Control Behavior Integrity for Distributed Cyber-Physical Systems

Sridhar Adepu*[1], Ferdinand Brasser†[2], Luis Garcia‡[3], Michael Rodler⊕[4],
Lucas Davi⊕[5], Ahmad-Reza Sadeghi†[6], Saman Zonouz°[7]
*Singapore University of Technology and Design Singapore; †Technische Universität Darmstadt;
‡University of California, Los Angeles; ⊕University of Duisburg-Essen; °Rutgers University
adepu@sutd.edu.sg[1]; {ferdinand.brasser[2],ahmad.sadeghi[6]}@trust.tu-darmstadt.de; garcialuis@ucla.edu[3];
{michael.rodler[4], lucas.davi[5]}@uni-due.de; saman.zonouz@rutgers.edu[7]

***Abstract*—Cyber-physical control systems, such as industrial control systems (ICS), are increasingly targeted by cyberattacks. Such attacks can potentially cause tremendous damage, affect critical infrastructure or even jeopardize human life when the system does not behave as intended. Cyberattacks, however, are not new and decades of security research have developed plenty of solutions to thwart them. Unfortunately, many of these solutions cannot be easily applied to safety-critical cyber-physical systems. Further, the attack surface of ICS is quite different from what can be commonly assumed in classical IT systems.**

We present SCADMAN, a novel control-logic aware anomaly detection system for distributed cyber-physical systems. By observing the system-wide behavior, the correctness of individual controllers (like programmable logic controllers–PLCs) in ICS can be verified. This allows SCADMAN to detect a wide range of attacks, including malware attacks, code-reuse and data-only attacks, as well as sensor attacks. We implemented and evaluated SCADMAN based on a real-world water treatment testbed for ICS security research and training. Our results show that we can detect a wide range of attacks–including attacks that have previously been undetectable by typical state estimation techniques–while causing no false-positive warning for nominal threshold values.

I. INTRODUCTION

Industrial control systems (ICS) are used in a multitude of control systems across several applications of industrial sectors and critical infrastructures, including electric power transmission and distribution, oil and natural gas production, refinery operations, water treatment systems, wastewater collection systems, as well as pipeline transport systems [50]. *industrial control system* (ICS) typically consist of interconnected embedded systems, called programmable logic controllers (PLCs). In a distributed ICS, multiple PLCs jointly control a physical process or the physical environment. Using a series of sensors and actuators, PLCs can monitor a physical system's state and control the system behavior. This makes the correct functioning of PLCs crucial for the correct and safe operation of such systems.

This critical role of the PLCs makes them a valuable target for adversaries aiming to interfere with any of these systems [10]. Past incidences show that such attacks are applied in practice, often remaining undetected over a long period of time. Examples include the infamous Stuxnet worm [25] against Iranian nuclear uranium enrichment facilities as well as the BlackEnergy crimeware [24] against the Ukrainian train railway and electric power industries. These attacks demonstrate impressively that targeted attacks on critical infrastructure can evade traditional cybersecurity detection and cause catastrophic failures with substantive impact. The discoveries of Duqu [17] and Havex [48] show that such attacks are not isolated cases as they infected ICS in more than eight countries. Also academic attacks impressively demonstrate that the cyber-components are critical targets in an ICS when they are manipulated to change the physical process of the system [26]. Therefore, a comprehensive defense must follow a holistic approach to detect system misbehavior by treading cyber- and physical components in a unified way. For ICS-based verification techniques, it has been shown that state estimation can be used to infer the control commands issued by distributed controllers [22] or to detect false data injection attacks [41] based on the sensor data. Such protection mechanisms may be circumvented via physics-aware attacks [26], [28], as they cannot *precisely* model the system's control logic. Further, supervised machine learning has been used to characterize physical invariants of the CPS [15]. However, such approaches depend on the training data to include all corner cases of the system execution and are not based on the software behavior of the cyber-components.

Approaches that focus on ensuring the integrity of controllers' software, like PLC-based verification solutions, typically cannot account for attacks that replace and/or modify either the application layer programs or the underlying firmware. For instance, ECFI [3] provides protection against run-time attacks targeting PLC control programs, but does not protect against data-only attacks nor maliciously modified/replaced control programs or firmware. Orpheus [16] monitors the behavior of a device's control program based on the invoked system calls. Attacks are detected based on a finite-state machine (FSM) representing the control program's benign system call behavior. Orpheus' behavior monitor is placed inside the device's OS; hence, a compromised OS can disable and circumvent its protection mechanism.

Similarly, solutions that enforce compliance via state estimation [9], control invariant monitoring [18], or cyber-physical access control [23] from within the controller could be circumvented as well. Zeus [33] uses side-channel analysis to verify the software control flow of programs running on a PLC, but cannot defend against firmware modifications nor

2642-9500/20/$31.00 ©2020 IEEE
DOI 10.1109/ICCPS48487.2020.00011

sensor data attacks. By extension, offline, static analysis of control programs being loaded onto PLCs [44] [20] provides even less run-time guarantees.

We present SCADMAN, the first holistic control-logic aware state estimation solution for distributed industrial control systems. Unlike previous state estimation approaches SCADMAN does not abstract the behavior of cyber-components (i.e., PLCs). Instead, SCADMAN *precisely* simulates the state of all PLCs. By monitoring the input and output behavior of the entire ICS, SCADMAN can detect inconsistencies within the actions of PLCs. This makes SCADMAN agnostic to all attack techniques that can be used to cause a PLC to deviate from its intended behavior and makes SCADMAN a powerful tool to protect ICS against a wide range of attack vectors.

We evaluated SCADMAN on real-world industrial control system equipment [43], which are the quasi-standard for security research validation in the context of ICS [36], [40], [15], [14], [31], [38], [54], [53], [35], [51], [45]. Simulation-based evaluation does not provide a viable option for SCADMAN. In general, simulations are based on models of an ICS similar to SCADMAN. Hence, such an evaluation would validate the accuracy of our models against the model of the simulator–leading to no meaningful results.

We make the following contributions:

- We present SCADMAN, the first holistic state estimation solution for distributed ICS based on a model comprising cyber *and* physical components.
- Our solution does not require any changes to the hardware or software of the PLCs, making it independent of the PLC manufacturers. Furthermore, leaving the PLCs unmodified is important for safety certifications to remain valid in the presence of SCADMAN.
- We provide an automated solution that allows SCADMAN to consolidate the control programs of all PLCs in an ICS. This allows us to comprehensively simulate the control behavior of the entire system, which is important for detecting inconsistent behavior across the borders of individual PLCs.
- We implemented SCADMAN using the MATIEC compiler from the OpenPLC project for automated generation of the consolidated PLC control program and LLVM for instrumentation of the consolidated PLC control program.
- We evaluated SCADMAN on real-world ICS network equipment. The results were very promising. Using its runtime control behavior monitoring, SCADMAN was able to detect all the attacks of different types against the platform in a timely manner.

The rest of the paper is structured as follows. First, we provide background on ICS in general and cyber-physical system modeling Section II. We define the assumptions and system model underlying our work in Section III. In Section IV we explain the design and main ideas of SCADMAN. We detail our implementation in Section V. In Section VI we discuss SCADMAN's security for various attack scenarios and present our evaluation results in Section VII. Relevant related work is discussed in Section VIII, while Section IX concludes.

II. BACKGROUND

In this section we first provide background on industrial control systems (ICS) in general. Afterward, we introduce the central concept of physical state estimation (cyber-physical systems modeling) which has been used in the past in an attempt to secure ICS.

