Hide and Seek: An Architecture for Improving Attack-Visibility in Industrial Control Systems

Jairo Giraldo[1], David Urbina[1], Alvaro A. Cardenas[2(✉)], and Nils Ole Tippenhauer[3]

[1] The University of Texas at Dallas, Richardson, TX 75080, USA
{jairo.giraldo,david.urbina}@utdallas.edu
[2] University of California Santa Cruz, Santa Cruz, CA 95064, USA
alvaro.cardenas@ucsc.edu
[3] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
tippenhauer@cispa.saarland

Abstract. In the past years we have seen an emerging field of research focusing on using the "physics" of a Cyber-Physical System to detect attacks. In its basic form, a security monitor is deployed somewhere in the industrial control network, observes a time-series of the operation of the system, and identifies anomalies in those measurements in order to detect potentially manipulated control commands or manipulated sensor readings. While there is a growing literature on detection mechanisms in that research direction, the problem of *where* to monitor the physical behavior of the system has received less attention.

In this paper, we analyze the problem of where should we monitor these systems, and what attacks can and cannot be detected depending on the location of this network monitor. The location of the monitor is particularly important, because an attacker can bypass attack-detection by lying in some network interfaces while reporting that everything is normal in the others. Our paper is the first detailed study of what can and cannot be detected based on the devices an attacker has compromised and where we monitor our network. We show that there are locations that maximize our visibility against such attacks. Based on our analysis, we design a low-level security monitor that is able to directly observe the field communication between sensors, actuators, and Programmable Logic Controllers (PLCs). We implement that security monitor in a realistic testbed, and demonstrate that it can detect attacks that would otherwise be undetected at the supervisory network.

1 Introduction

One of the recent research trends for the security of Industrial Control Systems (ICS) is to monitor the sensor and control signals being exchanged between different components of the system to verify that the system is operating as intended [5,7,12,23]. For example, if we have a sensor that monitors the height of

R. H. Deng et al. (Eds.): ACNS 2019, LNCS 11464, pp. 175–195, 2019.
https://doi.org/10.1007/978-3-030-21568-2_9

a bouncing ball, then we know that this height follows the differential equations from Newton's laws of mechanics. Thus, if a sensor reports a trajectory that is not plausible given the laws of physics, we can immediately identify that something has gone wrong with the sensor (a fault or an attack).

While previous research has contributed greatly to the literature, we have found that most papers working on this topic do not explicitly describe the trust assumptions for all parts of a control loop—controllers, actuators, and sensors.

In this paper, we show that without explicit trust assumptions, attacker models proposed in related work are ambiguous. In particular, we analyze the implicit assumptions made in previous works, and then use a logical attack-detection architecture to elucidate hidden assumptions, limitations, and possible improvements. Then, we develop and implement an architecture to maximize the visibility of attacks.

We summarize our main contributions in this work as follows:

- We review and classify different classes of attacks on a control loop, and map them to real-wold network topologies for industrial control systems.
- We show how implicit trust in subsets of the components in a real system can lead to attacks that deny visibility of the physical process to the control logic or SCADA.
- We provide a table articulating in detail the trust assumptions needed to be able to detect attacks when monitoring at the supervisory layer and at the field layer. As far as we are aware, we are the first to justify why monitoring at the field layer minimizes the number of devices we need to trust in order to detect attacks.
- We design and implement a *deep monitoring* system at the field layer and demonstrate the feasibility of our proposed system through a series of experiments. As far as we are aware, we are the first to illustrate the practical differences between implementing a security monitor at the supervisory layer vs. the field layer.

The remainder of this paper is organized as follows: In Sect. 2, we provide background on ICS networks, and related work. We then propose our new security monitoring architecture in Sect. 3. Based on that concept, we design and implement a deep ICS monitor in Sect. 4. Finally, we present the results of our prototype in Sect. 5.

2 Background

In this section, we first briefly summarize industrial control system networks, and then review related work on attack detection in ICS networks.

2.1 ICS Network Layers

Control systems have a layered hierarchy [24]. The two layers closest to the physical process are Layer 1 and Layer 0:

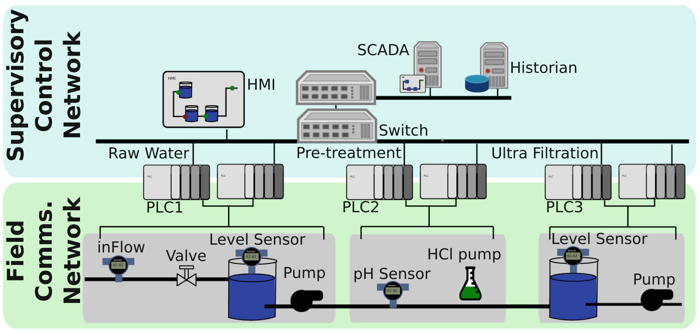


Fig. 1. A supervisory control network (SCN) enables communications between a central control server and embedded controllers. Field communication networks (FCN) enable controllers to contact sensor, actuators and remote IO terminals.

Layer 1 is where Supervisory Control and Data Acquisition (SCADA) Systems and other servers communicate with remote control equipment like Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs). The communication between servers in a control room and these control equipment is done via a Supervisory Control Network **(SCN)**;

Layer 0 is where PLCs or RTUs interface with sensors (thermometers, tachometers, etc.) and actuators (pumps, valves, etc.) in the field. While traditionally this interface has been analog (e.g., 4–20 mA), the growing numbers of sensors and actuators as well as their increased intelligence and capabilities, has given rise to new Field Communication Networks **(FCN)** where the PLCs and other types of controllers interface with remote Input/Output boxes or directly with sensors and actuators using new Ethernet-based industrial protocols like EtherNet/IP and Profinet, and wireless networks like WirelessHART. Several ring topologies have also been proposed to avoid a single point of failure for these networks, such as the use of Device Level Ring (DLR) over EtherNet/IP.

Figure 1 illustrates these two networks in the ICS we analyze later in the paper.

SCN and FCN networks have different communication requirements and different industrial network protocols. While SCN can tolerate delays of up to the order of seconds, FCN typically require an order of magnitude of lower communication delays, typically enabling communications between devices with a period of 400 μs.

2.2 Previous Work

In this paper we focus on *network intrusion detection* for ICS. One of the first papers to consider network intrusion detection in industrial control networks was Cheung et al. [8]. Their work articulated that network anomaly detection might be more effective in control networks where communication patterns are more regular and stable than in traditional IT networks. Similar intrusion detection systems have been proposed for building control systems [6] and general cyber-physical systems [19]; however, as Hadžiosmanović et al. showed [11], intrusion detection systems that fail to incorporate domain-specific knowledge of the context in which they are operating perform poorly in practical scenarios.

