A Virtual Environment for Industrial Control Systems: A Nonlinear Use-Case in Attack Detection, Identification, and Response

Andrés Felipe Murillo
Universidad de los Andes
Bogotá, Colombia
af.murillo225@uniandes.edu.co

Luis Francisco Cómbita
Universidad de los Andes
Bogotá, Colombia
ca.luis10@uniandes.edu.co

Andrea Calderón González
Universidad Nacional de Colombia
Bogotá, Colombia
anmcalderongo@unal.edu.co

Sandra Rueda
Universidad de los Andes
Bogotá, Colombia
sarueda@uniandes.edu.co

Alvaro A. Cardenas
University of Texas at Dallas
Richardson, Texas, USA
alvaro.cardenas@utdallas.edu

Nicanor Quijano
Universidad de los Andes
Bogotá, Colombia
nquijano@uniandes.edu.co

ABSTRACT

The integration of modern information technologies with industrial control systems has created an enormous interest in the security of industrial control, however, given the cost, variety, and industry practices, it is hard for researchers to test and deploy security solutions in real-world systems. Industrial control testbeds can be used as tools to test security solutions before they are deployed, and in this paper we extend our previous work to develop open-source virtual industrial control testbeds where computing and networking components are emulated and virtualized, and the physical system is simulated through differential equations. In particular, we implement a nonlinear control system emulating a three-water tank with the associated sensors, PLCs, and actuators that communicate through an emulated network. In addition, we design unknown input observers (UIO) to not only detect that an attack is occurring, but also to identify the source of the malicious false data injections and mitigate its impact. Our system is available through Github to the academic community.

KEYWORDS

Virtual Environment Testbeds, Industrial Control Systems, Network Security, Network Function Virtualization

ACM Reference Format:
Andrés Felipe Murillo, Luis Francisco Cómbita, Andrea Calderón González, Sandra Rueda, Alvaro A. Cardenas, and Nicanor Quijano. 2018. A Virtual Environment for Industrial Control Systems: A Nonlinear Use-Case in Attack Detection, Identification, and Response. In *Proceedings of Industrial Control System Security Workshop (ICSS 2018).* ACM, New York, NY, USA, Article 4, 8 pages. https://doi.org/10.1145/3295453.3295457

ICSS 2018, December 2018, San Juan, PR, USA

ACM ISBN 978-1-4503-6220-7/18/12…$15.00
https://doi.org/10.1145/3295453.3295457

1 INTRODUCTION

Developing high-fidelity emulation environments for industrial control systems is an essential tool for helping us understand the risks of attacks, to evaluate new defenses, and to deploy honeypots so we can understand better the attackers of these systems. A challenge for any virtual testbed, is to create an environment that behaves as close as possible to a real industrial control network as well as having a high-fidelity evolution of the physical system under control.

There is a lot of interest for designing virtual testbeds for industrial control systems [1, 9, 15, 16]. Some testbeds simulate the physical process and add real hardware to the system (hardware-in-the-loop testbeds), and some are complete simulations (simulating the process and the equipment). Nevertheless, to be able to use these testbeds to design *network* security products, or to allow attackers to connect to them and interact with them through network penetration tools, we need to provide an emulated network. In this paper we focus on the use of a virtualization environment to emulate the networks of an Industrial Control system (ICS) controlling a nonlinear physical plant and also in implementing security mechanisms to detect a false-data injection attack and localize where this attack is coming from.

Our virtualization environment represents networks in a realistic way since the virtual nodes have network stacks that represent the corresponding stacks of real devices in an ICS. By considering this stack, our co-simulation environment has higher fidelity of the behavior of a real system than purely simulation-based testbeds.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 presents our model of Network Control Systems, attacks, and defense mechanisms to mitigate such attacks. Section 4 explains the environment we deployed for tests. Section 5 shows the results of the experiments. Section 6 presents conclusions and future work.

2 RELATED WORK

Virtualization and cloud computing technologies have been used to create virtual testbeds for ICSs [1, 9, 16]. For example, the CPSTCS testbed [9] uses computer clusters, networking devices, and PLCs at its low layer, and virtualization at its upper layers to share physical

resources among a set of tenants using the testbed. The motivation behind this work is the need to create virtual environments that are flexible, scalable, and enable the testing of security mechanisms in IP-compatible networks. The authors divide their testbed into three layers. The bottom layer groups physical resources like physical switches, routers, hosts and controllers. The next layer, the virtual layer, has virtual switches and virtual hosts, mathematical models of controlled process, software simulators, APPs, and Scripts. Finally, the application layer dynamically assembles resources into a variety of service nodes through cloud-based functions. However, most authors do not show details about the mathematical models on their simulation tools neither they present experiments to evaluate prototype or performance. One difference with our work, is that we aim to build a virtual environment that can simulate and emulate with fidelity both, the network, and the physical system under control.

Alves et al. [1] propose a high-fidelity framework to virtualize the main components of a SCADA system, including the physical system, cyber-physical links, distributed control systems, and the network. To validate fidelity, they compare the behavior of their virtual representation and a real SCADA system, a virtualized gas pipeline. The testbed has a PLC that maintains the pressure between low and high setpoints. The virtual system has a Simulink/Matlab model representing the gas pipeline and they use the OpenPLC platform to implement the control logic of a PLC. Results of both systems are analyzed with similar time-responses. In addition, the authors run a DoS attack against both systems finding a great difference between delays (12 sec. in the physical vs. 0.54 ms in the virtual testbed) because the virtual testbed runs on more powerful hardware. Our work differs in the implementation of the network system. We use Mininet to emulate a network with higher fidelity, because each of our nodes implements the complete communication stack that a host or switch would have in a network. In addition, our plant model runs on Python libraries that enables the use of different mechanisms to simulate the physical process and to evaluate the impact of communication in the response of these systems.

