Securing Industrial Cyber-Physical Systems: A Run-time Multi-layer Monitoring

Anonymous Author(s)

Abstract—Industrial Cyber-Physical Systems (ICPSs) are widely deployed in monitoring and control of the nation's critical industrial processes such as water distribution networks and power grids. ICPSs are the tight integration of cyber (software) and physical entities connected via communication networks. Communication networks are typically realised via wireless channels to reduce the cost of wires and installation. However, they are also inherently unreliable, easy to disrupt and subvert, which makes them a potential target for cyber attacks. The failure of communication can cause data loss or delays, which can compromise system functionality and have catastrophic consequences due to the strict real-time requirements of ICPSs. Current run-time security monitors protect ICPSs either at communication level (through network intrusion monitors) or application level (through threat detection monitors). Such monitors are layer-specific and thus fail to detect advanced threats arising from the multi-layer disruption. In this paper, we present a multi-layer run-time security monitor that can detect discrepancies caused by interdependent application and communication layer attacks and prevent their propagation into the system's control loops. We demonstrate the effectiveness of the approach via an example of the ICPS used for control and monitoring of a water distribution network.

Index Terms—Industrial Cyber-Physical System (ICPS), security, communication and application layer attacks, run-time monitoring.

1 Introduction

Large scale infrastructure, such as water distribution networks or petrochemical systems, are getting smarter using networks of sensor and actuator devices. Such systems have evolved into Industrial Cyber-Physical Systems (ICPSs). These ICPSs enable fine-grained control of infrastructure assets, making them more efficient, reduce wastage, and increase resilience to failure. A wireless approach to ICPSs further reduces the cost of the wires and maintenance efforts and enables the scale as the system grows [1].

Despite their increasing potential in new generation ICPSs, Wireless Sensor and Actuator Networks (WSANs) face significant challenges that distinguish them from wired industrial networks. The WSAN communication relies on radio networks that are easy to disrupt and subvert, causing them to lose or delay data which will then compromise functionality of ICPS applications. If wireless communication is disrupted, the control schemes will not have access to temporally relevant data, and performance and safety guarantees can not be met.

Existing approaches have attempted to protect data communication (i.e., communication layer) and computation (i.e., application layer) of the ICPS separately. Regarding the communication layer, extensive research efforts have been put into hardening security of WSANs with the use of cryptographic primitives and key sharing schemes [2], [3], [4]. Additionally, the communication protection has been provided via run-time monitoring (through network intrusion monitors) [5], [6], [7]. Since these approaches protect only data communication, they fail to detect advanced threats that involve semantics of the data contents or data-based computations, e.g., application software.

Regarding the application layer, several run-time threat detection monitors have been developed to harden the security of computation. Such monitors establish good and bad reference behaviours. They can raise alarms whenever the observed behaviour matches the bad reference behaviour or deviates from the good reference behaviour. To operate, these monitors either employ data-driven approaches [8], [9], [10] that are known to suffer from a high rate of false alarms or model-based approaches [11], [12], [13], [14] that are limited to protecting application computations only.

Both, the communication layer approaches and the application layer approaches to security, fail to acknowledge the interdependencies between the communication network and control application and the effects that communication has on the operation of a control application. For example, a malicious activity at the communication layer may increase resource contention in WSANs, leading to long communication delays or failures which will not be necessarily accounted for or checked by the control application. Additionally, the operation of the control application would be similarly affected if the attack's intensity or the extent of the attack's impact changes.

In this paper, we present a *Multi-Layer Run-time Security Monitor (ML-RSM)* that can detect discrepancies caused by interdependent communication and application-layer attacks and prevent their propagation into the system's control loops. Our design methodology for developing reliable and secure ICPSs combines the strengths of the application behaviour monitor (that uses an application specification) and communication constraints model (that includes non-deterministic complexity and delay). As a representative ICPS communication protocol, we use WirelessHART [15] industry standard that has been adopted worldwide for industrial processes management and control [1]. Our threat model includes attacks specific to both, the communication layer and application layer. These attacks aim to disturb or delay the flow of data to the control system application. We demonstrate the effectiveness of the approach via an example of the ICPS used for control and monitoring of a

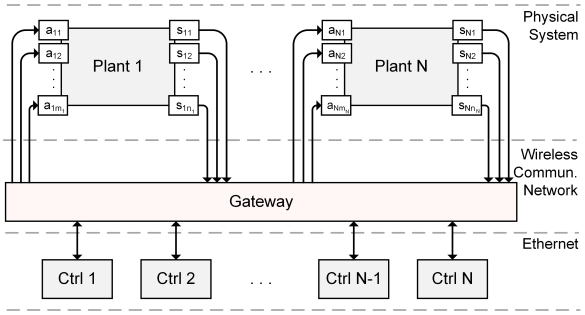

Fig. 1. The architecture of ICPS.

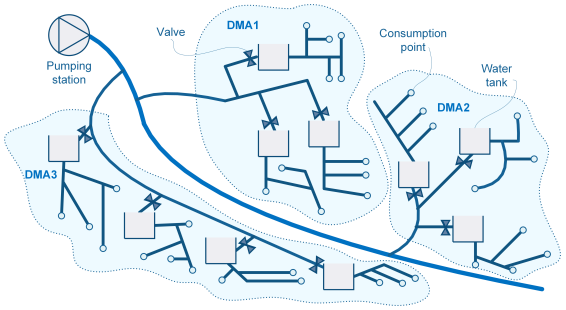

Fig. 2. The water distribution network.

water distribution network.