Even worse, an attacker can send false sensor or control values to the physical process while complying to typical IT traffic patterns used by classical intrusion detection systems (Internet Protocol (IP) addresses, protocol specifications with finite automata, connection logs, etc.). These false data injection attacks [5,10,16] manipulate the process under control by sending malicious sensor or control commands, and can cause waste water spills [1], or can sabotage nuclear enrichment by manipulating the rotation frequency of centrifuges [9,13]. To detect these types of attacks we need to monitor the sensor and control algorithms in the systems; i.e., the *semantic* information of the ICS [2,5,12,23].

Previous efforts on semantic monitoring for ICS, however, have been vague in describing the specific types of attacks their proposals can and cannot detect. In particular, previous work implicitly assumes certain elements in a control loop are not compromised in order for their system to work, and this lack of specificity leads to potential threat vectors not previously anticipated.

Before we describe the vulnerabilities of previous work, notice that an attacker can compromise different devices in its goal to physically attack an ICS. In particular the adversary can compromise and launch attacks from (1) SCADA servers [15], (2) controllers/PLCs [14], (3) sensors [16], and (4) actuators [22]. As we will show, it is important to understand where the adversary is launching attacks because it can have a drastic effect in attack-detection systems.

For example, McLaughlin [18] focuses on the field layer of a control system; specifically it tackles the problem of how to verify that control signals u_k sent by the PLC to the actuators do not drive the system to an unsafe state. The proposed approach, C^2, mediates all control signals u_k sent by PLCs to the physical system. McLaughlin mentions that "C^2 mitigates all control channel attacks against devices, and only requires trust in process engineers and physical sensors." This is, however, not true, as **an attacker that has compromised an actuator or a Remote I/O, can bypass C^2:** a PLC can send correct control signals, but if the actuators are compromised they do not need to follow the orders from the PLC, and can discard them to continue attacking the system.

Similarly Hadžiosmanović et al. [12] use network traces from an industrial site using Modbus/TCP to detect attacks by monitoring the state variables of the system, including constants, attribute data, and continuous data. Their network data was captured at the supervisory network, and this means that they are *implicitly trusting the PLCs to tell the truth to the supervisory network.* However, **if a PLC is compromised, it can lie to the supervisory network interface stating that everything is working properly, while at the same time sending erroneous commands via the field communications interface, and this attack would not be detected by monitoring the supervisory network**.

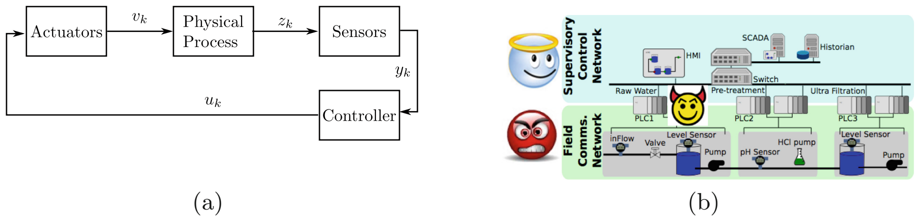


Fig. 2. (a) A control loop at time k, with sensor values y_k, control values u_k, actuator action v_k, and state of the system z_k. (b) A compromised PLC can send manipulated control commands to devices in the field while reporting an incorrect status of the system to the supervisory layer.

In another example, Carcano et al. [4] propose a safety monitoring system and raise alerts whenever it is in a critical state (or approaching a critical state). In order to detect that they are approaching an unsafe state, they *implicitly assume trusted sensors*. **If an attacker compromises the sensors, or the PLCs, it can lie to the network security monitor located in the supervisory network and bypass the system**.

The work of Carcano and C^2 rely on having sensors that haven't been compromised. It seems reasonable to assume that if the sensors are trustworthy, we should be able to detect if the system is approaching an anomalous or dangerous state and react accordingly. *Zero-dynamics attacks* [20–22] are examples of attacks where **even if we assume the sensors and the PLCs are not compromised, attackers with compromised actuators can mislead state estimation algorithms**.

In summary, attackers have many vectors for attacks, and none of the previous we analyzed has considered a detection architecture that can prevent an attacker from launching attacks while remaining hidden.

3 An Architecture to Reveal Hidden Attacks

Physical processes are regulated by a control loop, consisting of the following components: (1) the physical phenomena of interest (sometimes called the process), (2) sensors to observe the physical system and send a time series y_k denoting the value of the physical measurement at time k (e.g., the voltage at 3 a.m is 120 KV), (3) based on the sensor measurements received y_k, the controller/PLC sends control commands u_k (e.g., open a valve by 10%) to actuators, and (4) actuators that change the control command to an actual physical change (the device that opens the valve). To differentiate between the real state of the system and the sensor reading, let z_k denote the real value and y_k the one reported by sensors. Similarly, actuators might implement a different control action v_k than the one received from the PLC u_k. A summary is shown in Fig. 2a.

3.1 Limitations of Security Monitors Located (Only) at the Supervisory Control Network

Most of the previous work on network security monitoring has deployed network intrusion detection systems at the SCN; however, if an anomaly detection system is only deployed in the supervisory control network then a compromised PLC can send manipulated data to the field network, while pretending to report that everything is normal back to the supervisory control network, as illustrated in Fig. 2b. In the Stuxnet attack, the attacker compromised a PLC (Siemens 315) and sent a manipulated control signal u^a (which was different from the original u, i.e., $u^a \neq u$). Upon reception of u^a, the frequency converters periodically increased and decreased the rotor speeds well above and below their intended operation levels. While the status of the frequency converters y was then relayed back to the PLC, the compromised PLC reported a manipulated value $y_a \neq y$ to the control center (claiming that devices were operating normally). A similar attack was performed against the Siemens 417 controller [14], where attackers captured 21 s of valid sensor variables at the PLC, and then replayed them continuously for the duration of the attack, ensuring that the data sent to the SCADA monitors would appear normal [14].

If the network monitor is deployed at the supervisory control layer, it will not able to detect compromised PLCs, unless it is able to correlate information from other trusted PLCs, or unless it receives (trusted) sensor data directly (e.g., wireless sensors sending measurements directly to the control center). If the control center in the Stuxnet case had monitored the frequency converters directly through an independent channel, it could have detected the attack.

Another difference in the data visibility between FCN and SCN layers is that the request-and-respond communication generally implemented by SCN layers might miss some important data exchanges in the FCN layer: without a specific request-and-response exchange, the data of interest may not be present during the deep-packet inspection session. For example, if a specific data item request/response exchange occurs with a low frequency or under special circumstances, the data exchanged will be missed. For example, configuration files for field devices can be set so that they only to send data if some specific circumstances arise. Even if the PLC is trustworthy, this delay can prevent an anomaly detector at the SCN from detecting the onset of an attack.

