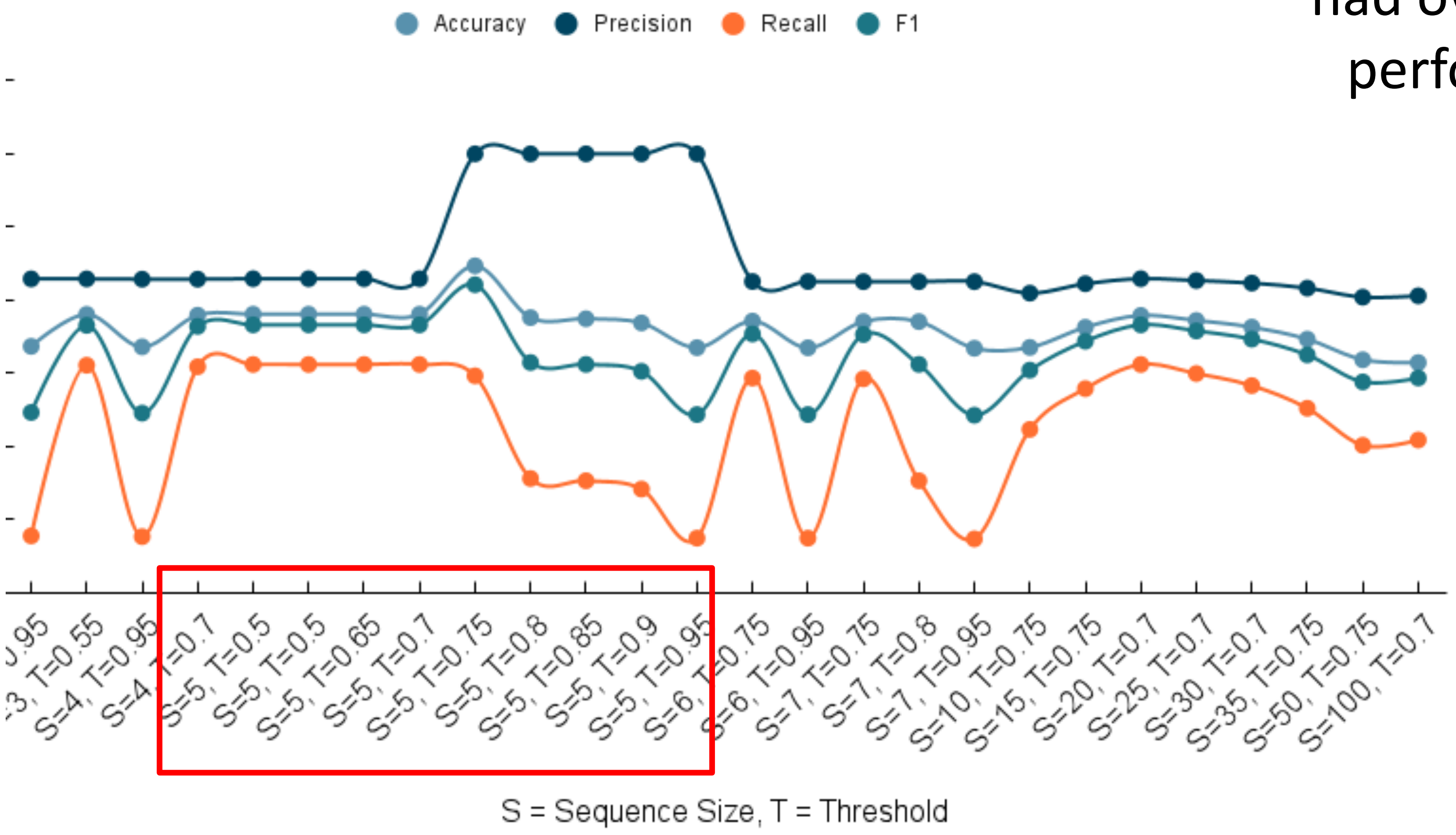



Georgia Tech

# Preliminary Results

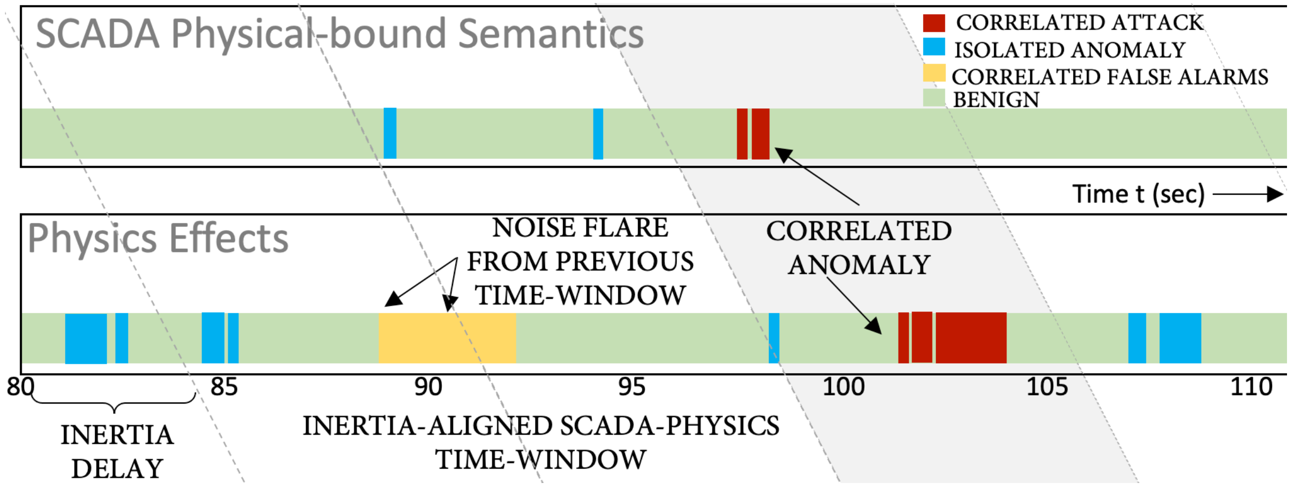
## Inertia-Informed sequence sizes




Sequence size of 5 had overall best performance

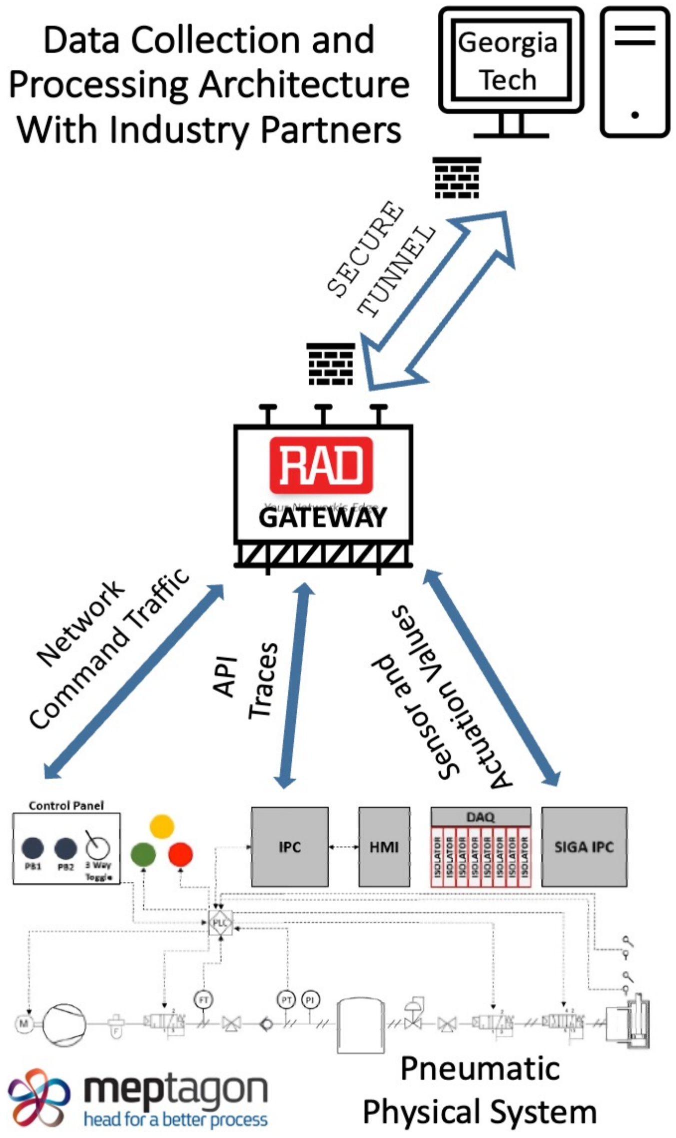
# Preliminary Results
## Anomaly Correlation




| Attack Category. | Total Attacks | TTP ID | Attack Description | TTP Goal/Device | Attacked Process (IDs in Table II) | SCADA. Ano C-TIME | SCADA. Ano C-BURST | SCADA. Ano C-FREQ | Physics Anomalies | Correld. ATTACKS | TP | FP | Avg. Detect Time (sec) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I. Stuxnet-Category | 10 | T831 | Control Manipulation | Impact Control | 2.1, 2.2, 6.2, 9.1, 9.2<br>10.1, 10.2, 3.1, 3.2, 6.1 | 3 | 7 | 4 | 21 | 12 | 12 | 0 | 10.2 |
| II. Industroyer Category | 10 | T855 | Unauthorised Command | Impair Process CTRL | 1.1, 1.2, 11.1, 7.1, 7.2<br>8.1, 8.2, 11.2, 9.2, 9.1 | | 8 | 5 | 19 | 10 | 9 | 1 | 7.8 |