Industrial Control Systems. Programmable logic controllers (PLC) are cyber-physical systems that are used to control industrial appliances. PLCs translate physical inputs – in most cases current on a wire–into digital values and vice versa, to interact with the physical appliances like sensors and actuators. They can convert sensor readings into digital values, process the readings with the built-in computing unit, and forward the outputs to actuators to manipulate the physical world. Based on the available information about the system state, a PLC calculates the next actuations to steer the system towards a desired state. The program running on the PLC, called *control logic*, defines the control algorithm used to decide actuations. The target system state towards which the PLC is working can be fixed in the control logic or could be set dynamically over the network by the ICS operator.

Control logic programs can be loaded onto PLCs and run on top of a privileged software layer like a real-time operating system (RTOS). This privileged software layer contained in the PLC's firmware provides services to the control logic programs (e.g., networking, storage) and manages the programs' updates and execution. The control logic programs are executed repeatedly in fixed intervals, called *scan cycles*.

In distributed ICS, the physical process is jointly controlled by multiple PLCs. To do so, PLCs are usually connected through a computer network, allowing them to share information like sensor readings or internal states.

The PLCs in an ICS are usually managed and monitored through central management systems, called *Supervisory Control and Data Acquisition* (SCADA). Typical components of a SCADA system are *historians*, which are databases logging data from all control devices in the ICS, IT infrastructure servers that connect the ICS to other systems such as a supply chain management system, *human machine interfaces* (HMI), which allow an operator to interactively control the system, and operator workstations that provide interactive control as well as PLC reprogramming.

The control logic running on the PLCs is highly application specific and is usually programmed by the plant operator itself. In particular, while the firmware and development tools for PLCs are usually closed source, the operator has full access to the control logic of the PLCs. In order to design and setup a control system, the operator usually needs knowledge, i.e., a model of the system's physics.

Cyber-Physical Systems Modeling. ICS comprise a class of cyber-physical systems that can be modeled as *hybrid systems*, or systems whose continuous evolution (physical equations) evolve based on the discrete-state transitions (controller actuations) of the system [13].

For instance, in Figure 1 a simplified example of two PLCs controlling the mixing and filling of colors is shown. Four input colors are mixed and filled into cans. The input of each color

is controlled by PLC1, which controls the respective valves. PLC2 controls the conveyor belt, using a scale to determine when the current can is full and the next one has to be placed under the mixer. The pseudo-code and control-flow graph show the relation between the actions of PLC1 and the readings of PLC2, i.e., the operations of PLC1 determine the physical behavior observed by PLC2. In such a hybrid system, the closing of the valve is a discrete event. The time required to fill a single can, on the other hand, will increase gradually (evolve continuously).

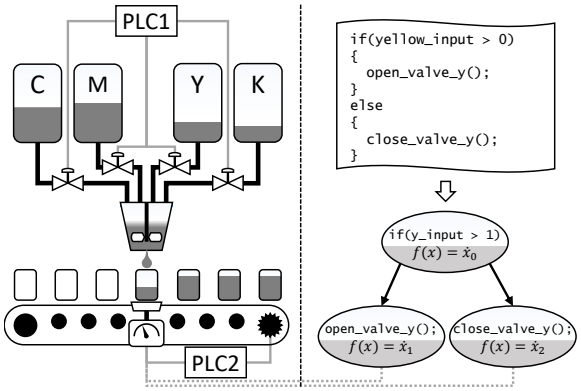


Figure 1: Simplified Industrial Control System, two PLCs control the mixing of four input colors–cyan (C), magenta (M), yellow (Y) and black (K for Key)–and filling into cans. PLC1's control actions on the valves of the individual color tanks will have an effect on the fill rate measured by PLC2.

In the context of ICS, state estimation techniques have been leveraged to model physical dynamics for particular discrete events of the system [21], [23]. However, the actuation of the associated controlling devices is typically abstracted to simplify the complexity of the model, neglecting the underlying control-flow behavior of any running programs. This simplification opens up these systems to motivated adversaries that exploit such abstractions to launch stealthy attacks [37].

III. Models and Assumptions

In this section, we present the system model and the adversary model considered in this paper.

A. System Model

We consider large, distributed industrial control systems (ICS) with a centralized monitoring system (SCADA). This is the predominant system design [50] for large scale industrial plants. The ICS consists of networked controllers (we use PLCs exemplary for all types of controllers used in ICS) that jointly control a (complex) physical process, where the actions of the individual PLCs are interdependent. In particular, actuations initiated by one PLC affect the system state which will be represented in the sensor readings of other PLCs. This means that all PLCs are indirectly connected with each other through the physical dynamics of the controlled physical system.[1]

[1]Note, the system does not need to be "fully" connected, i.e., the actuations of one PLC are not required to be observed by all other PLCs.

Each PLC is connected to its own local array of sensors and actuators.[2] These sensors and actuators are directly interfacing with the physical system, and associated discrete sampled values are accessed by the PLCs.

In addition to the indirect connection between PLCs, all components of a distributed ICS are also connected explicitly. That said, all PLCs are connected to each other and to SCADA over a computer network, e.g., Ethernet. By means of the computer network, input and output data of all PLCs are reported to the SCADA system, where data is recorded in a historian database and can be viewed by the operator.

B. Adversary Model

The adversary's goal is to cause misbehavior of the ICS while remaining undetected. The behavior of the system refers to the actions that influence the physical process controlled by the ICS. In particular, the adversary alters control commands sent to physical appliances (actuators) that can change the state of the physical process. Passive attacks that do not alter the system behavior, e.g., attacks that ex-filtrate data, are out of scope in this work.

ICS are usually monitored by a human operator. The adversary has to make sure that her manipulations do not cause suspicion on the operator's side [26], i.e., the attack needs to be stealthy. This precludes naïve attacks like denial-of-service (DoS) on devices or the network.

We assume that the adversary has complete control over the compromised PLC, i.e., she can compromise the firmware and the control logic of the PLC. The adversary can gain control over a PLC leveraging static or dynamic attack techniques. In a static attack, the adversary replaces the software (firmware or control logic) of a PLC, e.g., via a malicious software update. Dynamic attacks are based on injecting new code at run-time, manipulating the behavior of existing code by means of return-oriented programming (ROP) [47], or data-oriented programming (DOP) [34].

Similar to previous works that assumed the compromise of a single PLC [15] or single actuator/sensors [52], we assume the adversary can compromise one PLC (or a small subset of PLCs, or sensors/actuators controlled by these PLCs) in a distributed ICS.[3] Further, we assume the attacker in our adversary model has knowledge about the attacked system, so we have to assume a compromised PLC will report legitimate sensor values.

Network Attacks: For the sake of simplicity we consider network attacks out of scope. We assume a secure, i.e., integrity protected and authenticated channel between the controllers and SCADMAN.

[2]For simplicity we assume local sensors and actuators, however, remote I/O devices, i.e., networked sensors and actuators, can be modeled in our system as "PLCs without computations".

[3]We argue that this is a realistic assumption, as many attacks against PLCs require physical access, e.g., via JTAG [30], and hence do not scale to large and distributed ICS. For software attacks heterogeneous systems containing PLCs with different hardware or firmware versions, different models, or even from different vendors, minimize the attacker's capabilities of compromising all PLCs.

IV. SCADMAN DESIGN

Before we describe our SCADMAN design and framework, we discuss important challenges that we had to tackle for SCADMAN.