The rest of the paper is structured as follows: Section 2 describes the ICPS architecture and the application example used throughout the paper. Section 3 presents the WirelessHART communication model, the communication constraints and main security threats. Section 4 presents a runtime multi-layer security monitor design approach. Section 5 presents the results of vulnerability analysis. Section 6 lists potential future work directions and concludes the paper.

2 ICPS Architecture and the Application Example

In this section, we present the architecture of an ICPS. We also give an example of a typical ICPS control application that will be used throughout the paper.

2.1 System Architecture

We consider an ICPS with an architecture such as the one shown in Fig. 1. The ICPS consists of N subsystems each labelled i, where $i \in \{1, \ldots, N\}$. Each subsystem consists of a single plant, described using a linear time-invariant model in Eq. 1, and a linear input-to-state feedback controller in Eq. 2, both labelled with the same i as their subsystem:

$$\dot{\xi}_i(t) = A_i\xi_i(t) + B_i v_i(t), \quad (1)$$

$$v_i(t) = K_i\hat{\xi}_i(t). \quad (2)$$

The plant i is instrumented with sensors $\{s_{i1}, s_{i2}, \ldots, s_{in_i}\}$ to measure its physical processes where n_i corresponds to the number of sensors of the subsystem i. The measurements are taken periodically regardless of the state of the physical process and stored in the state vector $\xi_i(t) \in \mathrm{R}^{n_i}$. The periodic sampling sequence can be expressed as

$$\tau_{k_i} := \{t_{k_i} | t_{k_i} := k_i h_i, k_i \in \mathrm{N}\} \quad (3)$$

where $h_i > 0$ is the time between the samples and k_i is the number of the sample.

The values of the state vector, $\xi_i(t)$, are sent to the controller using a wireless communication network. The controller uses the values to calculate the state error vector $\hat{\xi}_i(t) = \xi_i(t) - \xi_{ref_i}$, where ξ_{ref_i} is the reference water level. According to Eq. 2, the controller uses a predesigned control gain $K_i \in \mathrm{R}^{n_i \times n_i}$ for which the controlled system is stable to provide the control input vector, $v_i(t) \in \mathrm{R}^{m_i}$. $v_i(t)$ defines the actions that are sent to actuators $\{a_{i1}, a_{i2}, \ldots, a_{im_i}\}$ using a wireless communication network. m_i corresponds to the number of actuators of the subsystem i. The goal of the controller is to maintain the values in vector $\xi_i(t)$ at a reference point (or a set point). $A_i \in \mathrm{R}^{n_i \times n_i}$ and $B_i \in \mathrm{R}^{n_i \times m_i}$ are the matrices appropriate to the application.

As mentioned previously, the information exchange between sensors and actuators and the controller is enabled by a wireless communication network. All N subsystems in Fig. 1, are coupled through the shared wireless communication network and use a single gateway. In this paper, we will use the term gateway and controller interchangeably, and we can treat them as one device. This is a safe assumption because the gateway and the controllers communicate via wired communication which is instantaneous and reliable. The communication layer will be described in more details in Sec. 3.

2.2 Application Example

In this section, we present the physical model of a Water Distribution Network (WDN) consisting of three District Meter Areas (DMAs) (see Fig. 2) [16]. Each DMA has three or four water tanks and supplies an average of 10-30 customer connections via a set of pressurised pipes, pumping stations, and valves.

For simplicity, we present a model of a single DMA with three tanks (DMA1). The DMA1 is modelled using the linear time-invariant model in Eq. 1 with the matrices: $A_i = diag\{-8.367 - 6.276 - 5.020\} \times 10^{-4}$, $B_i = \begin{bmatrix} 0.1068 & -0.0371 & -0.0371 \\ -0.0279 & 0.0801 & -0.0279 \\ -0.0223 & -0.0223 & 0.0641 \end{bmatrix}$. In the state vector $\xi_i(t) \in \mathrm{R}_i^n$ each state is measured by a sensor and represents the difference between the current water level and set point reference water level for a tank. The set-point in our system is ξ_{ref} is equal to 3 meters and the number of sensors, n_i, is equal to 3. The system is sampled at the rate of once every $h = 2$ seconds.

The control input vector $v_i(t) \in \mathrm{R}_i^m$ represents the degree to which the in-valves are open to fill each tank. To calculate the actions that are sent to actuators we use a linear input-to-state feedback controller in Eq. 2 with $K_i = \begin{bmatrix} -0.3024 & -0.0089 & -0.0238 \\ 0.0228 & -0.3034 & -0.0073 \\ 0.0357 & 0.0215 & -0.3024 \end{bmatrix}$. $K_i \in R^{n \times n}$ is a designed control gain for which closed-loop system is stable

and the number of actuators, m_i, is equal to 3. We assume a constant water demand for each of the three tanks.

3 ICPS Communication Layer

In this section, we present the ICPS communication layer. First, we derive the main requirements of the ICPS communication system. Then, we give an example of a representative protocol that satisfies the previously defined constraints. Finally, for the chosen communication protocol we present the timing constraints and the treat model.

3.1 ICPS Communication Requirements

ICPSs require reliable real-time communication. They have strict requirements on the maximum allowed transmission delays and message losss that they can tolerate and still maintain system stability. We refer to ICPSs as to the delay-intolerant, loss-intolerant applications.