The authors in [16] designed a middleware architecture to improve CPS resilience by coordinating and adjusting device interactions. Their prototype uses Sendim, an extension of the SDNSim testbed [15], that includes Cyber-physical (CPS) elements. These extensions however, run mathematical models that estimate the value of the real systems but do not include computing and networking stacks of the devices that are being emulated thus reducing results fidelity.

Authors in [10] present a Network Industrial Control System testbed based on emulab. The tesbted uses tightly and loosely-coupled code in the cyber layer (TCC and LCC, respectively) to represent the interaction between PLC and physical models. The physical plant behavior is simulated in Matlab, and communicates with PLC nodes using RPC/TCP, enabling the PLC to read every value generated by the Matlab code.

The testbed discussed in [12] aims to represent in a realistic manner the industrial physical equipment used in ICS. The testbed has real industrial equipment connected through a network. Two examples of man-in-the-middle attack are presented as proof concept of the testbed. The use of real equipment provides a high degree of fidelity of the ICS components, but result in a costly and hard-to-scale environment.

The Lancaster's ICS testbed [13] has four zones: safety, manufacturing, demilitarized, and enterprise. Each zone uses industrial and commercial physical equipment to represent a real industrial control system and traditional TCP/IP to communicate between zones. Although the testbed offers a wide variety of ICS equipment, network protocols, and processes to perform experiments expanding and maintaining such testbed might be cost-prohibitive and therefore a virtual environment may be cost-effective.

The work in [18] presents an electrical substation honeynet, based on virtualization technologies to achieve a high fidelity representation of a large substation. The authors use Mininet [17] to emulate the substation network and each Mininet node represents an IED. To minimize the logic complexity of the mininet nodes, each node implements a traffic mirroring service that redirects the messages to a separate machine running SoftGrid [14]. Also, another machine runs PowerWorld simulate the electrical system. Our work follows a similar approach.

NIST developed an ICS testbed to create three different scenarios [4] with physical equipment. The first scenario emulates the Tennessee Eastman Process using Simulink to represent the physical plant. The second scenario, emulates a cooperative robotic assembly for smart manufacturing using EtherCAT, a real-time industrial protocol, to communicate. The last scenario represents a SCADA, and a wide area network running a network emulator, in addition to real equipment. Network emulation tools provide traffic shaping functions to represent different communication links, available bandwidth and delays.

Table 1 summarizes related work. There is wide variety of design choices for each testbed as they have different goals. Testbeds focused on security and high fidelity of behaviors tend to rely on physical equipments to represent the involved entities [4, 12, 13]. Nevertheless, most of these testbeds present a trade-off between fidelity, enabled by using expensive real devices, and network complexity and scalability, enabled by using virtualized environments. The testbeds that focus on network security and the impact of network behavior tend to use virtual or emulation environments to represent the network [9, 10, 16, 18]. The complexity of represented networks varies with each testbed, but those using either Mininet or Emulab have the most complex networks. The majority of these systems use simulation to represent the physical plant behavior, with some exceptions which use physical equipment [4, 9, 13].

To contribute with the development of high-fidelity virtualized environments for security of CPS systems, we present a virtual environment to emulate a nonlinear plant, considering the behavior of the communication control network. Our goal is to evaluate the impact of message exchange and processing delays between nodes present in the system. The code is available as open source at Github[1].

In addition to developing a testbed we study in this paper techniques for attack detection, identification, and response. While several works have studied attack detection [11], attack *identification* (establishing specifically which device is the source of the malicious data) and attack *response* have received less attention.

[1] Available online in: https://github.com/Cyphysecurity/ICS-SDN

Table 1: Summary of related work on testbeds for Industrial Control System Security Experimentation

Proposal	Industrial Plant Representation	Industrial Equipment Representation	Network Representation
CPSTCS Testbed[9]	Simulation (Matlab, Modelica, Ptolemy)	Physical Equipment	Emulation (Open vSwitch)
Pipeline Virtual Testbed[1]	Simulation (Simulink)	Emulation (Open PLC)	Physical
SD-CPS[16]	Simulation (Sendim)	Simulation (Sendim)	Emulation (SDN Network)
Emulab ICS Testbed[10]	Simulation (Matlab)	Simulation (PLC Code)	Emulation (Emulab Network)
Siemens Testbed[12]	No Physical Plant	Physical Equipment	Physical Equipment
Lancaster ICS Testbed[13]	Physical Actuators and Sensors	Physical Equipment	Physical Equipment
Electrical Grid Honeynet[18]	Simulation (PowerWorld)	Emulation (Mininet node - SoftGrid)	Emulation (Mininet Network)
NIST ICS Testbed[4]	Simulation(Simulink) Physical Plants	Physical Equipment	Physical Equipment

This paper continues our previous work [7] on attack detection, identification, and isolation by running a virtualized environment that emulates the complete cyber-physical system, with detection, identification, isolation and mitigation of bias injection attacks on a three-tank system, validating our previous results [8].

3 NETWORKED CONTROL SYSTEMS - FAULT TOLERANT CONTROL

This section presents an overview of control systems, describes sensor integrity attacks, and introduces the concept of unknown input observers (UIOs) to identify specific sensors sending incorrect vales.

Networked Control Systems (NCS) are feedback control systems with their components linked via a real-time network. The main goal of feedback control systems is to meet requirements regarding process outputs. The controller achieves this by reading information from sensors and using that to compute, and send to actuators, control actions that reduce the error between a desired reference setpoint, and the current measurement of an output variable. These actions control external disturbances and mitigate the effect of the imperfections of the plant model.