A. Challenges

Important limitations in ICS stem from closed source, proprietary software. Control software, firmware, and compilers are usually manufacturer specific and cannot be modified by the customer. Thus, modification of the software running *on* the PLC is not feasible as it would require the cooperation of the manufacturer. The control logic, on the other hand, is usually developed by the plant operator, i.e., the operator has full access to its source code.

Modifications of the PLC software also can lead to undesirable implications that will hinder adoption in practice. Safety and reliability are paramount in ICS. Hence, all modifications that could impact them are unlikely to be adapted. In particular, in systems that require safety certification any modifications of the PLC software would void them, i.e., solutions that rely on the modification of control-components cannot be used in highly-sensitive environments.

Finally, having a comprehensive view of the system is necessary to detect sophisticated attacks from a powerful adversary. Although a single PLC may have access to monitor the values of other sensors/actuators of the ICS, the maintenance of state estimation of the physical processes will incur significant overhead in the PLC. Even if a PLC had the memory resources for such state estimation, the computation of the state estimation might cause a violation of the real-time constraints.

B. SCADMAN Design

The goal of SCADMAN is to detect deviations from the correct behavior of a distributed ICS. The correct behavior can be violated by different types of attacks, as discussed before in Section III-B. As a result, SCADMAN must provide a general mechanism that can counter all possible attacks that result in an incorrect control behavior of the system. Control behavior includes any action taken by any of the PLCs that modifies the overall system state. We consider the control behavior as correct, if it fits the behavior intended by the system operator. For example, a PLC is supposed to close a valve when a certain threshold is reached. The attacker then forces the PLC to keep the valve open, contrary to the original programming of the PLC.

The system can also deviate from the intended state for other reasons like faults, e.g., a faulty or manipulated sensor reporting incorrect values. SCADMAN can detect these situations, allowing the operator to repair the system.

In a distributed ICS, multiple PLCs interact independently with a physical process. However, the actions of one PLC influence the overall state of the physical system. This is reflected in the sensor readings of other PLCs. We exploit this interdependency to detect the misbehavior of a compromised PLC. For example, a compromised PLC cannot stealthily open a valve because a second PLC, which is not under attacker control, would measure the change in inflow. This discrepancy between expected sensor readings and the actual system state is used by SCADMAN to detect deviations in the control behavior.

SCADMAN-MONITOR: All PLCs report their actuation commands and sensor readings to a central entity, which we call SCADMAN-MONITOR. The SCADMAN-MONITOR is a program that interacts with a simulated physical system and allows SCADMAN to calculate the expected state of the overall ICS. Note that centrally reporting and logging all operations is very common in ICS [50], e.g., for reporting to HMI components. Based on the retrieved data, the SCADMAN-MONITOR will subsequently check whether any of the PLCs have been deviating from the intended behavior, i.e., that all PLCs have been following a small set of valid control flow paths given the current system state. This check requires two components of the SCADMAN-MONITOR. (1) A consolidated control logic code of *all* PLCs, and (2) a model of the physical process allowing SCADMAN-MONITOR to determine the interdependencies of the PLCs' inputs and outputs.

SCADMAN generates the consolidated control logic which combines the control logic of *all* PLCs into a single large program that represents the entire control actions of the ICS. This code is executed on the SCADMAN-MONITOR to determine valid actions of the PLCs. Based on the current state of the overall system and the model of the system's physical state, SCADMAN dynamically derives the legitimate program executions, i.e., control-flow paths, of the PLC code in a physics-aware manner. This means, SCADMAN does not accept all possible, benign control-flow paths in the control logic's control-flow graph as valid, as is the case with CFI [1], [2], but only those that are valid at any given time in the current state of the *cyber physical system* (CPS). This approach allows SCADMAN to precisely determine the correct behavior of each PLC given the current system state.

The physical process model allows the SCADMAN-MONITOR to estimate the influence of control commands sent by PLCs on the expected sensor readings. As the adversary cannot influence the physical model of the system (the laws of physics cannot be altered), an inconsistency between actuation commands and sensor readings implies that either the PLC controlling the actuation or the PLC controlling the sensors must behave incorrectly, i.e., issue wrong actuation commands or report forged sensor readings. SCADMAN can tolerate imprecise and incomplete models. The model quality largely determines the detection precision. However, an imprecise model, noisy sensors, and other factors impacting the state estimation are handled by SCADMAN as described below and in more detail in Section V of the technical report [5].

C. SCADMAN Framework

Our SCADMAN framework leverages a compiler-based approach to automatically generate the *consolidated PLC* code and connect it to the physical *state estimator*. Both are executed in SCADMAN-MONITOR, as shown in Figure 2.

Physical State Estimation: The state estimator uses physical models of the physical processes that are controlled by the PLCs of the ICS. The state estimator simulates the evolution of the physical system. Based on the current state of the physical systems and actuation inputs to the system the state estimator determines the following state of the system.

Physical systems usually evolve continuously, however, for SCADMAN the system state at the sampling point is relevant, i.e., at the points in time when a PLC reads the system state using its sensors. Models of the physical processes in ICS are usually known by the operator of a plant. Additionally, several recent works have developed methods to extract and generate such models, which can be used with SCADMAN [21], [23].

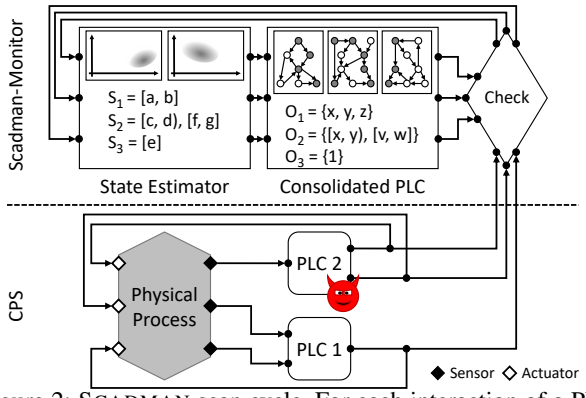


Figure 2: SCADMAN scan cycle. For each interaction of a PLC with the physical process, the SCADMAN-MONITOR performs the corresponding operations using the state estimated based on the system's past events. Afterwards, consistency between both sides (CPS and SCADMAN-MONITOR) is checked, and an alarm is raised when a deviation is detected.

SCADMAN-MONITOR Operation: SCADMAN runs in parallel with the CPS (on the top in Figure 2) and compares the sensor readings and actuation commands it receives over the network from all PLCs to a set of valid state determined by SCADMAN-MONITOR–allowing for the validation of the behavior of the CPS (i.e., distributed ICS).

SCADMAN-MONITOR works in iterations, similar to the scan cycle based operations of PLCs. During each iteration the current system state and actuation commands are fed into the state estimator of SCADMAN-MONITOR, in parallel the physical process in the CPS evolves, based on the current system state and past actuation commands. The state estimator calculates the state space into which the system should have evolved based on its history, in Figure 2 the state variables $S_1 \dots S_3$ have been predicted to lay within some given interval of possible values. The fuzziness of the state space can be, for instance, due to the imprecision of the physical model used in state estimator or due to (small) errors in the input value. However, SCADMAN can tolerate these imprecisions.

The actual physical state in the CPS is read by the PLCs's sensors and processed by them. In parallel, SCADMAN-MONITOR executes the consolidated PLC using the estimated state as input. Since the input state can be fuzzy the execution of SCADMAN-MONITOR has to account for it. By using a technique called *error-margin multi-execution* the execution is performed over the entire range of possible input values (see Section V of the technical report [5] for details). By executing–given the current system state context–*all* valid control-flow paths in the consolidated PLC SCADMAN-MONITOR determines the set of possible outputs. Again, the outputs ($O_1 \dots O_3$ in Figure 2) are represented as intervals or sets of allowed values.

