Chapter 11
Intrusion Detection in Industrial Control Systems

Edward J.M. Colbert and Steve Hutchinson

11.1 Introduction

Even if the threats, risk factors and other security metrics—which we discussed in previous chapters—are well understood and effectively mitigated, a determined adversary will have non-negligible probability of successful penetration of the ICS. In this chapter we use the word "intrusion" to refer to a broad range of processes and effects associated with the presence and actions of malicious software in an ICS. Once an intrusion has occurred, the first and necessary step for defeat and remediation of the intrusion is to detect the existence of the intrusion.

We begin this chapter by elaborating on the motivation for intrusion detection and briefly sketch the history—surprisingly long and going back to early 1980s—of intrusion detection technologies and systems (IDS). Much of the chapter's attention is on the difficult question of whether insights and approaches developed for IDSs intended for information and communications technology (ICT) can be adapted for ICSs. To answer this question, the chapter explores the modern intrusion detection techniques in ICT such as host-based techniques and network-based techniques, and the differences and relative advantages of signature-based and non-signature methods. Then, the chapter explores how such techniques may or may not apply in ICS environments. It is useful in such exploration to differentiate between early (let us say before 2010) and recent perspectives on such adaptations.

Finally, we introduce approaches based on an appreciable degree of knowledge about the process controlled by the ICS. These methods focus on monitoring the

E.J.M. Colbert (✉)
US Army Research Laboratory, Adelphi, MD, USA
e-mail: edward.j.colbert2.civ@mail.mil

S. Hutchinson
Adelphi Laboratory Center, US Army Research Laboratory, Adelphi, MD, USA

E.J.M. Colbert, A. Kott (eds.), *Cyber-security of SCADA and Other Industrial Control Systems*, Advances in Information Security 66,
DOI 10.1007/978-3-319-32125-7_11

underlying process in the control system rather than monitoring network traffic. One of the methods presented in the chapter attempts to model process variable excursions beyond their appropriate ranges using machine-learning techniques. The second method requires plant personnel input to define critical process variable limits. Semantic modeling of plant control variables is used in both methods. The chapter concludes with a detailed case study of an IDS in the context of a sample plant and its ICS.

11.2 Background

11.2.1 Motivation for Intrusion Detection Systems (IDSs) in Industrial Control Systems (ICSs)

An ideally secure computer system would not have any vulnerabilities and would not be able to be compromised at all. However, as security experts often half-jokingly say, a computer system with this level of security only exists if it can be completely isolated and never used by anyone. This illustrates a point. Even if all hardware, software, and network threats are mitigated fully, the users of the system are human and can still commit disrupting actions, intentional or not. Insider threat cannot be completely removed from any computer system with human users.

With this said, we adopt the premise that there is always a threat against a real computer system, no matter how well technical and physical vulnerabilities are removed. This is certainly true of computers that are part of Industrial Control Systems (ICSs) that are used to automate and control critical processes used in industrial settings. In fact, ICSs typically have far more technical and physical vulnerabilities than ICT systems, merely because modern ICT equipment is often designed with security in mind.

11.2.2 Early Intrusion Detection Systems

If there is always a vulnerability that can be exploited in a computer system, how does one protect that system? Early ideas (e.g. Anderson 1980) were to monitor accounting records that already existed (for the IBM System Management Facility [SMF] mainframe), and perform analytical tests against those data records for anomalous patterns. This work is often referenced as the first Intrusion Detection System (IDS). The idea is to provide anomalous pattern information as a tool for a human security monitor, with the expectation that most of the patterns flagged will not be malicious behavior. Denning (1987) later elaborated this IDS concept with a more sophisticated model and set of processes (for the same IBM SMF mainframe) developed at SRI International a few years earlier (Denning and Neumann 1985).

It is worth noting that vulnerabilities and threats to these early computer systems were quite different from those of most ICT systems used now. The systems were not openly connected to the large number of malicious actors on today's Internet, since the Internet as we know it did not exist, and global networking of computer systems was rare. The technical competence of a computer user was much higher then. Graphical User Interfaces (GUIs) and computer mice were not a common feature of these systems. Recreational computer user activity (e.g. "surfing" the net) was extremely rare, or non-existent at that time, so that external threats via the network were not a main concern. Aside from that, the types of threats and potential alerts examined then

- Attempted break-in
- Masquerading or successful break-in
- Penetration by legitimate user
- Leakage by legitimate user
- Inference by legitimate user
- Trojan horse
- Virus
- Denial of Service

are very similar to those considered today, even though the threat model used on this early system was much less complex.