3.1 Controlled System

Two common control objectives are a zero-steady state error and fast a response. Zero steady-state error means that the defined setpoint and the output value are equal after a transient period. Fast response to changes in the setpoint is usually obtained with a design based on pole placement. Figure 1 shows a typical block diagram of this kind of networked control system.

Many industrial plants are nonlinear. Frequently, it is convenient to use linear approximations of plant models to design controllers and fault identification mechanisms. Such linear approximation can be obtained if the nonlinear model is linearized around the operation point. When the controller is interconnected through a network to the physical process, it is necessary to have a discrete-time model. Thus, we need to choose a sampling time to generate a discrete-time linear time invariant model for a plant:

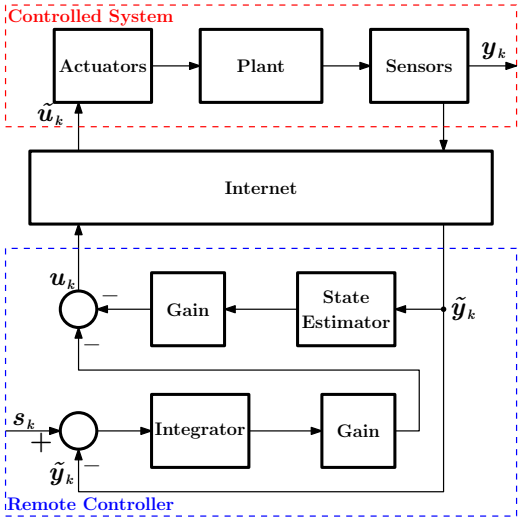


Figure 1: Networked control system with state feedback and reference tracking. The objective of the system is to maintain system output y_k, at the desired setpoint s_k.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k, \end{aligned} \tag{1}$$

where $k \in \mathbb{Z}^+$ represents the discrete time instant, $x_k \in \mathbb{R}^n$ represents the state of the system, $u_k \in \mathbb{R}^m$ represents the control input vector, and $y_k \in \mathbb{R}^p$ represents the measurement output vector. The system matrices representing the physical invariants of the system are $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, and $C \in \mathbb{R}^{p\times n}$.

3.2 Networked Controller

A network controller must satisfy some requirements: tracking of reference inputs, closed loop stability, disturbance rejection, and decoupling from other input interference in the case of multiple input/multiple output systems. A common strategy to meet these specifications is having a feedback state with one integral action. This type of control is usually referred to as PI control.

The discrete-time integrator is given by the following equation,

$$z_{k+1} = z_k + T_s(s_k - \tilde{y}_k), \tag{2}$$

where z_k is the output vector of the integrator, $s_k \in \mathbb{R}^m$ represents the set-point or reference input vector, $\tilde{y}_k \in \mathbb{R}^q$ represents the controlled output vector, and T_s represents the sampling time of the discrete-time system.

The nominal control law of a PI control system v_k is a state feedback given by

$$u_k = -\begin{bmatrix} K_1 & K_2 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ z_k \end{bmatrix}, \tag{3}$$

where K_1 and K_2 are constant gains and $\hat{x}_k$ is the estimation of the state variables. The gains K_1 and K_2 are computed to stabilize the closed loop control system and achieve the required performance of the whole system.

The dynamics of PI closed-loop control systems is given by

$$\begin{aligned} \begin{bmatrix} \hat{x}_{k+1} \\ z_{k+1} \end{bmatrix} &= \begin{bmatrix} A - BK_1 & -BK_2 \\ -T_sC & I \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ z_k \end{bmatrix} + \begin{bmatrix} 0 \\ T_sI \end{bmatrix} s_k \\ y_k &= \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} \hat{x}_k \\ z_k \end{bmatrix}, \end{aligned} \tag{4}$$

where I is an identity matrix with size equals to the number of controlled outputs.

The standard way to perform state estimation to obtain $\hat{x}$ in non-stochastic linear systems is through the use of a full order Luenberger observer:

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + Bu_k + L(y_k^a - \hat{y}_k) \\ \hat{y}_k &= C\hat{x}_k, \end{aligned} \tag{5}$$

where y_k^a is the sensor value received from the network, and where L is the Luenberger estimator gain.

3.3 Integrity Attacks

The objective of traditional security mechanisms is to guarantee integrity and authenticity of messages. Nevertheless, if an attacker breaks this first line of defense, it can send messages with tampered sensor readings. Integrity sensor attacks occur when any of the system sensors cannot maintain data integrity i.e. sensor measurements are not accurate [5].

A system output when a sensor is attacked can be modeled as

$$y_k^a = \begin{bmatrix} C & 0 \end{bmatrix} \begin{bmatrix} x_k \\ z_k \end{bmatrix} + F_s v_k, \tag{6}$$

where F_s represents the relation between the injected signal and the modified measurement. We only consider bias injection attacks, where the attacker introduces a bias to the real signal.

In a bias injection attack, the attacker's goal is to deceive the controller to produce an erroneous control action. These attacks may be defined as attacks on the system outputs, but keeping those outputs compatible with the measurement equation of the system [21].

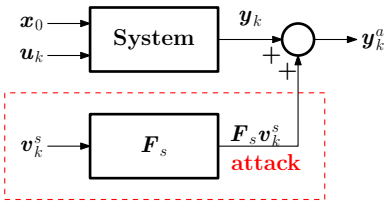


Figure 2: Bias injection attack. The attacker steals a sample from the measured variable and manipulates (adds a value to) it to deceive the controller.