When SCADMAN-MONITOR receives the sensor reading and actuation command of the PLCs from the CPS it performs the consistency check. If all PLCs were executing correctly–and all sensors and actuators operated correctly–the reported values must be a subset of the outputs determined by SCADMAN-MONITOR. Any deviation indicates an inconsistency within the behavior of the CPS and SCADMAN raises an alarm.

If the system was found to be correct the system state reported by the PLCs is accepted as the current system state and serves as input for the next iteration of SCADMAN-MONITOR. This is necessary to prevent the system state calculated by the state estimator to gradually deviate from the actual system state.[4]

V. SCADMAN IMPLEMENTATION

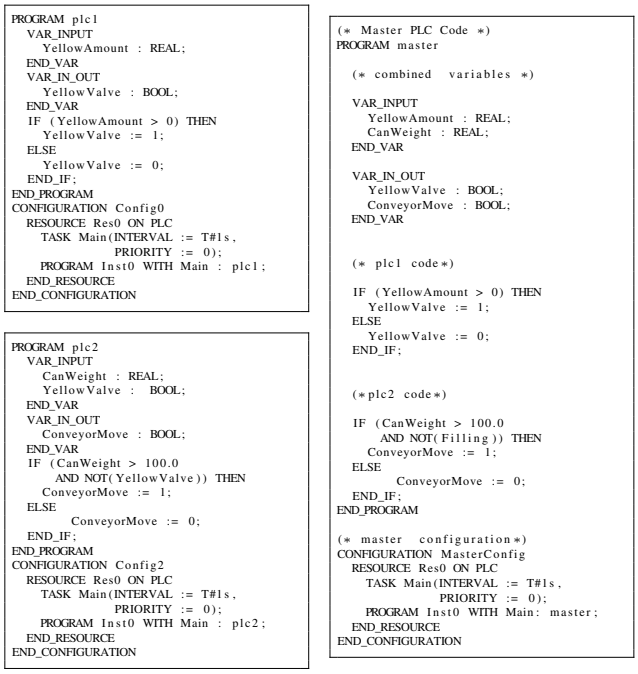
SCADMAN introduces the SCADMAN-MONITOR process, which is responsible for receiving all sensor values and actuations, and validating them using the consolidated PLC code and state estimation. We propose a generic approach to build the SCADMAN-MONITOR. We will first discuss how SCADMAN consolidates the control programs for a distributed PLC network along with the necessary assumptions for timing and functional correctness. We then demonstrate how the consolidated code will be compiled and instrumented into an executable that can receive the state of the ICS network, e.g., the state of the sensors, as input for each scan cycle and update the estimated state of the system. We also introduce a novel approach, so-called error-margin multi-execution that allows SCADMAN to account for cases where the model of the physical system deviates from the real system. Finally, we describe how these components of SCADMAN can be combined with the physical state estimation to detect any compromised components in the ICS.

Example. For the purpose of clarity, we will provide a simplified representation for a single process control for two PLCs from the system in Figure 1. Figure 3 shows two PLC programs, `plc1` and `plc2` that are consolidated by SCADMAN into a single PLC representation, `master`. PLC1 is responsible for controlling a valve, `YellowValve`, associated with the yellow color dispenser. The valve will open if an input amount is greater than 0. PLC2 is responsible for moving the conveyor belt if the current can is full and if the `YellowValve` is not open. Descriptive variable names have been used in the code. This example will be used to explain each component of the implementation.

A. PLC-Code Consolidation

The premise of generating a SCADMAN-MONITOR representation is to first merge the control program code of all the ICS PLC's into a single PLC program representation. This consolidated representation is necessary for two reasons. First, in order to monitor the distributed processes, we need access to all of the system parameters in order to successfully simulate the physical model of the overall system, i.e., we cannot simulate the physics of a process with partial sensor

[4]To detect and counter slow evolving attacks SCADMAN can be adapted to simulate the system state over multiple iterations at the cost of added impression in the state estimation (see Section IX of the technical report [5]).



```
PROGRAM plc1
  VAR_INPUT
    YellowAmount : REAL;
  END_VAR
  VAR_IN_OUT
    YellowValve : BOOL;
  END_VAR
  IF (YellowAmount > 0) THEN
    YellowValve := 1;
  ELSE
    YellowValve := 0;
  END_IF;
END_PROGRAM
CONFIGURATION Config0
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main : plc1;
  END_RESOURCE
END_CONFIGURATION
```

```
PROGRAM plc2
  VAR_INPUT
    CanWeight : REAL;
    YellowValve : BOOL;
  END_VAR
  VAR_IN_OUT
    ConveyorMove : BOOL;
  END_VAR
  IF (CanWeight > 100.0
      AND NOT(YellowValve)) THEN
    ConveyorMove := 1;
  ELSE
      ConveyorMove := 0;
  END_IF;
END_PROGRAM
CONFIGURATION Config2
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main : plc2;
  END_RESOURCE
END_CONFIGURATION
```

```
(* Master PLC Code *)
PROGRAM master

  (* combined variables *)

  VAR_INPUT
    YellowAmount : REAL;
    CanWeight : REAL;
  END_VAR

  VAR_IN_OUT
    YellowValve : BOOL;
    ConveyorMove : BOOL;
  END_VAR


  (* plc1 code*)

  IF (YellowAmount > 0) THEN
    YellowValve := 1;
  ELSE
    YellowValve := 0;
  END_IF;


  (*plc2 code*)

  IF (CanWeight > 100.0
      AND NOT(Filling)) THEN
    ConveyorMove := 1;
  ELSE
      ConveyorMove := 0;
  END_IF;
END_PROGRAM

(* master configuration*)
CONFIGURATION MasterConfig
  RESOURCE Res0 ON PLC
    TASK Main(INTERVAL := T#1s,
              PRIORITY := 0);
    PROGRAM Inst0 WITH Main: master;
  END_RESOURCE
END_CONFIGURATION
```

Figure 3: Code consolidation for two PLCs with respect to a single process of the paint mixing plant in Figure 1. The left 2 programs, `plc1` and `plc2` are merged into a `master` PLC code.

data. In theory, the consolidation would not be necessary for a subset of the PLCs that do not have any cyber-physical interdependencies. However, these dependencies are difficult to derive. As such, SCADMAN automatically generates models that incorporate these cyber-physical interdependencies as long as the distributed system conforms to the assumptions required to ensure functional and timing correctness, which are discussed at the end of this subsection. The details of our automated generation of consolidated code from ICE 61131 compliant PLC code are described in Section V of the technical report [5]. We further discuss the functional and timing correctness of the code consolidation process in the same section.

B. Compilation and Instrumentation

To compute the updated values of actuators at the end of a scan cycle, we need to execute the consolidated PLC code given the state of the sensors. We integrate the consolidated control code into the SCADMAN-MONITOR and record the actuations, performed by the control code. Figure 4 provides an overview of the implementation and shows how the consolidated code is incorporated into the SCADMAN-MONITOR. We use our extended MATIEC compiler[5] to compile the consolidated PLC code to C code. We modified the MATIEC compiler to automatically generate functions that allow easy access and modification of the internal state of the generated C code. This is used by the SCADMAN-MONITOR process to simulate access to sensor values and actuators.