However, the reality is that the transmission delays and message loss of wireless communication are inherently non-deterministic and unreliable. The most common way to constrain this unreliability is to use a class of reservation-based wireless protocols. For such protocols, sensor and actuator nodes are allocated, for example, a fixed time slot to transmit their sampled data and receive their control actions, respectively [17]. As a result, communication is collision-free and deterministic.

In this paper, we use WirelessHART [15] as a representative example of reservation-based wireless protocols. We chose WirelessHART as it is an industry standard that is adopted widely for control systems. However, the same approach could be applied for any other reservation-based protocol. Next, we present the network model that uses the WirelessHART protocol.

3.2 WirelessHART Network Model

A WirelessHART single-hop network consists of a set of end-devices denoted by $j = 1, 2, \ldots, M$ where M is the total number of end-devices, and a gateway. An end-device is either a sensor node or an actuator node (i.e., $M = n_i + m_i$ for the subsystem i), and is instrumented on the physical process (or plant) such as the one described in Section 2.2. The end-devices are communicating wirelessly to the gateway that is further connected to the controller. The communication between the gateway and the controller is wired, which is instantaneous and reliable.

WirelessHART operates in 2.4GHz Industrial Scientific and Medical (ISM) radio frequency band. It adopts the IEEE 802.15.4 PHY layer, and its MAC layer needs to conform to the IEEE 802.15.4 standard. The WirelessHART MAC layer uses two main features: 1) Time-diversity that is achieved using TDMA (Time Division Multiple Access), where each end-device is allocated a 10ms long time slot to communicate with the gateway. 10ms long time slots allow exactly one transmission and its associated acknowledgement. 2) Spectrum diversity that is achieved by using all 16 channels defined in IEEE 802.15.4. An end-device can change the channel to avoid potential interference (from coexisting wireless systems or jammers).

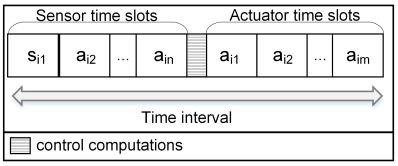

Fig. 3. The WirelessHART frame structure.

For simplicity, we assume that the subsystem i uses a single frequency channel to realise its uplink (sensor-to-controller) and downlink (controller-to-actuator) communication. Each sensor is allocated a 10ms uplink time slot to send its measurements. The number of uplink time slots is equal to the number of sensors in the subsystem, n_i. Similarly, each actuator is allocated a 10ms downlink time slot for receiving the corresponding control action. The number of downlink time slots is equal to the number of actuators in the subsystem, m_i. The sensor and actuator time slots are separated by the guard slot within which the controller performs the computation of control inputs.

The sensor, actuator, and guard time slots are organised within frames. The frame structure is depicted in Fig 3. The frames are occurring periodically where the length of the period corresponds to the sampling rate of sensor nodes, h_i, of the ICPS. For the presented network model, we give the timing constraints in the following section.

3.3 WirelessHART Timing Constraints

To specify the timing constraints of the WirelessHART network, we first need to define the end-to-end delay. The end-to-end delay is defined as the total delay from when the data is sampled by sensors and transmitted to the controller until when the corresponding control action is received at actuators. The total delay can be expressed as the sum of following terms:

$$t_{delay} = t_{uplink} + t_{comp} + t_{downlink}. \quad (4)$$

t_{uplink} is the time that a sensor node takes to access its time slot and send its measurements. t_{uplink} cannot be larger than $n_i \times 10$ ms (i.e., the worst-case scenario when the sensor is allocated the last available time slot). t_{comp} is the time that the controller takes to process the received measurements and compute the control input. We do not consider this delay in the analysis as it is negligible when compared to the other delay terms. $t_{downlink}$ is the time an actuator node takes to receive the control action from the controller. $t_{downlink}$ cannot be larger than $m_i \times 10$ ms (i.e., the worst-case scenario when the actuator is allocated the last available time slot).

To enable real-time communication, a set of timing constraints need to be satisfied in the following order:

1) The controller has to receive all sensor measurements before it performs computation, i.e. $t_{uplink} \leq n_i \times 10$ ms, $\forall s_{in_i}, \forall t$.
2) The actuators need to receive their control action before the frame is over, i.e. $t_{downlink} \leq m_i \times 10$ ms, $\forall a_{im_i}, \forall t$.
3) The total delay has to be smaller or equal to the sampling time, i.e. $t_{delay} \leq h_i, \forall t$.

If any of the constraints is not satisfied, the operation of the ICPS can be affected. Additional failures can happen also due to malicious disruption and subverting [18], [19]. The threat model of WirelessHART and its security features are given next.

3.4 WirelessHART Threat Model

WirelessHART implements several mechanisms to ensure data confidentiality, authenticity and integrity in both hop-by-hop and end-to-end transmissions [19]. However, the use of wireless interface makes WirelessHART vulnerable against a number of threats. An attacker can disrupt the flow of data by undertaking a number of the malicious activities [18], We categorise these into two main categories:

- *Computational threats* where an attacker can modify the instructions or internal-data values of the application computations. This falsifying of information can lead to a de-synchronisation which is critical for a correct operation of time-based protocols such as WirelessHART. The detection of such threats is discussed in Sec. 4.4.
- *Communication threats* where an attacker can cause a communication failure (e.g., by using a jammer) or injects a false data. In both cases, due to the lack of up-to-date readings, the controller will not be able to produce the correct action and the system operation will be severely affected. For this class of threats we consider two cases, a simple communication failure where some data is dropped randomly (discussed in Sec. 4.4) and a more advanced data injection which might go undetected by the monitor and would require a vulnerability analysis to be done (discussed in Sec. 5).