Figure 2 shows a way of implementing this kind of attack; there F_s indicates the output where the bias is added and, v_k^s is the bias the attacker wants the system to have. For these attacks, the attacker does not require knowledge about the model of the system; the knowledge about current values of the measurements is enough. With current values and the span of the measurements the attacker can compute an attack vector. The block diagram shows the signals that satisfy (6).

3.4 Anomaly Detection and Attack Mitigation

Anomaly detection and isolation techniques detect an anomaly and identify the particular device that is sending misleading data. There are several isolation mechanisms in the fault tolerant control literature, in this paper we work with structured residuals. Structured residuals have observers that are insensitive to one specific input and are sensitive to the other ones. This mechanism is built using Unknown Input Observers (UIOs). An UIO is a generalization of the Luenberger Observer [6]. A full-order observer can be expressed as

$$\begin{aligned} w_{k+1} &= Fw_k + TBu_k + K_U y_k^a, \\ \hat{x}_k^u &= w_k + Hy_k^a, \end{aligned} \tag{7}$$

where $\hat{x}_k^u \in \mathbb{R}^n$ is the estimated state vector, and $w_k \in \mathbb{R}^n$ is the state vector of this full-order observer, which is computed by the linear transformation $w_k = Tx_k$. F, T, and K_U are matrices that must be designed such that the unknown input v_s is decoupled from the other inputs.

The anomaly isolation mechanism is responsible for identifying the component that is sending misleading information. To achieve this purpose, and based on the concept of a structured residual set, one UIO is associated to each output; each observer is insensitive to anomalies on one sensor and sensitive to the other ones. The j^{th} observer is designed to be insensitive to anomalies on sensor j^{th}. Hence, we need to build a bank of observers.

We assume that there are no simultaneous anomalies on sensors, i.e. only one anomaly is active on the whole set of sensors, and we use UIOs to correct the malicious signal. The purpose of an attack mitigation mechanism is to reduce the impact of sensor attacks over networked controlled systems. Figure 3 shows a block diagram of this mechanism.

The diagram shows a typical networked controlled system, with an addition, a component to detect anomalies and mitigate attacks. All these blocks are developed as external digital controllers running over one industrial computer, or even over a PLC. This architecture only requires information about control actions, sensor information,

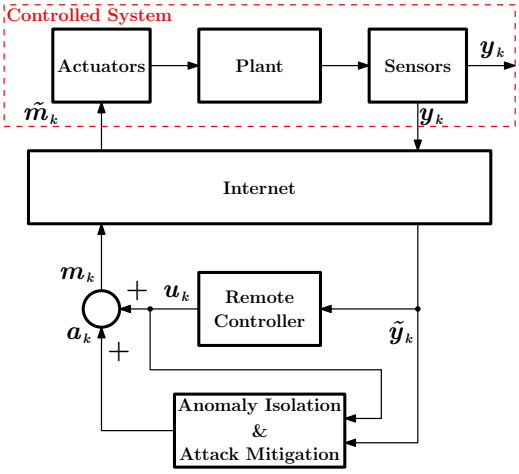


Figure 3: Mechanism to mitigate sensor attacks. It uses values reported by sensors, actuators, and the plant model to determine whether there is an attack or not. If an attack is detected then an additional signal, a_k, is injected.

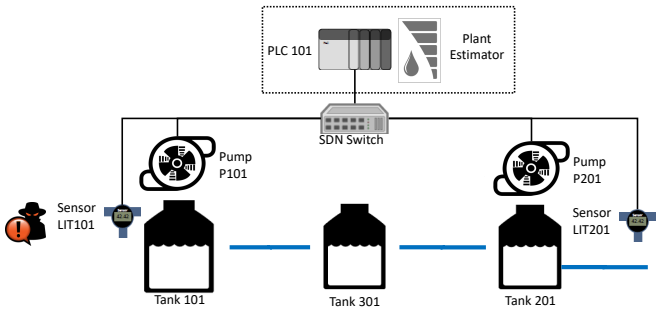


Figure 4: Experimental Setup for Virtual Experimentation of Non Linear Plant. A Mininet LAN was created with the control system components. The PLC101 only interacts with the plant through the sensors and actuators.

and the capability to modify the information that the actuator of the system receives through the network infrastructure. When there is no attack on sensors of the system, the signal a_k is equal to zero and the signal m_k is equal to u_k, but when an attack is isolated an additional signal a_k is computed and added to the original signal u_k and this sum m_k is sent to the controller in order to mitigate the effect of the attack.

4 EXPERIMENTAL SETUP

In previous work, we modeled attack-response with a fairly simple simulation of a physical linear system with uncoupled stages [19, 20]. This paper extends our previous work to incorporate a new nonlinear plant and a new attack-mitigation strategy based on UIOs. Our previous work had uncoupled batch processes where the variables were independent, and the control strategy was simple because we only needed to keep track of one variable to design one control signal. In our new model there are relations among the variables that must be handled, thus creating the need of a robust (potentially more complex) control strategy. In addition, message delays must be carefully managed. If an experimental environment does not represent delays, that environment is ignoring a key feature of real plants. On the other hand, an experimental environment needs to answer in real-time for a plant to behave as expected.