[5]https://github.com/thiagoralves/OpenPLC_v2/

As a second step, we compile the C code into an intermediate representation using the LLVM [39] compiler framework. Operating on the LLVM intermediate code allows us to perform analysis and instrumentation without having to analyze structured text or C code directly. We perform instrumentation on the generated LLVM intermediate code to introduce an execution mode that draws from the ideas of symbolic execution, abstract interpretation and interval arithmetic. We call this execution mode *error-margin multi-execution*. We use this execution mode to reduce the number of false positives by introducing the notion of an error-margin to accessed sensor values. We discuss the details of this approach in the following subsection.

The final step is to produce a runnable executable. We compile the instrumented PLC code to native code and link it with our support library, providing various utility functions. The C code generated by the MATIEC compiler is intended to be linked to a userspace driver that implements hardware access. Instead we link the generated code with our framework such that all hardware accesses are intercepted and forwarded to the state estimation and attack detection. The model of a physical system may deviate from the real system due to various reasons. In Section V of the technical report [5], we describe the approach to account for error-margins by utilizing *multi-execution*.

C. Attack Detection

To detect attacks, the SCADMAN-MONITOR performs two steps, where the results from the n-th scan cycle are used to predict and verify the $n+1$-th scan cycle of the system. First the SCADMAN-MONITOR compares sensor values received in scan cycle n with the sensor values estimated based on the inputs from scan cycle $n-1$. When the received senor values are verified, i.e., fall within the set of predicted values, the SCADMAN-MONITOR uses them as input to execute the consolidated PLC code. This results in a set of acceptable actuation operations for scan cycle n. SCADMAN-MONITOR compares this set against actuation operations reported by the real PLCs. If the actuations of the real PLCs are verified correctly they serve as input to the state estimator of the physical system, which will predict the sensor values for the next scan cycle $n+1$.

Incomplete data. SCADMAN can also be used on systems that cannot provide complete data or which involve (sub)process for which no accurate state estimation is possible. This can be due to various reasons, e.g., if a sensor state depends on human interactions with the system the SCADMAN-MONITOR cannot predict the state of that sensor in a meaningful way. However, the state of other sensors and actuators of the system that are not directly influenced by such external influence can still be validated by SCADMAN.

VI. SECURITY CONSIDERATIONS

In this section we discuss how SCADMAN can detect attacks on ICS. According to our adversary model (cf. Section III) we consider a subset of PLCs to be compromised, i.e., k out of n PLCs are compromised, where $k < n$.

For distributed ICS, the adversary needs to control *all* PLCs to provide a coherent view of *all* actuation commands and *all*

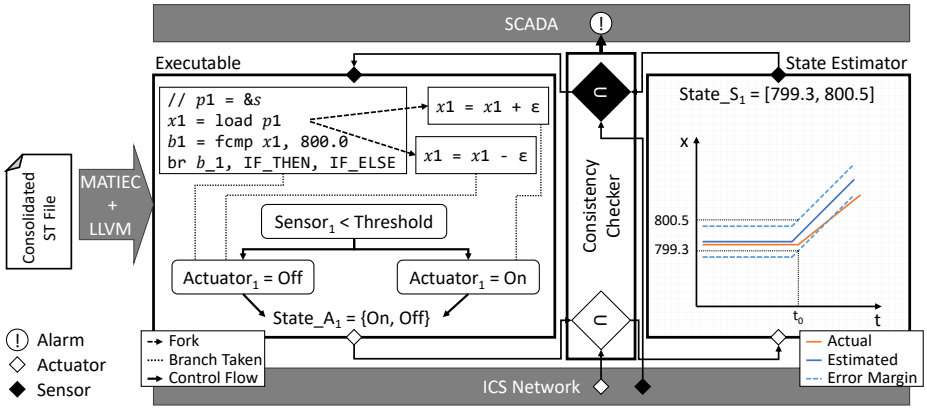


Figure 4: SCADMAN implementation overview. The consolidated PLC code is compiled in combination with MATIEC and LLVM to an executable. The ICS network feeds sensor values into SCADMAN-MONITOR to simulate the PLC scan cycle and check for consistency. The updated outputs are fed from the executable to the state estimation as well. Any deviations in the expected behavior is alerted to the SCADA monitor.

sensor readings reported to SCADMAN-MONITOR. This means the adversary has to simulate the expected behavior of the *entire* system and synchronizes the actions of all PLCs.

However, if the adversary controls only a subset of PLCs and acts stealthy, i.e., the adversary lets the compromised PLCs report sensor readings and actuation commands that meet the expectations of the operator, SCADMAN will detect the attack as the reported values will *not* match the expected values relative to the values reported by the non-compromised PLCs.

In particular, as long as one PLC that is physically interconnected with the compromised PLCs reports correct values, a discrepancy will emerge. SCADMAN will detect this discrepancy and will raise an alarm. While SCADMAN will not be able to identify which PLCs are compromised, it can still warn the operator, who can then start an in-depth investigation of the system. In the next section we evaluated SCADMAN on a large set of ICS attacks implemented for a real-world industrial control network (SWaT [43]) and show that it can detect all of these attacks.

Slow evolving attacks. SCADMAN is based on a closed-loop approach where, for each scan cycle, the system is analyzed for anomalies. The system continues when no anomaly is detected. By continuing, the current state of the system is accepted as benign and serves as the basis for estimating the system's future state. An adversary could try to exploit this scenario by *slowly* pushing the system towards a false state. On each iteration, the adversary would manipulate the system within the error margins of SCADMAN. However, ICS are usually designed to include safety measures programmed into the PLCs that prevent the system from being steered to an unsafe state. While the attack can slowly modify the system within the safety boundaries of the system without being detected, the system cannot be pushed to an unsafe state. SCADMAN would detect any deviations in the control flow path of the PLC, e.g., if an adversary pushes the system outside of the safety boundaries enforced by a safety check within the control flow of the original PLC program. The best the adversary can do is to leverage the simulation error margin used by SCADMAN to get the system slightly outside of its safety boundaries. However, the safety boundaries are chosen such that the system remains safe even in the presence of small errors, e.g., due to sensor measurement noise.

We discuss additional countermeasures in Section IX of the technical report [5].

VII. EVALUATIONS

In this section, we provide an overview of our experimental evaluation of SCADMAN. We first introduce the real-world industrial control network that we used for our evaluation. We then discuss how SCADMAN implements code consolidation for the proprietary PLCs used in our evaluation. Afterwards we describe the choice of physical state estimation equations used for our attack detection. Finally, we evaluate SCADMAN against a set of attacks that were enumerated by previous works.

A. Evaluation Environment and Dataset

The study reported here was conducted on data from a real distributed industrial control system [43] as shown below.

The network is a 6-stage water treatment plant, containing 68 sensors and actuators, where the sub-process of each stage is controlled by an individual PLC. More details are provided in the associated system paper [43].

Dataset. We evaluated SCADMAN using data generated by the industrial control network presented in [43]. The dataset includes both normal operations to evaluate the false-positive rate of SCADMAN as well as attacks to evaluate the detection performance of SCADMAN. The attack data, containing data from a total of thirty-six attacks, were generated independently of our work, modeled after Adepu et al. [8], [7] and was used in previous works to evaluate ICS security solutions [31].

The dataset contains data collected during seven days of continuous operation of the ICS network. It contains $496,800$ data points, each point representing the system state using 53 features of the system, e.g., sensor values and actuator states. The sensor data indicates the states of various plant components including tanks, valves, pumps, and meters, as well as data on chemical properties including pH, conductivity, and the Oxidation Reduction Potential (ORP).