3.2 Detectability of Attacks

In the previous section we saw how attackers can bypass intrusion detection systems when they have compromised a PLC and our monitor is in the supervisory network. If our intrusion detection system is in the field network, the attacker of the previous section cannot remain hidden as we can see the false commands coming out of the PLC and the incorrect sensor measurements from the sensors. But what if the attacker also compromises these other parts of the system?

We now systematically analyze what can be detected and what cannot be detected when we have access to data from the field devices (from the FCN) and when the attacker compromises different parts of the control loop, as illustrated in Fig. 3a. Attack 1 in Fig. 3a shows an attack on the actuator(s) $v_k \neq u_k$, the attack modifies the control command send to the plant. We note that the controller is not aware of the communication interruption. On the other hand, attack 2 in Fig. 3a shows an attack on the sensor(s) $y_k \neq z_k$, which allows the attacker to deceive the controller about the real state of the plant. The controllers can be compromised as well, as illustrated by attack 3 in Fig. 3a, $u_k \neq \mathcal{K}(y_k)$, where $\mathcal{K}$ is the logic the control algorithm should have implemented.

We also capture attacks coming from a compromised SCADA server as illustrated in Fig. 3b, as malicious control commands from the SCADA server or a malicious change of parameters to the controller will generate a false control command equivalent to $u_k \neq \mathcal{K}(y_k)$.

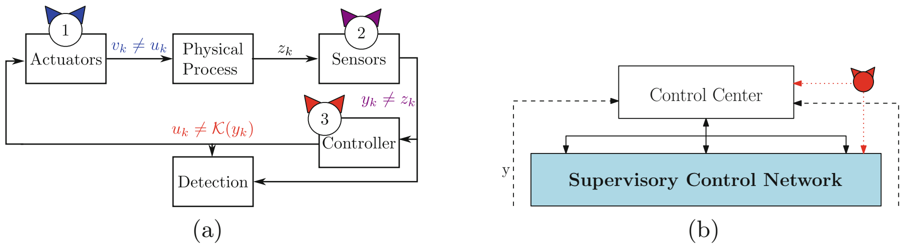


Fig. 3. (a) Different attack points in a control system: (1) Attack on the actuators, (2) Attack on the sensors, (3) Attack on the controller. (b) Attacks on central control or supervisory control network translate to attack $u_k \neq \mathcal{K}(y_k)$ in (a).

We now discuss the detectability of each attack.

1. If we trust the controller (e.g., the PLC) but do not trust sensors or actuators then, it is *game over*: the attacker can change the physical world with bad actuation actions while at the same time using the sensors to report that everything is working normally.
2. If we trust the actuators but not the controller or the sensors then it is also game over: the attacker can use the controller to send false control signals u_k^a to the actuator, while false sensor measurements can be generated to justify the false control action.
3. If we trust the sensors but not the controller or the actuators, then for most practical cases we can detect an attack using a Physics-Based Anomaly Detection (PBAD) as proposed e.g., by Urbina et al. [23]. A PBAD will work because the goal of the attack is to affect the physical system, and we assume we can monitor changes done by the attacker through the sensor time series y_k. Having said that, zero-dynamic attacks [22] are examples where even when we trust sensor measurements, we cannot detect attacks caused by a compromised actuator. Zero-dynamic attacks are rare and depend heavily on the properties of the process and the sensors we have deployed.

4. If we trust the actuator and the controller, then we know the control signal u_k will have the expected intended effect on the physical system. Any false data injected by the sensors will cause a control command u_k to be sent in response to these false measurements, and in turn, any implausible combination of control and sensor signals might be an indicator of an attack and can be detected by a PBAD.
5. If we trust the controller and the sensors, then we again have implausible combinations of control actions and sensor measurements that can be detected by a PBAD because we can see that the control command that was sent to the process did not have the expected result. With the possible exception of zero-dynamic attacks.
6. Finally, we can detect a compromised controller by identifying if a control action is the correct response to the current state of the system. This detection method is not PBAD, but it requires a Redundant Controller (RC) that can verify that the control action is indeed the intended one for the specific operation. Notice that a PBAD has limited use in this case as the physical evolution of the process with a compromised controller will still satisfy the "physics-based" model of the system because the false control signal u_k^a will be observed by the PBAD and it will match the expected result y^a caused by the attacker of the system.

Table 1. Detectability of attacks depending on which devices are compromised

Device status			Detection possible	Comment
PLC	Sensor	Actuator		
✓	×	×	×	False sensing hides bad actuation
×	×	✓	×	False sensing justifies bad controls
×	✓	×	∼	PBAD detectable except zero-dynamics
✓	×	✓	✓	PBAD detectable
✓	✓	×	✓	PBAD detectable
×	✓	✓	✓	RC-detectable
✓	✓	✓	✓	No attack possible

✓ = trusted/detection possible, × = compromised/detection not possible, ∼ = can detect most attacks except for zero-dynamics attacks

Summary and Takeaways. Table 1 summarizes our contributions. Based on our discussion, we can see that by monitoring FCN networks we can improve the number of attacks that we can detect (via PBAD or RC); however when more than one set of devices is compromised (e.g., sensors and actuators) detection is impossible, even at the FCN layer. Finally, while most previous work uses physics-based anomaly detection (PBAD) for detecting false data injection attacks, we showed in our analysis that PBAD is not enough; in particular we described why need to have a redundant controller (RC)-based detection.

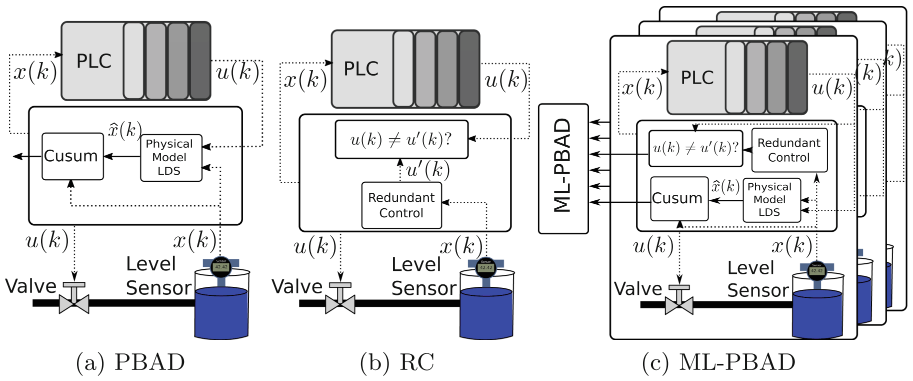


Fig. 4. (a) Illustration of physics-based anomaly detection (PBAD) algorithm. (b) Redundant control (RC) architecture to detect a compromised PLC. (c) Proposed attack detection architecture

Figure 4a illustrates a general Physics-Based Anomaly Detection (PBAD) system where the security monitor takes the control signal sent to the field and the sensor measurement received to see if its compatible to the predicted behavior of the system (a time-series anomaly detection algorithm like the CUSUM then performs statistical tests to see if the anomaly is persistent) [23]. Figure 4b on the other hand shows the RC detector, where a redundant (software-based) controller verifies that given the same sensor inputs, it obtains the same result as the controller.