In particular, in this paper we model a three-tank system [2] as illustrated in Figure 4 and according to equations in 8. Where $q_{13}(t)$ represents the water flow-rate from tank 1 to tank 3, $q_{32}(t)$ represents the water flow-rate from tank 3 to tank 2, $q_{20}(t)$ represents the water flow-rate of tank 2 draining the water out of the system, $Q_1(t)$ and $Q_2(t)$ represents the input water flow-rate to tanks 1 and 2 respectively, $L_1(t)$, $L_2(t)$, and $L_3(t)$ are the levels of the tanks 1, 2, and 3 respectively, S represents the cross sectional area of the tanks, S_n represents the cross sectional area of the pipes between tanks, μ_{13} represents the outflow coefficient from tank 1 to tank 3, μ_{32} represents the outflow coefficient from tank 3 to tank 2, and μ_{20} represents the outflow coefficient of tank 2.

$$
\begin{aligned}
S\frac{d}{dt}L_1(t) &= Q_1(t) - q_{13}(t), \\
S\frac{d}{dt}L_2(t) &= Q_2(t) + q_{32}(t) - q_{20}(t), \\
S\frac{d}{dt}L_3(t) &= q_{13}(t) - q_{32}(t), \\
q_{13}(t) &= \mu_{13} S_n \operatorname{sgn}[L_1(t) - L_3(t)]\sqrt{2g|L_1(t) - L_3(t)|} \\
q_{32}(t) &= \mu_{32} S_n \operatorname{sgn}[L_3(t) - L_2(t)]\sqrt{2g|L_3(t) - L_2(t)|} \\
q_{20}(t) &= \mu_{20} S_n \sqrt{2gL_2(t)},
\end{aligned}
\tag{8}
$$

In our prototype, we emulate the control system of the nonlinear plant, perform some attacks on the integrity of the data reported by one of the sensors and implement some countermeasures to mitigate the impact of these attacks.

Figure 4 shows the topology of our prototype. We used Mininet [17] and MiniCPS [3] to emulate the nonlinear plant and its control system. MiniCPS extends Mininet [17], a light virtualization environment tailored for SDN experiments and emulation, to support EtherNet/IP, an industrial network protocol commonly used in ICS for communications between the supervisory and field networks. To simulate the physical behavior of this process and emulate network components of this system, we extended MiniCPS [3] with a nonlinear plant, a malicious sensor performing integrity attacks, and a PLC that can identify this integrity attack and perform mitigation actions. Originally, in MiniCPS architectures, the PLCs directly read plant state and modify it by changing actuator values. Instead, we introduced a field communication layer, where sensors and actuators are the only elements that can interact directly with the plant. This creates a more realistic environment of the behavior of a real-world industrial control system.

In our topology, we have traditional network control system elements: a plant, a pair of actuators (P101 and P201), a pair of sensors (LIT101 and LIT201), and a PLC (PLC101) that includes a state estimator. Each one of these elements is represented by a

Mininet node. A Mininet node, is a light virtualization element, also called a container. Each node in Mininet has its own namespace and communication protocol stack. This means that when, for example, the sensor LIT101 sends a message to the PLC101, the whole Linux communication protocol stack is involved. We argue that this gives more realism to the experiments than simply using a simulation, that models a network, only based on link bandwidth and delay parameters.

As mentioned before, we extended the MiniCPS model by including sensors and actuators. For this reason, our plant node has no network interfaces and reads the state of the actuators using the MiniCPS database. The state of these actuators is used as input for the plant. Then, using the previous state of the plant and the actuators state, the next state of the plant is calculated. The next state of the plant is calculated solving the nonlinear differential equation system of the plant. For this, we used the `Odeint` solver of Python Numpy. Once the new state is calculated, the plant writes on the MiniCPS database the value of each tank level. Sensors LIT101 and LIT201 read the respective tank level and send it to the PLC101. The PLC101 estimates the state of the plant, detects integrity attacks on the sensor and if an attack is present, calculates a compensation action that mitigates this attack. Finally, the PLC101 sends the appropriate levels to the actuators to keep the level of each tank at the desired level. All the messages exchanged in the network pass trough the switch; the topology and flows could be controlled by the SDN Controller in the topology. In the current setup, however, we did not implement any SDN application because the defense mechanism is installed directly into the PLC101.

5 RESULTS

The controlled plant is a hydraulic system with three cylindrical tanks of the same dimensions, two cylindrical pipes connect the tanks, and two pumps governed by motors DC that supply liquid for two of the three tanks [2]. This system is a testbed that serves as an example of liquid storage in industrial or water treatment plants. In these processes a fundamental requirement is to maintain the predefined operational points because the chemical reactions that are fed with them require it. This hydraulic system prototype is commonly used to verify the effectiveness of controllers and model-based fault diagnosis systems.

Symbol	Value
S	0.0154 m^2
S_n	$5 \times 10^{-5} m^2$
$\mu_{13} = \mu_{32}$	0.5
μ_{20}	0.6
$Q_{i\,max}$ $i \in [1, 2]$	$1.5 \times 10^{-4} m^3 s^{-1}$
$L_{j\,max}$ $j \in [1, 2, 3]$	0.62 m

Table 2: Parameter values of the three tank system.

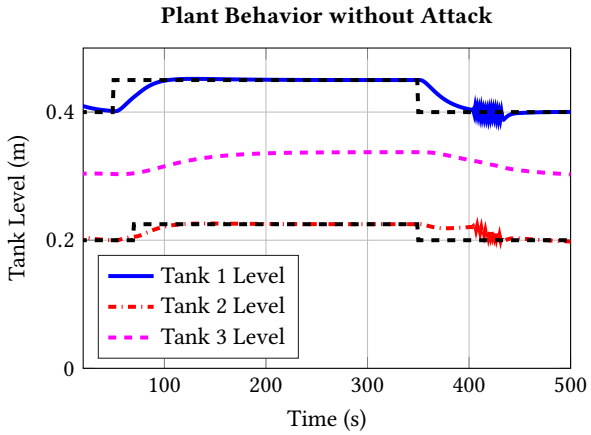


Figure 5: Water Tank Level behavior without an attack. The PLC101 can maintain the water tank level in the desired set-points.