B. SCADMAN *State Estimation*

We now describe how we evaluated each component of SCADMAN's implementation in order to generate the cyber-physical state estimator. The implementation details for how we consolidate the PLC code from an ICS network are provided in Section V of the technical report [5].

Physical state estimation. SCADMAN provides physical state estimation for sensors whose sensor and actuation dependencies are satisfied. For instance, we cannot predict the value of a water tank if we do not have access to the corresponding flow rate sensor. We provide generic physical state estimators for the water tank level sensors, the flow rate level sensors, as well as the status indicators for pumps and valves. However, models for other components of the system can be added in future work.

For water tanks, we used the same estimation and threshold values provided in prior analyses of the ICS network platform [9], [6]. SCADMAN implements the following closed-loop state estimation models:

$$TankLevel = TankLevel + (Inflow - Outflow) * F_c.$$

Where `Inflow` and `Outflow` are the inflow/outflow rates of the tank and F_c is a conversion constant for the flow rate.

For flow rates, we derived a closed-loop model that incorporates any actuators that may open/close the flow of water:

$$FlowRate = FlowRate * \prod_{n=1}^{N} Actuator_n$$

Where $Actuator_n$ represents any pump or value whose value is 0 (for *off*) or 1 (for *on*). For our analyses, we use the plant invariants that capture the state of the system at any point of time [6]. Each model is then invoked automatically when a particular variable needs to be estimated. In addition to providing generic models for these subsystems, the models for the binary values of the actuator states are automatically generated by our SCADMAN-MONITOR executable.

C. *Attack Detection*

We were able to successfully detect all attacks enumerated in the attack data set. We further evaluated SCADMAN against the record-and-replay attacks enumerated in a previous case study, where sensor values were recorded and replayed back to the HMI to spoof sensor values as was done in the Stuxnet malware [9]. For the non-optimized evaluation, SCADMAN had 0 false negatives with a very low false positive rate of 0.36% for the nominal water tank level deviation threshold in the implementation *without* multi-execution. The false positives were due to the cases mentioned in section V, where an actuator may open/close a tick too early or too late based on our estimated sensor values.

False positive pruning. All false positives were pruned by our error-margin multi-execution implementation for the nominal water tank level deviation threshold. We show the associated ROC curves of varying water tank level deviation thresholds for both the normal execution and the error-margin multi-execution in Figure 5. False positives only exist for very small threshold values, i.e., a threshold value that is less than 1mm

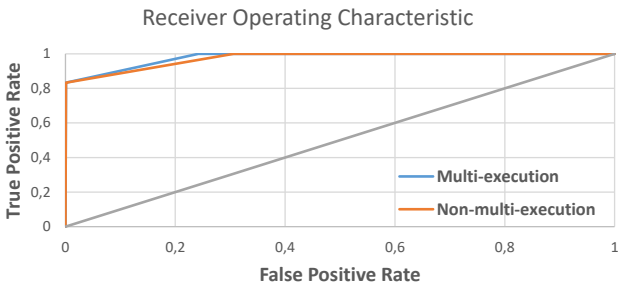


Figure 5: ROC curve for attack detection across varying water tank deviation thresholds for both single and multi-execution analysis.

for the water tank level will obviously result in some false positive rate. The nominal threshold values were based on the threshold values used for state estimation in a previous work [9]. The nominal threshold value of 5mm obtained from our ROC curve confirmed the choice in the previous work.

Performance. We performed our evaluation on a system equipped with an Intel Core i7-4710MQ Processor at $2.50\,GHz$, $16\,GB$ of RAM running Linux v4.4.0-112-generic. Running SCADMAN on the entire dataset of seven days took 30 hours for the single-threaded deviation-checking, and 51 hours for the multi-threaded error-margin multi-execution using our current prototype implementation that is *not* optimized for performance[6]. This shows that SCADMAN can "keep up" when running in parallel with the real system even on a desktop-grade computer.

SCADMAN's detection delay has not been measured, however, we do not consider it crucial since SCADMAN is a *detection* mechanism. Delays of the reactions–possibly by a human operator–will be significantly larger.

Memory. The memory usage of SCADMAN was $36\,MB$ on average with a peak memory requirement of $149\,MB$, with multi-execution turned off. This shows that SCADMAN can be used to constantly monitor the system behavior using standard server equipment.

Communication. By default, all PLCs communicate with the SCADA system to display the operational process data and to store the operational data in a historian. SCADMAN can retrieve its data from the historian causing no communication overhead in the PLC network.

Comparison to other works. Previous work [15] has evaluated their IDS for PLCs against the same attack data set. This work relies upon mutating code to generate simulated data traces of anomalous behavior. This work was only able to detect ~88 percent of the attacks, while our work was able to identify all the attacks. This is due to the fact that their algorithm for the generation of anomalous behaviors via code mutation chooses lines at random for a random choice of PLC. SCADMAN deterministically models the software state with respect to the physical state estimation. In terms

[6]For every state estimation update in our prototype, the values of the previous sample were read from the disk, updated, and written back to the disk. This file I/O accounts for almost the entire overhead and can be easily circumvented for real-time operation if the values are stored in memory.

of implementation, this work required a physical model to simulate the traces and also required access to the source code.

VIII. Related Work

The previous works on ICS security can be categorized into cyber-physical security mechanisms that are implemented within the ICS controller to enforce code integrity and monitoring solutions that abstract the control of PLCs to verify the overall cyber-physical system.

Internal CPS control security. ECFI [3] provides a control-flow integrity (CFI) solution for PLCs, where the code running on the PLC is instrumented to validate whether indirect branches follow a legitimate path in the control-flow graph (CFG). SCADMAN does not need any modifications of the code running on the PLC. In contrast, it monitors the overall behavior of the PLC reflecting its entire software (including the OS). Furthermore, SCADMAN provides context-sensitive control-flow checking, i.e., the set of allowed CFG paths is further restricted based on the current system state.

Control-Flow Attestation (C-FLAT) enables a prover device to attest the exact control-flow path of an executed program to a remote verifier [4]. However, it cannot be applied to existing systems that do not have the necessary hardware security extensions such as the ARM TrustZone. PyCRA [49] uses a physical challenge-response authentication to protect active sensing systems against cyber physical attacks. PyCRA's focus on active sensors, hence it is not applicable to passive sensors nor to actuators, both of which are also common in ICS. Orpheus [16] monitors the behavior of a program based on executed system calls and checks whether a system call is legitimate in the given context. The decision is made based on a finite-state machine (FSM) representing the program's system call behavior, i.e., system calls are only allowed to be executed in sequences for which valid transitions exist within the FSM. Orpheus requires an FSM of the monitored system, which needs to be constructed in a learning phase. SCADMAN does not require such a model of the overall system but only models for the individual subprocesses of the physical system. Also, Orpheus performs detection on the device and relies on an un-compromised OS and that physical event reports are untampered. SCADMAN does not require any modifications to the monitored devices and does not require a trusted channel to input sensors.

Zeus [33] monitors the control flow of a PLC control program by monitoring the electromagnetic emissions side channels of the PLC by a neural network model. Such a defense does not protect against data attacks and can further be circumvented via firmware modification attacks. Furthermore, Zeus cannot account for verifying other networked components as in the SCADMAN framework.