11.2.3 Evolution from Early to Modern IDSs

As methods for ICT IDSs developed further, additional techniques were implemented to flag alerts. Most of the threats and alerts pertained to the confidentiality of (assumed) private information existing inside the computer systems or networks. Security is often explained using the CIA "triad": Confidentiality, Integrity, and Availability. When one learns of a security breach in an ICT system, the first thought might be to determine if bank account or credit card information or sensitive emails were stolen. Integrity and Availability are important on an ICT system; however, ICT IDS systems were designed primarily to minimize confidentiality. The intent of most malicious actors breaking into ICT systems is different from that of those breaking into ICS systems. ICS systems control a process and the ICS adversary's intent is to undermine that process, i.e. the availability of the system.

Home personal computers (PCs) in the early 1980s had very little in terms of technical or network security. The best method of protecting PC systems was to provide physical security: lock the room, remove the hard drive, or even install a physical lock on the base system power supply. Some early PC users were able to install password authentication. Even so, the system was still usually extremely easy to break if one had physical access, and PC usage without physical access was very uncommon. Physical security was king. ICSs are similar to these early PCs. It is common for ICSs to use equipment that was manufactured in the 1980s,

or even before, and to use physical security as the primary line of defense. Why? The reason is because the older equipment still functions well and the main emphasis in ICSs is Availability.

Since then, home PCs have evolved into much more complex systems with much more storage and computing capacity. Network connectivity in home PCs and mobile devices is now commonplace. As a result, extensive information access is possible, often even from a single break-in. On the other hand, ICS systems have not changed much since the1980s. Even so, there is now a much more significant interest in connecting them with intranet and Internet, so that operators and managers can access them remotely. Once remote network connectivity is established to these inherently insecurity computer systems, Physical Security is no longer king. Break-ins can and will occur via the global network, leaving ICSs (and their processes) extremely vulnerable.

As mentioned, IDSs for ICT networks have become very popular; especially for identifying the signatures of many pieces of known malicious code (e.g. SNORT rules). Some IDSs utilize model-base anomaly detectors, such as those original proposed by Anderson (1980) and Denning (1987).

An important question is: can we really just transfer the methods developed and implemented for early and current ICT systems to ICSs? Most researchers (e.g., see review by Zhu and Sastry 2010) attest that the answer is no. ICS traffic is much different, ICS component security is much different, and as we have described above, the intent of the intruder is likely much different—to disrupt the availability of the process instead of to compromise the confidentiality of information.

It is important to keep these differences in mind while considering techniques and methods for Intrusion Detection in ICSs.

For the purposes of brevity, we abbreviate the term "Intrusion Detection" as "ID" in the remainder of this chapter.

11.3 Modern Intrusion Detection Techniques

We begin with a brief review on ID and IDSs to provide an overview of their history since the early 1980s, and a crude taxonomy.

11.3.1 Host-Based Intrusion Detection Systems (HIDS)

IDSs described by Anderson and Denning are host-based systems (Host-based Intrusion Detection Systems, or HIDs) for the IBM SMF mainframe systems. System data used in the analysis was internal accounting audit trail data from users of the system. Host-based systems still exist in present day ICT networks, and these systems now process much larger amounts of data. Modern operating system audit trails include both general user accounting information such as login times, system

reboot events, hardware access records, and specific security information such as login password failures and firewall denial events. Third-party security software such as anti-virus, personal firewalls and browsers can supplement the operating system information and provide an extensive set of current information for HIDs. This information can also be packaged and distributed to a central processing facility to provide situational awareness of the network nodes.

As mentioned, modern ICS equipment does not normally fall in the same category as computer systems in modern-day ICT networks. ICS equipment is not typically designed with security logging and processing in mind. It does not usually run standard operating systems used in ICT desktops and servers. The device operating software is often vendor specific and primitive. The devices are not refreshed as often as ICT network devices, and may very well be so old that they are no longer supported by the vendor. Upgrades or modifications of the operating software may not be available. Vendors will likely not support any security upgrades to unsupported products. While HIDS systems in general are common for ICT systems, they are not generally used on ICS hardware since it is not typically suitable for logging or monitoring processes.