The presented categories of the attacks considered a different levels of sophistication (i.e. more and less advanced attacker actions).

4 Multi-Layer Run-Time Security Monitor

In this section we present the Multi-Layer Run-time Security Monitor (ML-RSM) that combines strengths of an application behaviour model and communication constraints model. First, we describe the ML-RSM structure. Then, we demonstrate how it can be used with our application example in Sec. 2.2. Finally, we present the ML-RSM prototype implementation details and we discuss the detection of the computational and communication threats presented in Sec. 3.4.

4.1 Run-Time Multi-layer Security Monitor Design

Based on ARMET [14], we introduce our Multi-Layer Run-time Security Monitor (ML-RSM). The ML-RSM structure is given in Fig. 4.

ML-RSM allows monitoring of both, the application layer and communication-layer of an ICPS. In simple terms, ML-RSM checks if the observed behaviour is consistent with the expected behaviour of the ICPS. If any inconsistency is detected, the ML-RSM raises the alarm. The alarm could be then further used by a larger system (e.g., AWDRAT [20]) to

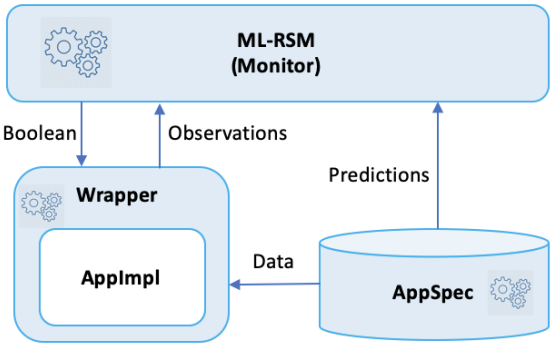

Fig. 4. Multi-layer run-time security monitor (ML-RSM)

enable a fail-safe operation of the ICPS. A more detailed explanation of the ML-RSM operating principle and the challenge of modelling and observing the application and communication layer behaviours together are given next.

To perform the monitoring, ML-RSM requires two inputs: the application specification (Apppec) and the application implementation (AppImpl) [21]. Since AppSpec (predicted behaviour) and AppImpl (observed behaviour) correspond to a different level of abstraction of an ICPS, Wrapper is used to encapsulate AppImpl and provide the monitor with the representation that is comparable to AppSpec. ML-RMS then runs AppSpec and AppImpl in parallel checking their consistency.

The challenge in modelling and observing the application layer and communication layer behaviours together lies in the understanding of their inter-dependencies. For example, discrepancies at the communication layer will propagate into the ICPS's control operations at the application layer. Therefore, to grasp these inter-dependencies our AppSpec models the two types of the behaviour together:

- *Computational behaviour* (or application layer behaviour) by establishing the definitions of good and bad behaviour for the ICPS (for its cyber and physical parts). The good behaviour is obtained by decomposing the ICPS into a number of submodules that are encoded using pre- and post-conditions. Submodules are connected via data-flow and control-flow links. The bad behaviour is obtained by defining known and suspected attack plans.
- *Communication behaviour* (or communication layer behaviour) by establishing the timing constraints of the underlying communication scheme as discussed in Sec. 3.3. These constraints are modelled as an invariant.

The AppSpec language allows modelling cyber and physical components of ICPS as first-class models together with functional (computational behaviour) and non-functional (communication behaviour) characteristics. The formalism of the language combines monadic second-order logic and event calculus that enables describing system behaviour at various levels of abstraction, with a higher degree of modularity. Based on the formalism, the specification can be directly compiled into executable code, and is thus, inherently efficient to ensure real-time requirements of ICPS.

```
class DMASpec {
  private static final double VALVE_CLOSED = 0.0;
  private static final double VALVE_OPENED = 360.0;
  private double[] water_levels = {0.0,0.0,0.0};
  private double[] valve_degrees = {VALVE_CLOSED,
     VALVE_CLOSED, VALVE_CLOSED};
  ...

  public double[] sense_levels(double[] readings){
    for(int i=0; i<3; i++){
        readings[i] != null /\ readings[i] >= 0.0 ->
        water_levels[i] = readings[i];
     }
     return water_levels;
  }

  public double[] set_valve_degrees(double[] degrees){
    for(int i=0; i<3; i++){
        valve_degrees[i] = degrees[i];
    }
    return valve_degrees;
  }

  public void timestamp(){
    for(int i=0; i<3; i++){
        delta_level = valve_degrees[i] *
     VALVE_FLOW_PER_DEGREE - EMPTY_RATE
                /\ 0 <= water_levels[i] + delta_level <=
     MAX_LEVEL
       ->
            water_levels[i] = min(MAX_LEVEL,
                                  max(0, water_levels[i]+
     delta_level))
    }
  }
}
```

Listing 1. The specification of the DMA

Semantically, the formalism of the language translates into a finite automaton that recognises only the words that satisfy the specification [22].

4.2 Application Example Specification

To demonstrate an AppSpec example, we use the model of a single DMA presented in Sec. 2.2. In this model, a single linear input-to-state controller is used to maintain the water levels of three water tanks at the desired height. Our AppSpec for the chosen example includes the following:

- A *physical model* that specifies the physical characteristics and dynamics of the DMA,
- A *cyber model* that specifies the computations performed by the controller, and
- A *communication model* that specifies timing constraints at the controller in terms of delays (these are consistent with real-time requirements of an ICPS) as invariant.