3.3 Attack Detection Architecture

In the last section we saw that by monitoring the field communications of a control loop we can get better detection results than by monitoring the supervisory network; however, we still get limited attack-detection when an adversary has compromised more than one set of devices as summarized in Table 1.

Real-world industrial systems, however, are far larger than a single control loop. They contain multiple stages controlling interdependent parts of a complex system. For example, the process illustrated at the beginning of this paper in Fig. 1 has three stages of a water purification process, each controlled by a different set of PLCs, sensors, and actuators. These different parts of the process can be used to try to identify the attacks that are not detected in Table 1. For example, if the attacker compromises both sensors and actuators in one control loop, then it can take complete control of the system without being detected; however the effects on one loop will have other evident side-effects on another system and we can hope to detect the attack there. This will need coordination between different anomaly detection systems in each loop, so they can verify with each other what each of them is currently "seeing".

Our proposed anomaly detection architecture is illustrated in Fig. 4c. As described in the previous subsection, by deploying network security monitors in

Table 2. Comparison of detection capabilities of PBAD, RC, ML-PBAD.

	Attack location		
Detection Module	Actuator	Sensor	PLC
PBAD	◐	◐	○
RC	○	○	●
ML-PBAD	●	●	○

◐Partially detect ●Fully detect ○Cannot detect

the field communication network of the system we can use a PBAD algorithm to detect compromised sensors or actuators, and also detect compromised PLCs by using the RC attack-detection algorithm. Each of these anomaly detection tools will then share their data with a Multi-Loop Physics-Based Anomaly Detection (ML-PBAD) algorithm that will detect if the reports from a control loop in one subsystem are consistent with the other control loops. Table 2 summarizes this.

In the next section we will discuss the development and implementation of our architecture in an industrial system.

4 Implementation of Our Security Monitor

In this section, we present the design and implementation of a security monitor that is explicitly placed as *deep* in the ICS network hierarchy as possible—in the field network immediately next to sensors and actuators, and which reports data to a supervisory ML-PBAD algorithm. As such, the monitor is expected to reliably obtain information from the sensors and actuators, without the risk of obtaining manipulated data from intermediate PLCs or SCADA.

4.1 Testbed Description

The testbed we use for our experiments is a water treatment plant consisting of 6 main stages to purify raw water. The testbed is described in more detail in [17]. The testbed has a total of 12 PLCs (6 main PLCs and 6 in backup configuration to take over if the main PLC fails). *Raw water storage* is the part of the process where raw water is stored and it acts as the main water buffer supplying water to the water treatment system. It consists of one tank, an on/off valve that controls the inlet water, and a pump that transfers the water to the ultra filtration (UF) tank. In *Pre-treatment* the Conductivity, pH, and Oxidation-Reduction Potential (ORP) are measured to determine the activation of chemical dosing to maintain the quality of the water within some desirable limits. *Ultra Filtration* is used to remove the bulk of the feed water solids and colloidal material by using fine filtration membranes that only allow the flow of small molecules. The accumulated contaminants are removed by back-washing away the membrane surface depending on the measure of a differential pressure sensor located at the two ends of the UF. After the small residuals are removed by the UF system, the remaining chlorines are destroyed in the *Dechlorinization* stage, using ultraviolet chlorine destruction unit and by dosing a solution of sodium bisulphite. *Reverse*

Osmosis (RO) system is designed to reduce inorganic impurities by pumping the filtrated and dechlorinated water with a high pressure through reverse osmosis membranes. Finally, in *RO final product* stage stores the RO product (clean water).

Each stage has two PLCs (one primary and one redundant in hot-standby mode). The field devices, i.e. sensors/actuators, send and receive sensor information and control actions, respectively, to/from the PLCs through Remote I/O modules (digital input and output, and analog input) in a EtherNet/IP ring topology (EtherNet/IP is a popular industrial control protocol).

4.2 Challenges for Parsing the FCN Layer

Implementing a FCN monitor is more challenging than implementing one at the SCN level. The network of the testbed illustrated in Fig. 1 uses the Common Industrial Protocol (CIP) stack [3] for device communications at the SCN and FCN layers. This is a common industrial protocol and is representative of a wide variety of industry sectors. There a variety of differences between the FCN and SCN layers, as illustrated in Fig. 5.

One difference in the data visibility between FCN and SCN layers is that the request-and-respond communication implemented by SCN layers might miss some important data exchanges in the FCN layer: without a specific request-and-response exchange, the data of interest may not be present during the deep-packet inspection session. For example, if a specific data item request/response exchange occurs with a low frequency or under special circumstances, the data exchanged will be missed.

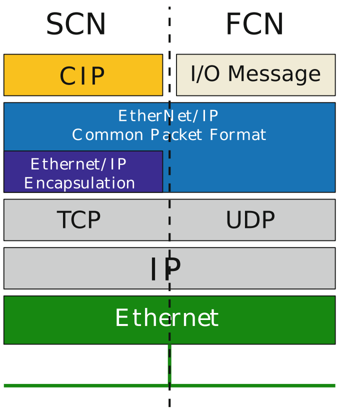


Fig. 5. Differences between the SCN and the FCN network stacks.

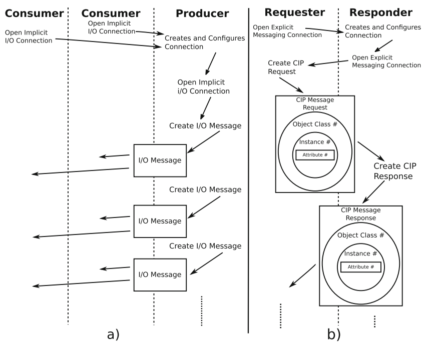
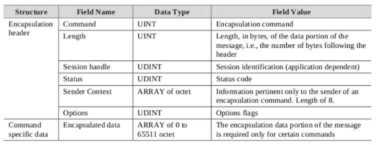


Fig. 6. (a) FCN's multicast implicit I/O connections, and (b) SCN's request/response-oriented explicit messaging connections.