11.3.2 Network-Based Intrusion Detection Systems (NIDS)

Network-based IDSs are a network device that collects network traffic directly from the network, often from a central point such as a router or switch. Data from multiple network sensors can be aggregated into a central processing engine, or processing may occur on the collection machine itself. For NIDS, useful audit data must be extracted from raw, unformatted network packet data itself rather than from pre-formatted audit log information used by HIDs.

In general, two methods are used for alerting. The network traffic can be scanned for known malicious code with specific bit signatures, which is known as signature-based detection. The network traffic can also be analyzed for unsatisfactory traffic or behavior patterns; either patterns that are anomalous to a previously established traffic or behavior model, or specific traffic patterns that display non-conformity to standards, e.g. violations of specific communication protocols.

11.3.2.1 Signature-Based Intrusion Detection Methods

The most common signature-base security tool used in IDSs is Snort, which was developed by Marty Roesch in 1998 and is currently maintained by Cisco at www.snort.org. Signature-based detection methods are sometimes referred to as misuse-based detection or knowledge-based detection, since knowledge of the software threats under search must have already been gained. Signature-based detection is very commonly used by anti-virus scanners as the *de facto* security measure on ICT systems. The intent of this method is often to determine whether software is safe to

import into a system, or whether a system hard drive is safe from malicious code. Likewise, it can often be used as the primary method used in NIDs to determine whether malicious traffic is present on the network. A drawback is that adversaries can easily create dynamically changing code for known malicious software so that the code escapes signature-based detection. Unknown malicious code also escapes detection. Thus, while widely used, signature-based detection methods are not foolproof. As far as security tools go, though, they are very accurate in providing knowledge of specific detectible intrusions of known malicious code. If one considers an intrusion alarm as a "positive," the rate of false alarms, or false positives, is generally low for signature-based detectors.

11.3.2.2 Non-signature-Based Intrusion Detection Methods

Non-signature-based intrusion methods are more difficult to evaluate for implementation since there are a large number of different methods used, and the rate of false positives is often higher than that of signature-based methods. If false positive rates are too high, it becomes difficult or impossible for a security analyst to monitor the system (see, e.g., Axelsson 2000). Calibration processes for the algorithm can be complex as well, potentially thwarting acceptance and implementation by system owners. When the calibration method includes developing a model for comparison (anomaly-based detection), a significant benefit over signature-based methods can be achieved in that *unknown* malicious events can be detected. Anomaly-based detection is sometimes referred to as behavior-based detection when the baseline reference is behavior based, or model-based detection when the baseline is model-based.

Another non-signature-based detection method relies on searching for nonconformity or deviations from accepted industry guidelines, such as protocol standards. IEEE and IETF network communication protocols are often robust, but improper use of the protocols can produce malicious or covert activity. Vendor products do not typically enforce protocol standards strictly or uniformly, allowing adversaries to create and transmit malicious packets. The ability to detect nonconformities requires intensive deep inspection of network traffic. An accurate reference model of both conforming and non-conforming aspects or patterns of the usage of the protocol is also required. This detection method is known as specification-based detection or stateful protocol analysis.

11.3.2.3 Methods Used in Practice

While the taxonomy of modern intrusion detection techniques just described is rough and simplistic, it does show that IDS methods have expanded to include much more data and analytics since the early 1980s. As noted, the primary market for IDSs has been for ICT systems, either networked desktop or server HIDS running standard computer operating systems, or NIDS connected directly to ICT networks. A more

comprehensive review and taxonomy of modern IDS techniques and can be found in Liao et al. (2013). In practice, a commercial HIDS or NIDS will often employ many different intrusion detection and alerting methods, including signature-based, anomaly-based, and specification-based methods. For example, the U.S. Army Research Laboratory has designed a network-based intrusion detection framework, Interrogator, which utilizes an array of different detection methods in its network sensors, and a library of different analytical methods for alerting at its central repository (Long 2004).

11.4 Intrusion Detection in ICSs

In this section, we discuss in detail how the general ID methods for ICT systems have been adapted for ICSs. In the previous sections, we described the original concept and some example development of modern intrusion detection for ICT technologies. Although ICS system components are much different from those of ICT systems, many of the same ideas have been brought forward for ICS IDSs.