The physical model is given in Listing 1, while the cyber model and the communication model are sketched in Listing 2. For readability, we have presented our models in a Java-like language.

Listing 1 presents the operation principle of the DMA system that involves the following three operations:

- Obtaining the water levels based on the measured sensor values,
- Setting degrees to which valves should open and
- Adjusting the water levels based on the flow of water (the valves openings).

Each of these operations has a corresponding sub-model and implementation module.

```
public enum Action { PROCESS, WAIT, NOTHING }

class ControllerSpec {
  DMASpec dma = ...;
  ...

  public double[] compute_errors(double[]
     sensed_water_levels){
    double[] errors = new double[3];
    for(int i=0; i<3; i++){
        errors[i] = sensed_water_levels[i]-SET_POINT;
    }
    return errors;
  }

  public double[] compute_control_inputs(double[] errors){
    double[] inputs = new double[3];
    for(int i=0; i<3; i++){
        valve_degrees[i] = errors[i] * GAIN;
        valve_degrees[i] = min(DMASpec.VALVE_FULLY_CLOSED,
                               max(DMASpec.VALVE_FULLY_OPEN
     , valve_degrees[i]));
    }
    return inputs;
  }

    public void process(){
        long start = System.currentTimeMillis();
        while(System.currentTimeMillis()<start+
     SIMULATION_DURATION){
            double[] sensed_water_levels = dma.
     sense_levels();
            long now = System.currentTimeMillis();
            Action, Time act, t =
            { a, t | (a = PROCESS ->
                     (t >= now+T_UPLINK ->
                        errors = compute_errors(
     sensed_water_levels) /\
                        new_valve_degrees =
     new_valve_degrees(errors)
                              -> (now = System.
     currentTimeMillis() /\
                                 t >= now+T_DOWNLINK) ->
                                 dma.setvalve_degrees(
     new_valve_degrees)))
                /\ (a = WAIT -> now = System.
     currentTimeMillis() ->
                    (now <= t < now+T_UPLINK) \/ (now <=
     t < now+T_DOWNLINK) -> skip)
            };
            ...
        }
    }
}
```

Listing 2. The specification of the controller

Listing 2 presents the flow of operations performed by the controller. These are:

1) Use the sensor values to measure the state vector of the system, $\xi_i(t) = [\xi_{i1} \quad \xi_{i2} \quad \xi_{i3}]^T$.
2) Compute the difference (i.e., the error) between the measured system states and the set point of the controller to get the state error vector, $\hat{\xi}_i(t) = [\hat{\xi}_{i1} \quad \hat{\xi}_{i2} \quad \hat{\xi}_{i3}]^T$.
3) Multiply the error vector by a pre-defined gain K_i to get the control input vector, $v_i(t) = [v_{i1} \quad v_{i2} \quad v_{i3}]^T$.
4) Send the control input vector, $v_i(t)$, to the actuators.

The flow of operations performed by the controller is affected by the timing constraints of the communication model. The timing constraints are:

1) Step 1 has to be executed at the sampling time, $t_{k_i} = kh_i$, $k \in \mathrm{N}$ only.
2) Steps 2 and 3 require all three sensor readings to be successfully received via an uplink wireless

TABLE 1
Monitor overhead performance

Execution mode	Per 1×10^{-1} cycle	
	CPU time	Real time
End-to-End (no invariant)	2.90×10^{-5}	3.09×10^{-5}
End-to-End (with invariant)	3.63×10^{-4}	4.47×10^{-4}
Full ML-RSM	8.17×10^{-3}	8.32×10^{-3}

communication channel within the time t_{uplink} (see Sec. 3.3).

3) Step 4 requires the controller to get the access to a downlink wireless channel. It has to be executed within the time $t_{downlink}$.
4) All four steps have to be executed within the time t_{delay} which has to be less than the sampling time, $h_i, \forall t$.

4.3 The ML-RSM Prototype Implementation

Our current prototype implementation is simulated on a Mac Pro with a 2.6 GHz 6-Core Intel Core i7 processor. The controller algorithm runs as an application in Programmable Logic Controller (PLC), while the ML-RSM runs as a middleware for PLC. The controller application can be developed based on any other industry standard (i.e., IEC 61131-3 [23]) language using any or many among ladder diagram, sequential function charts (SFC), structured text (ST) and function block diagram (FBD). ML-RSM can run on top of the operating system, e.g., Real-Time Operating Systems (RTOS) [24].

To demonstrate the ML-RSM performance, we simulated the experiment for 200 seconds using different monitoring execution modes. The results are presented in Table 1. We can observe that the monitor runs efficiently with negligible overhead even in the case of fine-grained monitoring (i.e., CPU time of 8.17 x 10^{-3} seconds for each cycle of the ML-RSM algorithm). We also show that the monitor can be tuned to achieve better performance. An example is bypassing intermediate checks, such as the invariant, and instead monitoring the correctness of the data and control flows only.

4.4 Computational and Communication Threats Detection

The current prototype of ML-RSM aims to detect the computational and communication threats presented in Sec. 3.4. This is done by checking if the observed behaviour is consistent with the expected behaviour of the ICPS.

More specifically, in the case of a computational attack where an attacker modifies the internal sensor values, the deviation will be detected as the values would be significantly different than what has been predicted by the App-Spec. Similarly, if an attacker manipulates the operations (i.e., command) of the controller implementation (AppImpl) or the information flow, the inconsistency will be detected again.