In addition, at the SCN layer, devices communicate through *point-to-point* connections over the Transmission Control Protocol (TCP) and exchange **explicit** CIP messages—see Fig. 7; these explicit messages are standard and openly accessible formats defining a clear semantic of the messages exchanged between devices. As shown in Fig. 6(b), each *Messaging Connection* (MC) provides generic, multi-purpose communication paths by carrying well-known and semantically-rich explicit CIP Messages between two devices. Creating a protocol parser to extract the sensor and actuation commands in this setting is straightforward because we only need to follow the standard specification and all the data types and their interpretation can be understood by the parser.

On the other hand, at the FCN layer, devices communicate through *multicast* connections over User Datagram Protocol (UDP) and exchange **implicit** *I/O Connections* between a producer device and one or more consuming devices (See Fig. 6(a)). The semantic and structure of the data inside the I/O Message is implicitly known by the communicating devices, and is device and vendor dependent (Allen-Bradley in this deployment). In particular, these I/O Messages in the FCN layer follow a flat structure (stream of bits), of fixed size and of untyped data. Therefore we need to work with low-level data where values are exchanged without standard units of measurement, and where the protocol is not publicly available. In order to develop a parser for this layer, we require extra information provided by the electrical drawings of the equipment, illustrating how each field device (e.g. sensor or actuator) is wired to the specific modules of the PLC.



Structure	Field Name	Data Type	Field Value
Encapsulation header	Command	UINT	Encapsulation command
	Length	UINT	Length, in bytes, of the data portion of the message, i.e., the number of bytes following the header
	Session handle	UDINT	Session identification (application dependent)
	Status	UDINT	Status code
	Sender Context	ARRAY of octet	Information pertinent only to the sender of an encapsulation command. Length of 8.
	Options	UDINT	Options flags
Command specific data	Encapsulated data	ARRAY of 0 to 65511 octet	The encapsulation data portion of the message is required only for certain commands

Fig. 7. Explicit CIP message encapsulation over EtherNet/IP.

I/O Message	Signal size (bits)	# signals	Avg. Freq. (ms)
Digital Input	1	32	50
Digital Output	1	16	60
Analog Input	16	12	80

Fig. 8. Modules for each PLC.

4.3 Extracting the Semantics of FCN Data

After implementing the parser in Python, we now need to interpret the data we see in the wire. According to the electrical drawings, we found that each PLC had three modules: a digital input module (to receive on/off status reports from senors or fault alarms from devices in the field), an analog input module (to receive fine-grain information from sensors in the field such as the height of the water level in a tank, or the pH level of the water), and a digital output (to turn on/off actuators in the field). None of the PLCs in this testbed had an analog output module (analog outputs are used to control continuous variables such as the speed of a motor or the partial valve opening).

The number of signals available per module are summarized in Fig. 8. For example, a digital input module for the PLC consists of a stream of 32 bits, corresponding to each of the digital inputs signals. The *spare* channels are those not in use by the current deployment. The digital outputs are grouped in 16-bit stream (1 bit per signal), while the analog inputs are grouped in a 24-byte stream with 16 bits per signal 2's complement.

Electrical drawings of the plant tell us which specific bit (or word) in the PLC module corresponds to each signal. For example, Fig. 9 shows the electrical diagram indicating the description of each bit in the stream for a digital input module (the top part of the figure is our own illustration showing how these sensors connect to the PLC).

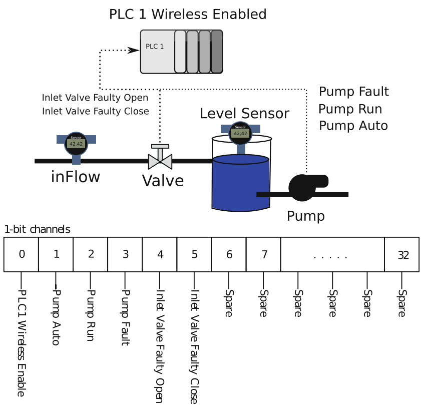


Fig. 9. Digital input module with 32 input signals (1-bit signals) for the raw water storage stage PLC.

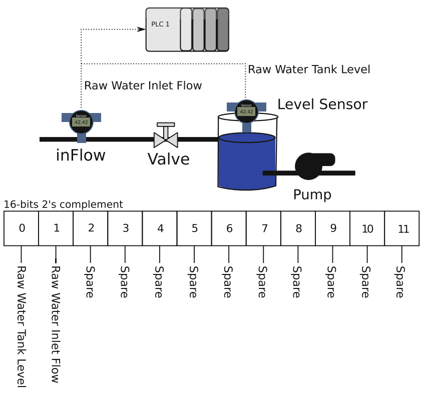


Fig. 10. Analog input module with 12 input signals (16-bits signals) for the raw water storage stage PLC.

The I/O Messages containing the analog signals are sent by the field devices to the PLC with an average frequency of 80 ms. They transport the numeric representation of the 4–20 mA analog electrical signals measured by the analog sensors and converted to their raw digital version using an Analog-Digital Converter (ADC). For example the analog inputs for the first stage of the testbed are shown in Fig. 10.

In order to scale back and forth between the 4–20 mA analog signal and the real measurement with standard units, we use Eq. (1). In this equation, *EUMax* and *EUMin* are the desired maximum and minimum limits of the specific *Engineering Unit* (e.g. millimeters (mm), *pH*, cubic meters per hour (m^3/h), etc.) to which the Raw signal is being scaled; RawMax and RawMin are the maximum and minimum possible limits for the original Raw signal. These constant values depend on the type of sensors and the physical property being measured.

$$Out = (In - RawMin) * \frac{EUMax - EUMin}{RawMax - RawMin} + EUMin \tag{1}$$

By looking at packet captures between the PLC and the field devices we found that each packet represented a specific exchange between a module in the PLC and the field devices. Therefore by simply looking at the packet payload size (32 bits for the digital input module, 16 bits for the digital output module, and 192 bits for analog inputs) we were able to identify the type of communication.

Based on this information, we developed parsers for the three types of packets for all PLCs, and a command-line interpreter (CLI) application which includes a library of attacks and a network monitoring module implementing attack-detection mechanisms. The attack modules are capable of launching diverse spoofing and bad-data-injection attacks against the sensor and actuator signals of the testbed. The attack modules can be loaded, configured, and run independently of each other, allowing to attack sensors/actuators separately. The attack modules can also be orchestrated in teams in order to force more complex behaviors over the physical process, while maintaining a normal operational profile on the HMI. The CLI application consists of 632 lines of Python [26] 2.7 code and the only external dependencies are Scapy and NetFilterQueue.