State estimation has been used within the PLCs to detect if any of the invariant properties of the system have been violated [9], [6], [11]. This enforcement resides in the application layer of a single PLC, which can be circumvented if the PLC is compromised. Furthermore, the physical invariants and their dependencies are specified manually. SCADMAN automatically enforces the checking of discrete-state transitions by analyzing the consolidated PLC code. Similarly, on-device runtime verification has been proposed for PLCs with coupled hypervisors [29]. The hypervisor resides above the firmware and relies on the integrity of the PLC control logic. Cyber-physical *control* invariants have also been used for monitoring within controllers of robotic vehicles [18]. These control invariants are generated by learning safe coefficients of the associated cyber-physical control system using a System Identification tool and a generated data set of device behavior. The associated closed-source controller binary is augmented with a generated monitor that checks the state variables against the invariants. SCADMAN is not data-driven and captures the deterministic properties of the control logic for a distributed set of sensors and actuators.

TSV [44] verifies the integrity of any program being loaded onto a PLC by lifting the associated binary to an intermediate language to symbolically execute the program and verify that it is not violating any of the provided infrastructural safety requirements. The safety requirements are enforced within the PLC by extension of the guarantees provided by TSV. Similarly, PLCVerif [20] provides a framework for checking safety properties of PLC code against finite-state automata. These solutions are offline analyses that do not provide any runtime guarantees and only verify the control logic application code. HyPLC [27] provides a sound theoretical CPS model of PLC code in the context of ICS for verifying safety properties, but was not designed with security in mind for the purpose of intrusion detection. Such modeling would be supplementary to SCADMAN in terms of understanding which code is more critical for safety and establishing safety bounds, but is orthogonal to our contributions.

External CPS control security. Previous works have proposed means of detecting stealthy attacks in the context of ICS. Urbina et al. [52] reported on limiting the impact of stealthy attacks on industrial control systems. Liu et al. [42] presented false data injection attacks against state estimation in electric power grids. This work is implemented mainly in a simulation environment, where they are considering stealthy attacks on smart meters. In SCADMAN, we also consider stealthy attacks on multiple sensors and actuators on real-time operational data. Zheng et al. [56] provided a means of detecting data integrity and false command attacks on a single PLC via path redundancy in a distributed setting. Although such a framework considers attack prevention, it can only detect if there is a discrepancy between the value a sensor is reporting versus the value a particular PLC is reporting for that sensor. It does not validate the integrity of the sensor value nor the PLC actuation with respect to the cyber-physical system state. Furthermore, such a solution requires modification of the networking and assumes other PLCs can provide redundant sensing.

Chen et al. [15], [14] proposed an approach for learning physical invariants that combines machine learning with ideas from mutation testing. Initial models are learned using support vector machines. These learned models are used for code attestation and identifying standard network attacks. Configuration based intrusion detection system have also been proposed for Advanced Metering Infrastructure [12]. The AMI behavior is modeled using event logs collected at smart meters. Event logs are modeled using Markov chains and linear temporal logic for the verification of specifications. However, such models depend on the completeness of the training data set used for the learned models. A water control system was modeled

using an autoregressive model in order to monitor the physics of the system [32]. For distributed systems with complex cyber-physical interdependencies, it is infeasible to assume all discrete states of the system will be traversed. SCADMAN automatically contains a discrete-state model of the entire ICS and depends only on the accuracy of the physical state estimation. In a similar vein, the idea of detecting attacks by monitoring physics [46], [19] of the ICS by using invariants has been applied. However, in these instances the invariants were derived manually based on domain knowledge. SCADMAN automatically derives these cyber-physical invariants and significantly reduces the probability of human error during the modeling phase of complex systems. Zhangh et al. [55] similarly generate invariants from data traces and establish temporal dependencies with causality graphs generated from static program analysis of the PLC code to detect hidden safety violations. SCADMAN provides real-time cyber-physical state estimation that also estimates the state of the PLC code.

IX. Conclusions and Summary

Industrial control systems (ICS) are ubiquitous and increasingly deployed in critical infrastructures. In fact, recent large-scale cyber attacks (e.g., Stuxnet, BlackEnergy, Duqu to name a few) exploit vulnerabilities in these systems. Building a generic defense mechanism against the various ICS attack flavors is highly challenging. However, we observe that all these attacks influence the physics of these devices. As a result, we developed SCADMAN, a system that enables control-logic aware anomaly detection for distributed cyber-physical systems. SCADMAN provides real-time monitoring for intrusion detection and sensor fault detection by maintaining a cyber-physical state estimation of the system based on a novel control code consolidation generation as well as state estimation equations of the physical processes. SCADMAN monitors the correctness of individual controllers in the system by verifying the actuation values being sent from the PLCs as well as the associated changes that propagated through the physical dynamics of the system. We evaluated SCADMAN against an enumerated set of attacks on a real water treatment testbed. Our results show that we can detect a wide range of attacks in a timely fashion with zero false positives for nominal threshold values.

Acknowledgements

This work is based upon work supported by the Department of Energy under Award Number DE-OE0000780, the National Science Foundation (NSF), the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – SFB 1119 – 236615297 (P3 and S2 within the CRC 1119 CROSSING) and as part of the project HWSec, by the German Federal Ministry of Education and Research (BMBF) and the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP, by BMBF within the project CloudProtect, and by the Intel Collaborative Research Institute for Collaborative Autonomous & Resilient Systems (ICRI-CARS).

References

[1] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow Integrity. In *Conference on Computer and Communications Security*, CCS, 2005.

[2] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti. Control-flow Integrity Principles, Implementations, and Applications. *ACM Trans. Inf. Syst. Secur.*, 13(1), Nov. 2009.

[3] A. Abbasi, T. Holz, E. Zambon, and S. Etalle. ECFI: Asynchronous Control Flow Integrity for Programmable Logic Controllers. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, ACSAC, 2017.

[4] T. Abera, N. Asokan, L. Davi, J.-E. Ekberg, T. Nyman, A. Paverd, A.-R. Sadeghi, and G. Tsudik. C-FLAT: Control-Flow Attestation for Embedded Systems Software. In *Conference on Computer and Communications Security*, CCS, 2016.

[5] S. Adepu, F. Brasser, L. Garcia, M. Rodler, L. Davi, A.-R. Sadeghi, and S. Zonouz. Control behavior integrity for distributed cyber-physical systems. *arXiv preprint arXiv:1812.08310*, 2018.

[6] S. Adepu and A. Mathur. Distributed detection of single-stage multipoint cyber attacks in a water treatment plant. In *Proceedings of the 11th ACM Asia Conference on Computer and Communications Security*, pages 449–460, New York, NY, May 2016. ACM.

[7] S. Adepu and A. Mathur. Generalized attacker and attack models for Cyber-Physical Systems. In *Proceedings of the 40th Annual International Computers, Software & Applications Conference, Atlanta, USA*, pages 283–292, Washington, D.C., USA, June 2016. IEEE.

[8] S. Adepu and A. Mathur. An investigation into the response of a water treatment system to cyber attacks. In *Proceedings of the 17th IEEE High Assurance Systems Engineering Symposium, Orlando*, pages 141–148, January 2016.

[9] S. Adepu and A. Mathur. Using Process Invariants to Detect Cyber Attacks on a Water Treatment System. In *IFIP International Information Security and Privacy Conference*, 2016.

[10] S. Adepu, G. Mishra, and A. Mathur. Access control in water distribution networks: A case study. In *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*, July 2017.

[11] S. Adepu, S. Shrivastava, and A. Mathur. Argus: An orthogonal defense framework to protect public infrastructure against cyber-physical attacks. *IEEE Internet Computing*, 20(5):38–45, 2016.