On the other hand, in the case of a communication attack that may falsify the information or cause a transmission failure when some of the sensor readings were dropped, the monitor will successfully detect the attacks because there is no significant gap between the model of the system and its corresponding example implementation. Both of the attacks were launched artificially through modifications during a simulation. For instance, in case of falsifying information, when the timing information was tampered, it was immediately detected by the monitor being inconsistent with predicted time value whose model is given in Lines 30–40 of Listing 2. Similarly, in case of transmission failure, when some of the sensor readings were dropped, the monitor immediately alarmed detecting violation of the condition that the reading is null (the model is given in Line 10 of Listing 1).

As demonstrated above, our ML-RSM can detect various computational and communication attacks, however, there is still a possibility of eluding the monitor through stealthy attacks [25]. These are characterised as more advanced false data injection and tampering attacks and we analyse them in the next section.

5 Vulnerability Analysis

In this section we present the results of a vulnerability analysis to detect stealthy attacks that cannot be detected by ML-RSM. We demonstrate our approach based on a variation of the application example in Sec. 2.2.

5.1 Stealthy Attacks

This category of the attacks considers more advanced adversaries that can tamper with sensor readings or control action signals over an extended period of time. The tampering is done subtly (to approximate the expected behaviour of the system) so that the values are not detected as anomalous by anomaly detection mechanisms. However, the underlying physical system will be compromised, and over time its degraded performance can lead to a complete denial of service. We refer to these as to *stealthy attacks* [25].

Most of the existing approaches to handle stealthy attacks are limited as they either harden the security of the network through switching an underlying topology [26], [27] or by measuring the magnitudes of behaviour residuals [28]. Such approaches will fail to detect stealthy attacks in the cases when an attacker has enough knowledge of the system topology to manipulate the readings such that they do not violate the desired threshold.

In contrast to the contemporary approaches, we handle the stealthy attacks based on a method that identifies the vulnerabilities in the system design [14], [29] that are the basis of these attacks. The method is iterative; the design will be refined until no vulnerabilities are identified, or the identified vulnerabilities are covered by the monitor. The method requires a model of the system as a state function. It is then analysed to determine a set of input values that establish a stealthy attack. The method is described in more details next.

5.2 Stealthy Attack Detection Method

Consider an ICPS that implements a control loop for some process that is modelled as a function $P(x_t)$. $P(x_t)$ accepts a set of input variables (x_t) at every time instant t. In the implementation of the system, the set of variables x_t is measured through sensor readings (y_t), that is later input to

the controller and used to estimate the state of the system as well as the necessary actions (a_t). Furthermore, we consider that the system is monitored by a monitor $mon(x, y)$ that takes the measurements y at time t and evaluates them for their acceptance due to potential sensor failures.

During secure operations, the monitor $mon(x, y)$ accepts all measurements y and uses them to estimate the state as modelled by $P(x, y)$. A set of measurements y' constitutes a successful stealthy attack if

- The monitor $mon(x, y')$ accepts the measurements y' and
- The measurements y' are tampered/injected values of the actual measurements y such that $y' \neq y$.

To detect such an attack we ask a question to find a set of measurements y' that correctly estimates the system state $P(x, y)$ (i.e., within error threshold) and also the monitor $mon(x, y)$ accepts them. To realise the detection of such attack(s), we require to model:

- The state of the system with a real function $f()$,
- The above question as $\exists y' : P(x, y) \wedge mon(x, y) \wedge y' \neq y$.

The above models are encoded in an SMT solver for reals, namely, dReal [30] that implements δ-precision decision procedures. These procedures can find values with given error δ. If the solver answers "yes" then it also provides one set of y' that constitutes a stealthy attack. Once the set is identified, the system state model $f()$ can be refined by introducing constraints that eliminate the attack constituted by the y'. Each refinement reduces the search space of the model until either all such attacks have been identified or we collect identified vulnerable values to be handled at run-time by the monitor ML-RSM. We use the presented detection approach to detect a stealthy attack on our application example which is given next.

5.3 Stealthy Attack to the Application Example

We demonstrate our approach based on a variation of the application example in Sec. 2.2. We consider a DMA that has 3 valves to distribute water to three tanks. Each water tank has a sensor to measure the water level. We consider that the outgoing flow of water from the valves is r_1, r_2 and r_3. These are monitored by respective sensors that give the valve open-degree and represented by real numbers, r_1, r_2 and $r_3 \in \{0.0, 1.0\}$. The horizontal cross-section area of the tanks is 1. These standard values lead to a property that the volume of water in each tank has the same value as the water level in it.