Table 2 shows the parameter values we used in our Python implementation of the nonlinear equation (8) of the three tanks system. The operation point of the level of the tanks is given by $L_{1_{oper}} = 0.4$ m, $L_{2_{oper}} = 0.2$ m, and $L_{3_{oper}} = 0.3$ m, where $L_{j_{oper}}$ ($j \in [1, 2, 3]$) represents the level of tank j respectively. The corresponding operation point for the pumps that govern the inputs of the tanks is given by $Q_{1_{oper}} = 3.5018 \times 10^{-5}$ $m^3 s^{-1}$ and $Q_{2_{oper}} = 3.1838 \times 10^{-5}$ $m^3 s^{-1}$ where $Q_{i_{oper}}$ ($i \in [1, 2]$) represents the input flow rates needed to reach the operation point levels, when the system does not have any disturbance.

For the three tank system we can find a linear discrete-time model around the operation point (equation (1)) so that we can design the control parameters, and the UIO parameters. The sampling time of the system is $T_s = 1$ s. The matrices A, B, and C are given by

$$A = \begin{bmatrix} 0.9888 & 0.0001 & 0.0112 \\ 0.0001 & 0.9781 & 0.0111 \\ 0.0112 & 0.0111 & 0.9776 \end{bmatrix}, \quad B = \begin{bmatrix} 64.5687 & 0.0014 \\ 0.0014 & 64.2202 \\ 0.3650 & 0.3637 \end{bmatrix},$$

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

The state variables are chosen as the liquid levels in tanks 101, 301 and 201, respectively.

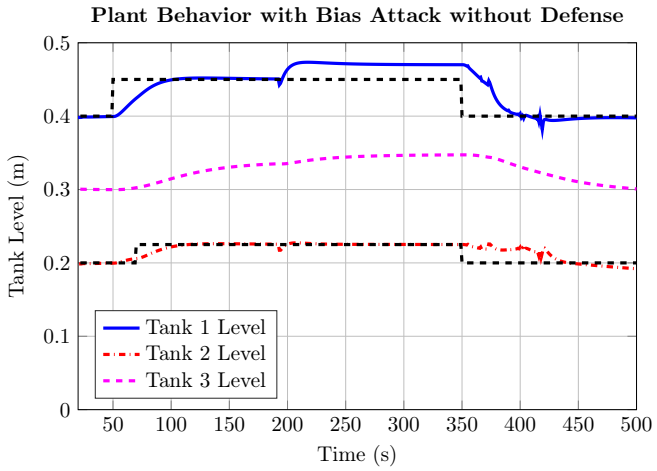
The remote controller gain for state feedback is K_1 and the integral gain is K_2, these gains are obtained using the pole placement technique, and they are given by

$$K_1 = 10^{-4} \begin{bmatrix} 21.6 & 3.0 & -5.0 \\ 2.9 & 19.0 & -4.0 \end{bmatrix}, \; K_2 = 10^{-4} \begin{bmatrix} -0.95 & -0.32 \\ -0.30 & -0.91 \end{bmatrix}.$$

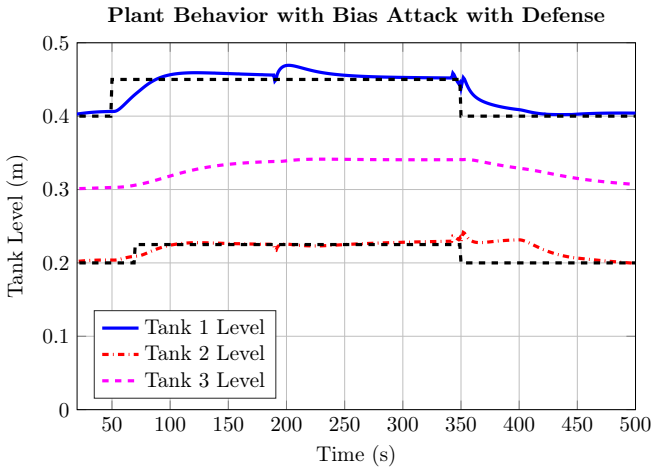
The Luenberger estimator gain is given by

$$L = \begin{bmatrix} 0.9995 & 0.0005 \\ 0.0005 & 0.9995 \\ 45.0167 & 42.5017 \end{bmatrix}.$$

All mentioned components are part of the original networked control system and were developed before any security design.



(a) Water Tank Levels with a bias injection attack on LIT101, of 20×10^{-3} for $200 \leq t \leq 350$, and no defense on the PLC101. This bias injection attack causes water level to stay above the desired setpoint during the attack.



(b) Water Tank Levels with a bias injection attack on LIT101, of 20×10^{-3} for $200 \leq t \leq 350$, and defense on the PLC101. The defense in the PLC101 is able to mitigate the impact of the bias attack, and the water level is mostly maintained in the desired setpoint.

Figure 6: Bias attack experiments.

Figure 5 shows the response of the system to changes in the set-points or reference inputs. The figure presents the level of the three tanks and the dotted black line indicates the desired behavior following the setpoints established for each water tank. The control system acts upon Tank1 and Tank3 levels; the level of Tank 2 is a consequence of the level in the other two tanks. The figure shows that the water level follows closely the desired results. There is a small delay between a change of the setpoint and the water tank level, but this delay is always present in Control Systems and is called the transient response of the system.