[12] M. Q. Ali and E. Al-Shaer. Configuration-based ids for advanced metering infrastructure. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 451–462, New York, NY, USA, 2013. ACM.

[13] P. J. Antsaklis, J. A. Stiver, and M. Lemmon. Hybrid system modeling and autonomous control systems. In *Hybrid systems*, pages 366–392. Springer, 1993.

[14] Y. Chen, C. M. Poskitt, and J. Sun. Towards Learning and Verifying Invariants of Cyber-Physical Systems by Code Mutation. In *International Symposium on Formal Methods (FM 2016)*, LNCS, 2016.

[15] Y. Chen, C. M. Poskitt, and J. Sun. Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system. In *IEEE Symposium on Security and Privacy (S&P 2018)*. IEEE Computer Society, 2018. To appear.

[16] L. Cheng, K. Tian, and D. D. Yao. Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks. 2017.

[17] E. Chien, L. OMurchu, and N. Falliere. W32.Duqu - The precursor to the next Stuxnet. Technical report, Symantic Security Response, nov 2011.

[18] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Xinyan. Detecting attacks against robotic vehicles: A control invariant approach. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 801–816. ACM, 2018.

[19] A. Choudhari, H. Ramaprasad, T. Paul, J. W. Kimball, M. Zawodniok, B. McMillin, and S. Chellappan. Stability of a cyber-physical smart grid system using cooperating invariants. In *2013 IEEE 37th Annual Computer Software and Applications Conference*, pages 760–769, 2013.

[20] D. Darvas, E. Blanco Vinuela, and B. Fernández Adiego. PLCverif: A tool to verify PLC programs based on model checking techniques. 2015.

[21] K. R. Davis, C. M. Davis, S. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, P. W. Sauer, et al. A cyber-physical modeling and assessment framework for power grid infrastructures. 2015.

[22] S. Etigowni, M. Cintuglu, M. Kazerooni, S. Hossain, P. Sun, K. Davis,

O. Mohammed, and S. Zonouz. Cyber-Air-Gapped Detection of Controller Attacks through Physical Interdependencies.

[23] S. Etigowni, D. J. Tian, G. Hernandez, S. Zonouz, and K. Butler. CPAC: securing critical infrastructure with cyber-physical access control. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 139–152. ACM, 2016.

[24] F-Secure Labs. BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks, 2016.

[25] N. Falliere, L. O. Murchu, and E. Chien. W32.Stuxnet Dossier. Technical report, Symantic Security Response, Oct. 2010.

[26] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA*, pages 26–28, 2017.

[27] L. Garcia, S. Mitsch, and A. Platzer. Hyplc: Hybrid programmable logic controller program translation for verification. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, pages 47–56. ACM, 2019.

[28] L. Garcia, H. Senyondo, S. McLaughlin, and S. Zonouz. Covert channel communication through physical interdependencies in cyber-physical infrastructures. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 952–957. IEEE, 2014.

[29] L. Garcia, S. Zonouz, D. Wei, and L. P. de Aguiar. Detecting plc control corruption via on-device runtime verification. In *Resilience Week (RWS), 2016*, pages 67–72. IEEE, 2016.

[30] L. A. Garcia, F. Brasser, M. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz. Hey, my malware knows physics! attacking plcs with physical model aware rootkit. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, NDSS, 2017.

[31] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*, pages 140–145, Jan 2017.

[32] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel. Through the eye of the PLC: Semantic security monitoring for industrial processes. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 126–135. ACM, 2014.

[33] Y. Han, S. Etigowni, H. Liu, S. Zonouz, and A. Petropulu. Watch Me, but Don't Touch Me! Contactless Control Flow Monitoring via Electromagnetic Emanations. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1095–1108. ACM, 2017.

[34] H. Hu, S. Shinde, S. Adrian, Z. L. Chua, P. Saxena, and Z. Liang. Data-Oriented Programming: On the Expressiveness of Non-control Data Attacks. In *IEEE Symposium on Security and Privacy*, S&P, 2016.

[35] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops 2017, New Orleans, LA, USA, November 18-21*, pages 1058–1065, 2017.

[36] K. N. Junejo and D. K. Yau. Data driven physical modelling for intrusion detection in cyber physical systems. In *Proceedings of the Singapore Cyber-Security Conference (SG-CRC)*, pages 43–57, 2016.

[37] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur. Model-based security analysis of a water treatment system. In *Proceedings of the 2Nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, SEsCPS '16, 2016.

[38] P. Kong, Y. Li, X. Chen, J. Sun, M. Sun, and J. Wang. Towards concolic testing for hybrid systems. In *FM 2016: Formal Methods*, pages 460–478, 2016.

[39] C. Lattner and V. Adve. LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation. In *Proceedings of the 2004 International Symposium on Code Generation and Optimization (CGO'04)*, Palo Alto, California, Mar 2004.

[40] Q. Lin, S. Adepu, S. Verwer, and A. Mathur. Tabor: A graphical model-based approach for anomaly detection in industrial control systems. 2018.

[41] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[42] Y. Liu, P. Ning, and M. K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):13, 2011.

[43] A. P. Mathur and N. O. Tippenhauer. Swat: a water treatment testbed for research and training on ics security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pages 31–36. IEEE, 2016.

[44] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel. A trusted safety verifier for process controller code. In *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2014.

[45] K. Pal, S. Adepu, and J. Goh. Effectiveness of association rules mining for invariants generation in cyber-physical systems. In *18th IEEE International Symposium on High Assurance Systems Engineering, HASE 2017, Singapore, January 12-14, 2017*, pages 124–127, 2017.

[46] T. Paul, J. W. Kimball, M. Zawodniok, T. P. Roth, B. McMillin, and S. Chellappan. Unified invariants for cyber-physical switched system stability. *IEEE Transactions on Smart Grid*, 5(1):112–120, 2014.

[47] R. Roemer, E. Buchanan, H. Shacham, and S. Savage. Return-Oriented Programming: Systems, Languages, and Applications. *ACM Trans. Info. & System Security*, 15(1), Mar. 2012.

[48] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell. A quantitative evaluation of the target selection of havex ics malware plugin.

[49] Y. Shoukry, P. Martin, Y. Yona, S. N. Diggavi, and M. B. Srivastava. Pycra: Physical challenge-response authentication for active sensors under spoofing attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1004–1015. ACM, 2015.

[50] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. *NIST special publication*, 800(82):16–16, 2011.

[51] M. A. Umer, A. Mathur, K. N. Junejo, and S. Adepu. Integrating design and data centric approaches to generate invariants for distributed attack detection. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy, Dallas, TX, USA, November 3*, pages 131–136, 2017.

[52] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1092–1105. ACM, 2016.

[53] J. Wang, X. Chen, J. Sun, and S. Qin. Improving probability estimation through active probabilistic model learning. In *Formal Methods and Software Engineering*, pages 379–395, 2017.

[54] J. Wang, J. Sun, Q. Yuan, and J. Pang. Should we learn probabilistic models for model checking? a new approach and an empirical study. In *International Conference on Fundamental Approaches to Software Engineering*, pages 3–21. Springer, 2017.

[55] M. Zhang, J. Moyne, Z. M. Mao, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, et al. Towards automated safety vetting of plc code in real-world plants. In *Towards Automated Safety Vetting of PLC Code in Real-World Plants*, page 0. IEEE.

[56] Z. Zheng and A. Reddy. Towards improving data validity of cyber-physical systems through path redundancy. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pages 91–102. ACM, 2017.