| III. Oldsmar-Category | 10 | T836 | Modify Parameter | Impact Control | 4.1, 4.2, 8.1, 8.2, 11.2<br>7.2, 3.1, 3.2, 10.1, 10.2 | 17 | 11 | | 35 | 24 | 24 | 0 | 5.4 |
| IV. Triton-Category | 10 | T801 | Corrupt Process State | Inconsistent State | 6.1, 6.2, 8.1, 5.1, 5.2<br>6.1, 6.2, 9.1, 9.2, 7.1 | 8 | 2 | 13 | 27 | 22 | 22 | 0 | 9.4 |

Georgia Tech

# Summary of our work (SCAWATCH)




- SCAWATCH is a **passive alerting system** to alert ICS operators of suspicious malware activities in SCADA host
- We are collaborating with industry partners, **Meptagon and RAD,** to integrate SCAWATCH to deploy in practical ICS/SCADA networks
- SCAWATCH detects attacks in SCADA via (1) statistical violations of process-control executions and (2) suspicious manipulation of SCADA physical-control resources (e.g., COM ports, PLC interfaces)
- SCAWATCH then correlates detected SCADA attacks with anomalies in running processes via a ML neural network framework.
- Tested on experimental data. Details and technical paper can be found in https://github.com/lordmoses/SCAWATCH.

# QUESTIONS

- Thank You