The Anomaly Isolation and Attack Mitigation block is designed using two UIOs, one for each sensor that may be attacked. [6]. UIO1 has three inputs: 1) the control action of pump P101, 2) the control action of pump P201, and 3) information from the sensor LIT201. It is worth noting that UIO1 does not have information about sensor LIT101. The recursive equation for the UIO1 is given by equation (7) and the matrices

$$F1 = \begin{bmatrix} 0.9888 & 1.0000 & 0.0112 \\ 0 & 0.0010 & 0 \\ 0.0112 & -0.9890 & 0.9776 \end{bmatrix}, \quad K1_U = \begin{bmatrix} -1.0000 \\ -0.0010 \\ 1.0000 \end{bmatrix},$$
$$T1 = \begin{bmatrix} 1.0000 & -0.0001 & 0 \\ 0 & 0 & 0 \\ 0 & -0.0001 & 1.0000 \end{bmatrix}, \quad H1 = \begin{bmatrix} 0.0001 \\ 1.0000 \\ 0.0001 \end{bmatrix}.$$

UIO2 also has three inputs: 1) the control action of pump P101, 2) the control action of pump P201, and 3) information from the sensor LIT101. It is worth noting that UIO2 does not have information about sensor LIT201.

The recursive equation for UIO2 is given by (7) and the matrices

$$F2 = \begin{bmatrix} 0.0010 & 0 & 0 \\ -1.0000 & 0.9781 & 0.0111 \\ -0.9889 & 0.0111 & 0.9776 \end{bmatrix}, \quad K2_U = \begin{bmatrix} -0.0010 \\ 1.0000 \\ 1.0000 \end{bmatrix},$$
$$T2 = \begin{bmatrix} 0 & 0 & 0 \\ -0.0001 & 1.0000 & 0 \\ -0.0001 & 0 & 1.0000 \end{bmatrix}, \quad H2 = \begin{bmatrix} 1.0000 \\ 0.0001 \\ 0.0001 \end{bmatrix}.$$

5.1 Integrity Attack

We implemented the bias integrity attacks described in section 3.3. In this attack, the sensor LIT101 attacks the integrity of the water tank readings by subtracting a value from the sampled value. In our case, this value was 0.02m. Hence, for each sample reported by LIT101 a value of 0.02m was subtracted. This attack begins around 200s and ends at 350s.

Figure 6a shows the results of the bias attack when no defense is present. We can see the impact of the attack; the PLC101 believes that the plant is some centimeters below the desired setpoint level, this causes the PLC101 to compensate for this error causing the water tank level to stay above the desired setpoint level across the entire duration of the attack. We can see that the water tank level 2 is also impacted by these erroneous control actions. Figure 6b shows the system response when our defense is present. The defense is able to properly mitigate the impact of the bias attack by keeping the water level above almost all of the attack duration. Shortly after the attack starts, it is identified and the PLC starts correcting the control action to mitigate the impact of the attack. For this reason, after the initial overshoot present around the start of the attack, the water level starts returning to its desired point and stays in this point for the whole duration of the attack.

We performed additional experiments to further test the behavior of the biass attack and our defense. We ran five sets of experiments;

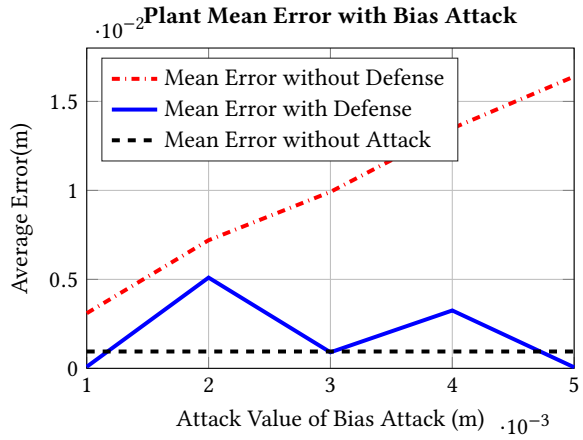


Figure 7: Mean error of tank1 water level with and without defense and different values for the bias attack. The maximum mean error with defense is 0.05. The error with the defense is always less than the error without defense.

in each set we tested the behavior of the system with and without defense. Also, we ran all the experiments for 500 seconds. In all the experiments, the attack starts at 200 seconds and stops at 350 seconds, and the setpoints of water tank 1 and 2 were the same. During the bias attacks, we changed attack values, starting from 0.01 until 0.05, increasing 0.01 each time. To synthesize the results of these experiments, we calculated the mean error of the tank 1 water level. The mean error is defined as the average error between the desired setpoint for all times t of the experiment, and the current water tank 1 level. Figure 7 shows the results of these experiments. The mean error without defense increases as the value of the bias attack is increased. Until it reaches a value higher than 0.015. In contrast, when our defense is present, the mean error stays at a maximum of 0.05. For comparison, we show in the figure the mean error when no attack is present, we can see that the behavior of the plant is very close to the behavior when no attack is present.

6 CONCLUSIONS AND FUTURE WORK

This paper presents a virtual environment to emulate a Network Control System controlling a nonlinear plant, and detect and identify attacks. The environment uses UIOs to identify the source of a malicious attack. We ran bias attacks to evaluate our environment responses and found that the mean error of tank1 water level, with defense, was always lower than mean error without defense, and close to mean error without attack. In future work we plan to extend our virtual environments to consider Real-Time Operating Systems.

ACKNOWLEDGMENTS

This work is partially supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0135, the Department of Commerce by NIST Award 70NANB17H282 and by the Colombian Administrative Department of Science, Technology, and Innovation (Colciencias).