Specifically, making use of Scapy [27], we developed a new protocol parser for the Allen-Bradley proprietary I/O Messages used at the FCN layer, and for the EtherNet/IP Common Packet Format wrapper that encapsulates it. This parser allows us to sniff, in real-time, the sensor readings and actuation commands, and to inject fake packets in the network. When injecting fake data, our software calculates the data integrity checksums used by the Transport Layer protocol.

Instead of injecting fake packets crafted from scratch, our attack modules catch the original packets from the communication stream and insert fake sensing/control data, before sending the packets to their original destination. Inserting fake packets may result in race conditions when the original and the fake packet are both process by the PLC. We employed the NetFilterQueue [25] Python bindings for libnetfilter queue to redirect all the I/O Messages between PLC and the field devices to a handling queue defined on the PREROUTING table of the Linux firewall *iptables*. The queued packets can be modified using Scapy and the previously mentioned message parser, and finally released to reach their original destination e.g., PLC or field devices. Likewise, this technique allowed us to avoid disruptions on the sequence of EtherNet/IP counters, and injection of undesirable perturbations in the EtherNet/IP connections established between field devices. Our final security monitor is inserted in the EtherNet/IP ring between the PLCs and the field devices.

5 Experiments

We now illustrate how our monitor system can be used to launch and detect attacks at the FCN in the testbed. In the following experiments, the goal of the attacker is to deviate the water level in a tank as much as possible until the tank overflows, without being detected. We assume an attacker who has complete knowledge of the physical behavior of the system and can manipulate EtherNet/IP field communications or has compromised the PLC.

Our network monitoring module was setup to use a stateful CUmulative SUM (CUSUM) anomaly detection on the residuals, with a LDS model of the process. In particular, we use a mass balance equation that relates the change in the water level h with respect to the inlet water flow Q^{in} and outlet water flow Q^{out} volume of water, given by $Area\frac{dh}{dt} = Q^{in} - Q^{out}$, where $Area$ is the cross-sectional area of the base of the tank. Note that in this process the control actions for the valve and pump are On/Off. Hence, Q^{in} or Q^{out} remain constant if they are open, and zero otherwise. Using a time-discretization of 1 s, we obtain an estimated model of the form

$$\hat{h}_{k+1} = h_k + \frac{Q_k^{in} - Q_k^{out}}{Area} \tag{2}$$

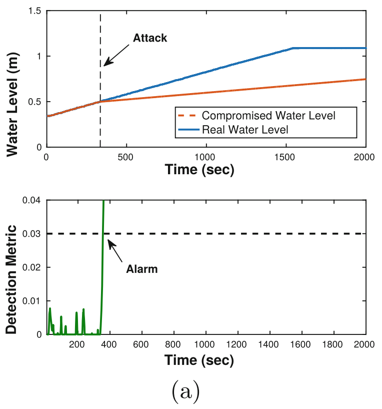
where h_k represents the received sensor measurement for the water level at time k, Q_k^{in} represents the on/off variable of the state of the inlet valve at time k, and Q_k^{out} represents the on/off variable of the state of the pump that takes water off the tank. Given these variables, we can predict the height of the tank at the next time step $\hat{h}_{k+1}$.

A residual statistic keeps track of the difference between the height of the tank received at time $k+1$ and the expected height $r_k = |h_k - \hat{h}_k|$. A cumulative sum of these errors (minus a forgetting factor δ) is then computed as part of the CUSUM anomaly detection test: $S_{k+1} = \max(0, S_k + r_k - \delta)$, see [23]. If this statistic is greater than a user-specified threshold τ (usually selected to maintain a low false alarm rate) then we raise an alarm; i.e., if $S_k > \tau$ then we send an alert to the operator.

We now show how our field-level implementation has enough visibility to detect a variety of attacks to the system.

5.1 Sensor Attack (Water Level)

We assume the adversary has gained access to the communication link between the sensor and the PLC and she is able to manipulate the sensor information as we described above. At the moment of the attack, the valve was open and the pump was off, so the water level in the tank starts increasing. This attack corresponds to attack 2 in Fig. 3a. The sensor information is used by the PLC to determine the control action; therefore, if the attacker lies and tells the PLC that the water height is increasing at a slower rate it actually is increasing, the PLC will keep the valve open and the tank overflows before the valve closes. Figure 11a illustrates how the compromised water level increases at a slower rate than the real one, and as a consequence, when the sensor information reaches 0.8 m and the PLC closes the valve, the real water is already overflowing (the height of the tank is 1 m). However, our proposed detection mechanism detects that the sensor measurement received h_k does not match the rate $\hat{h}_k$ at which the water should be increasing. The consecutive differences between h_k and $\hat{h}_k$ form the residual r_k (i.e., $r_k = |h_k - \hat{h}_k|$). Taking a CUSUM detection statistic over r_k triggers an alarm a few seconds after the attacked is launched.



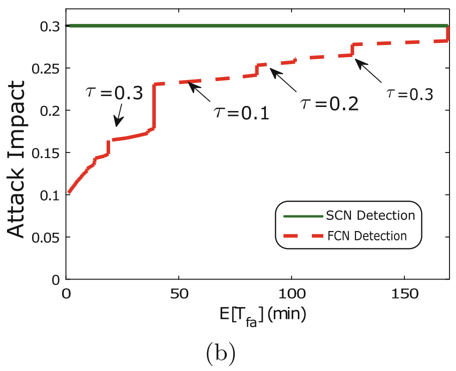
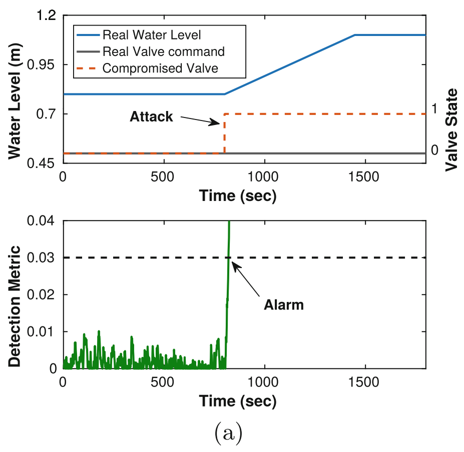


Fig. 11. (a) Sensor attack (water level). (b) Impact of a stealthy sensor attack with detection strategies in SCN and FCN.