The system operates in a discrete time and at the beginning of each time period the state of the valves may change. The overall state of ICPS including three tanks and DMA, which is monitored by the monitor $mon(x, z)$, is expressed using the following function:

$$H_1 + H_2 + H_3 = r_1 + r_2 + r_3, \tag{5}$$

where $H_1 = H_1(t+1) - H_1(t)$, $H_2 = H_2(t+1) - H_2(t)$ and $H_3 = H_3(t+1) - H_3(t)$ are the water levels of the three tanks at time $t+1$, and r_1, r_2 and r_3 are the rates of water at time $t+1$ as measured by the sensors. In principle, the state function expresses that sum of the water level differences in the tanks at $t+1$ is equal to the rate of water that flows in the tanks at time $t+1$.

```
(set-logic QF_NRA)
(declare-fun h1t () Int)
(declare-fun h1t1 () Int)
(declare-fun h2t () Int)
(declare-fun h2t1 () Int)
(declare-fun h3t () Int)
(declare-fun h3t1 () Int)
(declare-fun a () Int)
(declare-fun r1in () Real)
(declare-fun r2in () Real)
(declare-fun r3in () Real)
(assert (and (and (= h1t 2) (= h1t1 3)) (= a 1)))
(assert (and (= h2t 2) (= h2t1 3)))
(assert (and (= h3t 2) (= h3t1 3)))
(assert (and (<= 0 r1in) (<= r1in 1)))
(assert (and (<= 0 r2in) (<= r2in 1)))
(assert (and (<= 0 r3in) (<= r3in 1)))
(assert (= (+ (- h1t1 h1t) (+ (- h2t1 h2t) (- h3t1 h3t)))
     (+ (/ r1in (^ a 2)) (+ (/ r2in (^ a 2)) (/ r3in (^ a 2)
    )))))
(check-sat)
(exit)
```

Listing 3. Application example model

To demonstrate the stealthy attack, consider the scenario where at the time t the system state has been configured as

$$H_1(t) = H_2(t) = H_3(t) = 2, \tag{6}$$

$$r_1(t+1) = r_2(t+1) = r_3(t+1) = 0.5. \tag{7}$$

Then, at the time $t+1$, the states should be $H_1(t+1) = H_2(t+1) = H_3(t+1) = 2.5$ satisfying Eq. 5.

In the case when the sensors provide correct values for the rates and heights as above, then the monitor $mon(x, y)$ will accept all values as they satisfy Eq. 5. However, an attacker can launch a stealthy attack by injecting false values in two ways, the detected-way or the evaded way.

In *detected-way*, the attacker injects values that do not satisfy Eq. 5 and thus are detected by the monitor. For instance, the injected values for $H_1(t+1) = H_2(t+1) = H_3(t+1) = 3$ are detected because $(3-2)+(3-2)+(3-2) \neq 0.5+0.5+0.5$.

In *evaded-way*, the attacker provides values that are false but satisfy Eq. 5 and thus are undetected by the monitor. For instance, the injected values for $H_1(t+1) = H_2(t+1) = H_3(t+1) = 3$ and $r_1 = r_2 = r_3 = 1$ satisfy $(3-2)+(3-2)+(3-2) = 1+1+1$ but are clearly not the legitimate system state at $t+1$ as per configuration in Eq. 6 and Eq. 7.

The previous example demonstrates that the system is vulnerable to stealthy attack when the sensor values for H and r are compromised. However, our method is able to detect such an attack using SMT solver, providing the state function as an input and asking if there is a solution to this equation with parameter values $H_1(t+1), H_2(t+1), H_3(t+1)$ and $r_1(t+1), r_2(t+1), r_3(t+1)$ in the defined ranges, which is different from the real action. The model of our described stealthy attack example is shown in Listing 3. The successful detection of the attack to the system is presented in Listing 4.

Based on the detection, these values can be monitored by ML-RSM or the design can be constrained to reduce the attack surface.

6 Conclusions and Future Work

We have introduced a rigorous multi-layer run-time security monitor that protects ICPS against advanced attacks that

```
a : [ ENTIRE ] = [1, 1]
h1t : [ ENTIRE ] = [2, 2]
h1t1 : [ ENTIRE ] = [3, 3]
h2t : [ ENTIRE ] = [2, 2]
h2t1 : [ ENTIRE ] = [3, 3]
h3t : [ ENTIRE ] = [2, 2]
h3t1 : [ ENTIRE ] = [3, 3]
r1in : [ ENTIRE ] = [1, 1]
r2in : [ ENTIRE ] = [1, 1]
r3in : [ ENTIRE ] = [1, 1]
delta-sat with delta = 0.00100000000000000
```

Listing 4. Application example stealthy attack detection

arise from the inter-dependence of application-layer and communication-layer vulnerabilities. The monitor detects attacks at run-time by comparing the model behaviour with the observed behaviour of the ICPS execution. The feasibility of the approach has been demonstrated through its application to a small-scale water distribution system. Our results show the protection of the ICPS not only against the computational attacks but also the stealthy attacks.

In our future work, we will extend our approach to include novel control strategies that use aperiodic communication patterns. The aperiodic strategies are based on events occurring in the underlying, monitored physical process. Events are triggered at the sensors asynchronously only when the sample measurement of the process indicates that a change will negatively affect the stability or performance of the system. This makes it difficult to model and monitor. We will also consider larger-scale industrial systems that consist, for example, of tens of DMAs that are distributed but all sharing common communication resources.

REFERENCES

[1] C. Lu, A. Saifullah, B. Li *et al.*, "Real-time wireless sensor-actuator networks for industrial cyber-physical systems," *Proc. of the IEEE*, vol. 104, no. 5, pp. 1013–1024, 2015.

[2] A. Liu and P. Ning, "Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks," in *2008 Int. Conf. on Inf. Processing in Sensor Netw. (IPSN)*. IEEE, 2008, pp. 245–256.

[3] J. Louw, G. Niezen, T. Ramotsoela, and A. M. Abu-Mahfouz, "A key distribution scheme using elliptic curve cryptography in wireless sensor networks," in *2016 IEEE 14th Int. Conf. on Industrial Informatics (INDIN)*. IEEE, 2016, pp. 1166–1170.