REFERENCES

[1] Thiago Alves, Rishabh Das, and Thomas Morris. 2016. Virtualization of Industrial Control System Testbeds for Cybersecurity. In *Proceedings of the 2Nd Annual Industrial Control System Security Workshop (ICSS '16)*. 10–14.
[2] Amira. 2002. *Laboratory Setup: Three-tank System DTS200.* Amira GmbH, Duisburg.
[3] Daniele Antonioli and Nils Ole Tippenhauer. 2015. MiniCPS: A Toolkit for Security Research on CPS Networks. In *Proceedings of the First ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy*. 91–100.
[4] Richard Candell, Timothy A. Zimmerman, and Keith A. Stouffer. 2015. *An Industrial Control System Cybersecurity Performance Testbed.* Technical Report NISTIR 8089. National Institute of Standards and Technology. 55 pages.
[5] A. A. Cardenas, S. Amin, and S. Sastry. 2008. Secure Control: Towards Survivable Cyber-Physical Systems. In *Proceedings of the 28th International Conference on Distributed Computing Systems Workshops*. 495–500.
[6] Jie Chen and Ron J. Patton. 1999. *Robust Model-based Fault Diagnosis for Dynamic Systems.* Kluwer Academic Publishers, Norwell, MA, USA.
[7] L.F. Combita, J. Giraldo, A. Cardenas, and N. Quijano. In Press. *Handbook of Dynamic Data Driven Applications Systems.* Springer Nature Switzerland AG, Chapter DDDAS for Attack Detection and Isolation of Control Systems.
[8] L. F. Cómbita, A. A. Cárdenas, and N. Quijano. 2017. Mitigation of Sensor Attacks on Legacy Industrial Control Systems. In *2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC)*. 1–6.
[9] H. Gao, Y. Peng, K. Jia, Z. Wen, and H. Li. 2015. Cyber-Physical Systems Testbed Based on Cloud Computing and Software Defined Network. In *2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*. 337–340.
[10] BéLa Genge, Christos Siaterlis, Igor Nai Fovino, and Marcelo Masera. 2012. A Cyber-physical Experimentation Environment for the Security Analysis of Networked Industrial Control Systems. *Computers and Electrical Engineering Journal* 38, 5 (Sept. 2012), 1146–1161.
[11] J. Giraldo, E. Sarkar, A. A. Cardenas, M. Maniatakos, and M. Kantarcioglu. 2017. Security and Privacy in Cyber-Physical Systems: A Survey of Surveys. *IEEE Design Test* 34, 4 (Aug 2017), 7–17.
[12] Benjamin Green, Marina Krotofil, and Ali Abbasi. 2017. On the Significance of Process Comprehension for Conducting Targeted ICS Attacks. In *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and PrivaCy (CPS '17)*. 57–67.
[13] Benjamin Green, Anhtuan Lee, Rob Antrobus, Utz Roedig, David Hutchison, and Awais Rashid. 2017. Pains, Gains and PLCs: Ten Lessons from Building an Industrial Control Systems Testbed for Security Research. In *10th USENIX Workshop on Cyber Security Experimentation and Test (CSET 17)*. USENIX Association, Vancouver, BC.
[14] Prageeth Gunathilaka, Daisuke Mashima, and Binbin Chen. 2016. SoftGrid: A Software-based Smart Grid Testbed for Evaluating Substation Cybersecurity Solutions. In *Proceedings of the 2Nd ACM Workshop on Cyber-Physical Systems Security and Privacy (CPS-SPC '16)*. ACM, New York, NY, USA, 113–124.
[15] Pradeeban Kathiravelu and Luís Veiga. 2016. Software-Defined Simulations for Continuous Development of Cloud and Data Center Networks. In *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*, Christophe Debruyne, Hervé Panetto, Robert Meersman, Tharam Dillon, eva Kühn, Declan O'Sullivan, and Claudio Agostino Ardagna (Eds.). Springer International Publishing, Cham, 3–23.
[16] P. Kathiravelu and L. Veiga. 2017. SD-CPS: Taming the Challenges of Cyber-Physical Systems with a Software-Defined Approach. In *2017 Fourth International Conference on Software Defined Systems (SDS)*. 6–13.
[17] Bob Lantz, Brandon Heller, and Nick McKeown. 2010. A Network in a Laptop: Rapid Prototyping for Software-defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (Hotnets-IX)*. ACM, New York, NY, USA, Article 19, 6 pages.
[18] D. Mashima, B. Chen, P. Gunathilaka, and E. L. Tjiong. 2017. Towards a Grid-wide, High-fidelity Electrical Substation Honeynet. In *2017 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. 89–95.
[19] Andrés F. Murillo Piedrahita, Vikram Gaur, Jairo Giraldo, Alvaro A. Cardenas, and Sandra Julieta Rueda. 2018. Leveraging Software-Defined Networking for Incident Response in Industrial Control Systems. *IEEE Software* 35, 1 (January 2018), 44–50.
[20] Andrés F. Murillo Piedrahita, Vikram Gaur, Jairo Giraldo, Alvaro A. Cardenas, and Sandra Julieta Rueda. 2018. Virtual Incident Response Functions in Control Systems. *Computer Networks* 135 (2018), 147 – 159.
[21] André Teixeira, Daniel Pérez, Henrik Sandberg, and Karl Henrik Johansson. 2012. Attack Models and Scenarios for Networked Control Systems. In *Proceedings of the 1st International Conference on High Confidence Networked Systems (HiCoNS '12)*. ACM, New York, NY, USA, 55–64. https://doi.org/10.1145/2185505.2185515