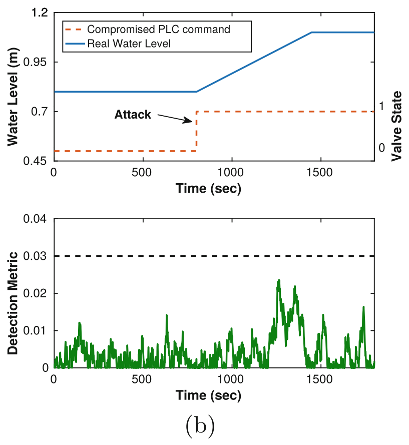


Fig. 12. (a) Actuator attack (inlet valve). (b) Controller attack (PLC/SCADA).

In this case it does not matter if the security monitor is at the FCN layer or at the SCN layer, the attack can be detected at any layer because both layers have visibility into the false sensor data. However, to illustrate the problem of relying only in supervisory networks when a PLC is compromised, we ran the same attack, but this time the PLC reported the reading the anomaly detector was expecting to the supervisory network. The anomaly detector in the field network on the other hand, was able to see the raw sensor measurements being produced and how they did not match with the control commands of having the inlet valve open and the pump extracting water closed. Motivated by the performance metric proposed by Urbina et al. [23], we compute the trade-off between attack impact and frequency of false alarms that a stealthy attacker (one that does not raise any alarms) causes to both systems: an PBAD at the

SCN, and a PBAD at the FCN (see Fig. 11b). In the figure it is clear that if the attacker wants to remain hidden, it has better chances to cause damages to the system if the defender only monitors the SCN.

5.2 Actuator Attack (Inlet Water Valve)

Now we turn our attention to attacks against the actuators, as illustrated in Fig. 3(a). In this scenario, our security monitor observes the intended control command by the PLC to the actuator, but then notices that the sensor measurement does not correspond to the intended control command.

We consider a state of the system where the water in the tank is at 0.8 m and the PLC intends to keep that level by having the intake valve closed, and the pump taking water out of the tank off. Figure 12a shows how a false actuation command to open the valve increases the height of the water in the tank. Again, our anomaly detection system detects that the predicted height (0.8 m) based on the current control commands (both the inlet valve and the pump are off) should remain constant, so the increase of the water level is detected as anomaly.

As in the previous case (sensor attack), if the only attacked device in the system is the intake valve position, then it doesn't matter if the security monitor is at the FCN layer or at the SCN layer, the attack can be detected at any layer because both layers have visibility into the truthful sensor data and notice that it does not correspond to the control commands sent to the field.

5.3 PLC Attack (RC-Detection)

In this attack the logic of the PLC is modified so that it sends a false control command, but reports that everything is fine to the SCN as in Fig. 2b. In particular, the PLC sees that the water is at the high level of 0.8 m (so it shouldn't open the intake valve); however, our change of logic in the PLC instead asks it to open the intake valve to allow more water into the tank, while reporting to the supervisory control layer that the intake valve is closed and that the water level is still 0.8 m.

As discussed before, this attack cannot be detected with a security monitor at the supervisory control layer, because it does not have visibility to the field control commands sent by the PLC. As discussed before, this attack cannot even be detected using PBAD at the FCN because there is no discrepancy between the observed control commands, and their effect on sensor measurements. Figure 12b shows how a command from a compromised PLC cannot be detected by our physics-based anomaly detection statistic as our FCN monitoring tool observes the command to open a valve, and then predictably, the height of the tank begins increasing at the appropriate rate.

To detect this attack we require a redundant control architecture in this case is illustrated in Fig. 4b. In this case the RC attack detection algorithm notices the compromised PLC sends a signal that is not authorized to send in its current state, thus detecting the compromised PLC.

5.4 Multi-loop Anomaly Detection (ML-PBAD)

The worst type of attack corresponds to the case when the sensors and actuators are attacked simultaneously. To show the generality of our ML-PBAD architecture originally presented in Fig. 4c, we implemented it in two different systems; we now present the results for the water system we have been considering.

Here we assume the adversary has compromised both the actuators and the sensors, and therefore is able to send arbitrary actuator control signals (e.g. open the intake valve when the water in the tank is 0.8 m) while at the same time, lie about the sensor readings (i.e., tell our security monitor that the water in the system is constant at 0.8 m).

Now, recall that the testbed has multiple stages controlled by various PLCs, each of them receiving different field signals from the physical process. We focus our attention on the water level of two consecutive stages with two tanks, each of which is controlled by hysteresis switched controls that depend on those levels. We want to show how attacks over sensors and actuators affect the performance of the system, and it is even possible to lead the water level to overflow.

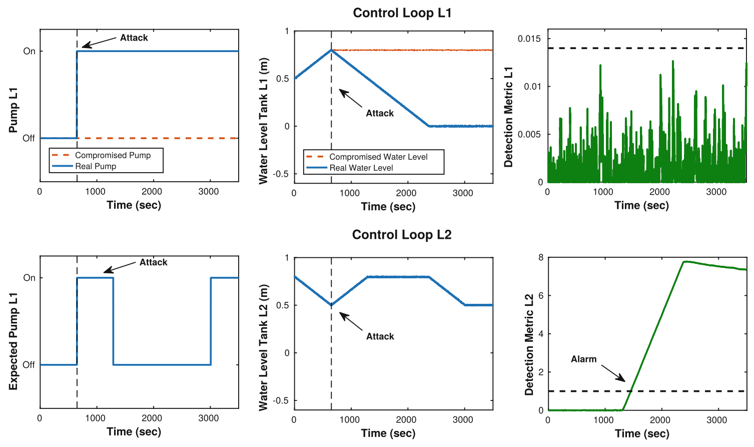


Fig. 13. Sensor and actuator attack in stage 1. The attack cannot be detected by the detection algorithm in process 1, but it can be observed in the other stage.

In particular, we assume our attacker has compromised both the pump actuator and the water level sensor in the first stage of the testbed. As result, if the attacker wants to damage the pump, it can turn on the pump directly from the actuator command. As our security monitor will not see that command, it will assume the pump is off. The attacker then can lie about the water in the system, and tell the PLC (and thus the security monitor) that the water remains at 0.8 m, while in reality the water level in tank one is decreasing.

From the point of view of the security monitor however, the pump is off, and the water level is stable (the security monitor sees the red lines in Fig. 13 (top)) and therefore, the anomaly detection statistic for stage 1 does not increase. However, from the point of view of the field security monitor in stage two of the plant monitoring the control loop of the second PLC, the water level for the second tank will appear to rise without any apparent reason, and this will raise an alarm, as illustrated in Fig. 13 (bottom).