[4] S. Athmani, A. Bilami, and D. E. Boubiche, "Edak: An efficient dynamic authentication and key management mechanism for heterogeneous wsns," *Future Gen. Comp. Sys.*, vol. 92, pp. 789–799, 2019.

[5] R. Mitchell and R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," *IEEE Trans. on Dependable and Secure Computing*, vol. 12, no. 1, pp. 16–30, 2014.

[6] R. Mitchell and I.-R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Computing Survey*, vol. 46, no. 4, pp. 55:1–55:29, Mar. 2014.

[7] I. Tomić and J. A. McCann, "A survey of potential security issues in existing wireless sensor network protocols," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 1910–1923, 2017.

[8] Z. Niu, S. Shi, J. Sun, and X. He, "A survey of outlier detection methodologies and their applications," in *Int. Conf. on Artificial Intelligence and Comput. Intelligence*. Springer, 2011, pp. 380–387.

[9] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, p. 217–228, 2005.

[10] R. Mitchell and R. Chen, "A survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surveys (CSUR)*, vol. 46, no. 4, pp. 1–29, 2014.

[11] S. Adepu and A. Mathur, "Using Process Invariants to Detect Cyber Attacks on a Water Treatment System," in *31st IFIP International Information Security and Privacy Conference (SEC)*, ser. ICT Systems Security and Privacy Protection, J.-H. Hoepman and S. Katzenbeisser, Eds., vol. AICT-471, Ghent, Belgium, May 2016, pp. 91–104.

[12] C. M. Ahmed, C. Murguia, and J. Ruths, "Model-based attack detection scheme for smart water distribution networks," in *Proc. of the 2017 ACM on Asia Conf. on Comp. and Commun. Security*, ser. ASIA CCS. New York, NY, USA: ACM, 2017, pp. 101–113.

[13] S. Adepu and A. Mathur, "Distributed detection of single-stage multipoint cyber attacks in a water treatment plant," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, AsiaCCS*, C. Xiaofeng, W. XiaoFeng, and H. Xinyi, Eds. ACM, 2016, pp. 449–460.

[14] M. T. Khan, D. Serpanos, and H. Shrobe, "Armet: Behavior-based secure and resilient industrial control systems," *Proceedings of the IEEE*, vol. 106, no. 1, pp. 129–143, Jan 2018.

[15] J. Song, S. Han, A. Mok *et al.*, "WirelessHART: Applying wireless technology in real-time industrial process control," *2008 IEEE Real-Time and Embedded Tech. and Appl. Symposium*, pp. 377–386, 2008.

[16] I. Tomic, M. Breza, and J. A. McCann, "Jamming-resilient control and communication framework for cyber physical systems," 2019.

[17] P. Suriyachai, U. Roedig, and A. Scott, "A survey of mac protocols for mission-critical applications in wireless sensor networks," *IEEE Commun. Surveys & Tutorials*, vol. 14, no. 2, pp. 240–264, 2011.

[18] S. Raza, A. Slabbert, T. Voigt, and K. Landernäs, "Security considerations for the wirelesshart protocol," in *2009 IEEE Conference on Emerging Technologies & Factory Automation*. IEEE, 2009, pp. 1–8.

[19] L. Bayou, D. Espes, N. Cuppens-Boulahia, and F. Cuppens, "Security analysis of wirelesshart communication scheme," in *International Symposium on Foundations and Practice of Security*. Springer, 2016, pp. 223–238.

[20] H. Shrobe, R. Laddaga, B. Balzer *et al.*, "AWDRAT: A Cognitive Middleware System for Information Survivability," in *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ser. IAAI'06. AAAI Press, 2006, pp. 1836–1843.

[21] M. T. Khan, D. Serpanos, and H. Shrobe, "A rigorous and efficient run-time security monitor for real-time critical embedded system applications," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, Dec 2016, pp. 100–105.

[22] B. Courcelle and J. Engelfriet, *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. USA: Cambridge University Press, 2012.

[23] R. Ramanathan, "The iec 61131-3 programming languages features for industrial control systems," in *2014 World Automation Congress (WAC)*, Aug 2014, pp. 598–603.

[24] Yanbing Li, M. Potkonjak, and W. Wolf, "Real-time operating systems for embedded computing," in *Proc. Int. Conf. on Computer Design VLSI in Computers and Processors*, 1997, pp. 388–392.

[25] C. Kwon, W. Liu, and I. Hwang, "Analysis and design of stealthy cyber attacks on unmanned aerial systems," *Journal of Aerospace Information Systems*, vol. 11, no. 8, pp. 525–539, 2014.

[26] M. A. Rahman, E. Al-Shaer, and M. A. Rahman, "A formal model for verifying stealthy attacks on state estimation in power grids," in *IEEE Fourth International Conference on Smart Grid Communications, SmartGridComm 2013, Vancouver*, October 2013, pp. 414–419.

[27] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Trans. on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sept 2012.

[28] A. Teixeira, I. Sha'mes, H. Sandberg, and K. H. Johansson, "Revealing stealthy attacks in control systems," in *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct 2012, pp. 1806–1813.

[29] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Trans. of Information Systems Security*, vol. 14, no. 1, pp. 13:1–13:33, Jun. 2011.

[30] S. Gao, S. Kong, and E. M. Clarke, "dreal: An smt solver for nonlinear theories over the reals," in *Proceedings of the 24th International Conference on Automated Deduction*, ser. CADE'13. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 208–214.