We found that without our proposed ML-PBAD scheme, the attacker is able to raise the water level of the tank to the point of overflow (0.4 m above the setpoint) without being detected by the single loop PBAD. In contrast, the attacker can only raise the water level to 0.1 m above the setpoint without detection if ML-PBAD is used

6 Conclusions

In this paper we have presented a detailed discussion of the lack of missing trust models in previous work, and why specifically looking where to deploy physics-based anomaly detectors is of high importance. In particular we show why deploying security monitors in the field level of industrial control systems has several advantages over deploying only at the supervisory control layer.

We then implemented a field security monitor. We showed the differences between implementing a detector in the field level versus at the supervisory control layer, and then showed its effectiveness to detect more attacks than what is possible at the supervisory control layer.

Finally, by experimenting with attacks to all components of the system, we were able to identify new tools to mitigate some corner cases that cannot be addressed solely with PBAD-anomaly detection algorithms [23]. We then presented a new holistic detection architecture that covers detection of attacks not previously discussed in the literature.

A limitation of a field monitor is that if both, sensor and actuators are compromised, then an attacker can still bypass this detection. To mitigate this problem we proposed the integration from multiple field monitors at different stages in a large process. Our work in this distributed architecture improves the visibility of our system, and makes the work of an attacker who wants to remain stealthy much harder. Powerful attackers will always be able to bypass the system, but our architecture will raise the bar in the amount of effort and knowledge required by attackers to be successful.

Acknowledgements. We would like to thank SUTD for giving us access to their SWaT testbed to conduct our experiments. This material is based on research sponsored by the National Science Foundation with award number CNS-1718848, by the National Institute of Standards and Technology with award number 70NANB17H282, and by the Air Force Research Laboratory under agreement number FA8750-19-2-0010. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government.

References

1. Abrams, M., Weiss, J.: Malicious control system cyber security attack case study-Maroochy water services, Australia. The MITRE Corporation, McLean (2008)
2. Ahmed, C.M., et al.: NoisePrint: attack detection using sensor and process noise fingerprint in cyber physical systems. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 483–497. ACM (2018)
3. Brooks, P.: EtherNet/IP: industrial protocol white paper. Technical report, Rockwell Automation (2001)
4. Carcano, A., Coletta, A., Guglielmi, M., Masera, M., Fovino, I.N., Trombetta, A.: A multidimensional critical state analysis for detecting intrusions in SCADA systems. IEEE Trans. Ind. Inform. **7**(2), 179–186 (2011)
5. Cardenas, A.A., Amin, S., Lin, Z.S., Huang, Y.L., Huang, C.Y., Sastry, S.: Attacks against process control systems: risk assessment, detection, and response. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 355–366 (2011)
6. Caselli, M., Zambon, E., Amann, J., Sommer, R., Kargl, F.: Specification mining for intrusion detection in networked control systems. In: 25th USENIX Security Symposium (USENIX Security 2016), pp. 791–806 (2016)
7. Cheng, L., Tian, K., Yao, D., Sha, L., Beyah, R.A.: Checking is believing: event-aware program anomaly detection in cyber-physical systems. IEEE Trans. Dependable Secur. Comput. (2019)
8. Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., Valdes, A.: Using model-based intrusion detection for SCADA networks. In: Proceedings of the SCADA Security Scientific Symposium, vol. 46, pp. 1–12 (2007)
9. Falliere, N., Murchu, L.O., Chien, E.: W32: stuxnet dossier. White paper, symantec corp., security response (2011)
10. Gerdes, R.M., Winstead, C., Heaslip, K.: CPS: an efficiency-motivated attack against autonomous vehicular transportation. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 99–108. ACM (2013)
11. Hadžiosmanović, D., Simionato, L., Bolzoni, D., Zambon, E., Etalle, S.: N-gram against the machine: on the feasibility of the N-gram network analysis for binary protocols. In: Balzarotti, D., Stolfo, S.J., Cova, M. (eds.) RAID 2012. LNCS, vol. 7462, pp. 354–373. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33338-5_18
12. Hadžiosmanović, D., Sommer, R., Zambon, E., Hartel, P.H.: Through the eye of the PLC: semantic security monitoring for industrial processes. In: Proceedings of the 30th Annual Computer Security Applications Conference, pp. 126–135. ACM (2014)
13. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur. Priv. **9**(3), 49–51 (2011)
14. Langner, R.: To kill a centrifuge: a technical analysis of what stuxnet's creators tried to achieve. Langner Group, Arlington (2013)
15. Lee, R.M., Assante, M.J., Conway, T.: Analysis of the cyber attack on the ukrainian power grid. Technical report, SANS Industrial Control Systems, March 2016
16. Liu, Y., Ning, P., Reiter, M.K.: False data injection attacks against state estimation in electric power grids. In: Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 21–32. ACM (2009)

17. Mathur, A., Tippenhauer, N.O.: SWaT: a water treatment testbed for research and training on ICS security. In: Proceedings of Workshop on Cyber-Physical Systems for Smart Water Networks (CySWater), April 2016. https://doi.org/10.1109/CySWater.2016.7469060
18. McLaughlin, S.: CPS: stateful policy enforcement for control system device usage. In: Proceedings of the 29th Annual Computer Security Applications Conference, ACSAC 2013, pp. 109–118. ACM, New York (2013)
19. Mitchell, R., Chen, I.R.: A survey of intrusion detection techniques for cyber-physical systems. ACM Comput. Surv. **46**(4), 55:1–55:29 (2014)
20. Pasqualetti, F., Dorfler, F., Bullo, F.: Attack detection and identification in cyber-physical systems. IEEE Trans. Autom. Control **58**(11), 2715–2729 (2013)
21. Teixeira, A., Pérez, D., Sandberg, H., Johansson, K.H.: Attack models and scenarios for networked control systems. In: Proceedings of the 1st International Conference on High Confidence Networked Systems, pp. 55–64. ACM (2012)
22. Teixeira, A., Shames, I., Sandberg, H., Johansson, K.H.: Revealing stealthy attacks in control systems. In: 2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pp. 1806–1813. IEEE (2012)
23. Urbina, D., et al.: Limiting the impact of stealthy attacks on industrial control systems. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS), October 2016. https://doi.org/10.1145/2976749.2978388
24. Williams, T.J.: The purdue enterprise reference architecture. Comput. Ind. **24**(2), 141–158 (1994)
25. Python bindings for libnetfilter_queue, February 2017. https://github.com/fqrouter/python-netfilterqueue
26. Python Language: version 2.7.10, February 2017. https://docs.python.org/2/
27. Scapy Packet Manupulation Program: version 2.3.1, February 2017. http://www.secdev.org/projects/scapy/doc/