11.4.1 Anatomy of An Industrial Control System

Before proceeding to discuss ICS IDS systems, we need to understand how a typical ICS "network" is used.

In Fig. 11.1 we show a rough sketch of a simple ICS (see Chap. 2 for additional information on ICS components). This control system has two Programmable Logic Controllers (PLCs), each of which are connected (upper panel) to a standard ICT device network with a few Workstations. The workstations typically run Microsoft Windows or Linux, as in a standard Enterprise network. In the diagram, this network is annotated as "Primary Bus." The traffic on this network is usually IP packet-based, but parts of it could be hard-wired as a set of serial lines.

Downward from the PLCs are Secondary Buses that control field devices, such as boilers, electronic lighting, and packaging units. While these buses or networks may be IP packet-based, they are usually simple hard-wired cables with specialized voltage or current control needed to run the field devices. In other words, they are not meant to have a standard network communication protocol such as TCP/IP.

Also notice that most of the equipment is NOT computer servers, network switches, or routers, such as you might find in an ICT network. Even the workstations connected to the Primary Bus are doing atypical work. They are not meant to be connected to the Internet to browse the web. They are specifically configured to only perform their function in the ICS. There is often little interest in following security measures such as installing anti-virus or keeping the operating system up to date because, ideally, the systems are not supposed to be accessed from the outside, and are not supposed to access the outside. The field devices and PLCs do not run standard operating systems and most likely will not be modified to do so.

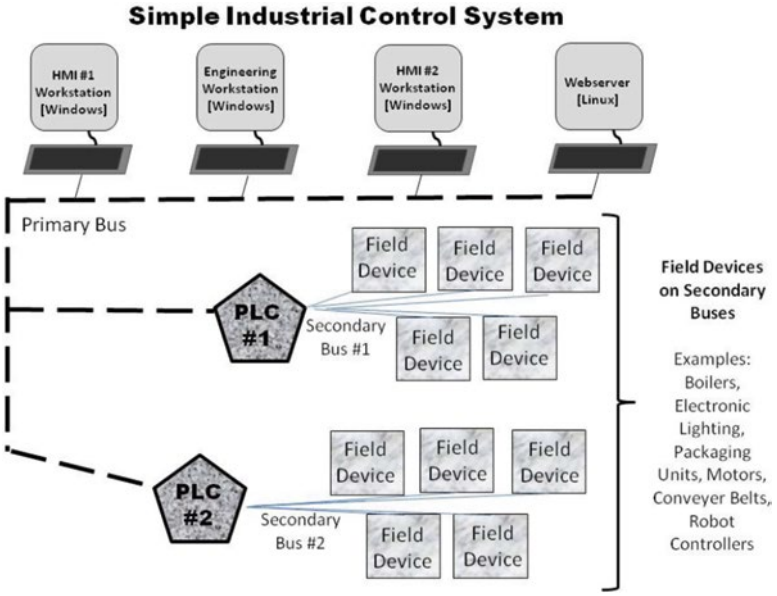

Fig. 11.1 Sample of a simple ICS

11.4.2 Host-Based Intrusion Detection Systems (HIDS) in ICSs

HIDS were developed for standard ICT computer systems, and can certainly be used on normal computer workstations and other similar system on the Primary Bus, or on the corporate network. It would be necessary to ensure that the availability of the workstations would not be affected by the operation of the HIDS or by any central network monitoring system collecting HIDS data over the Primary Bus.

HIDS are rarely if ever able to run on the PLCs or the field devices. First, the device firmware was never intended to run arbitrary software, and second, the devices' processing CPU, memory, and communication links were never designed to accommodate the additional burden. The idea of converting these devices to accommodate a HIDS is tantamount to the idea of replacing and re-testing all of the field hardware devices, which would generate tremendous resistance in an operational environment of an ICS since it implies lack of availability and implies risk of future inoperability.

While future ICS component design may accommodate embedded HIDS-like software or features, these features are not generally used in current ICS devices.

11.4.3 Network-Based Intrusion Detection Systems (NIDS) in ICSs

If we follow the analogous ICT IDS taxonomy, NIDS would be the logical type of IDS to use on ICSs. However, there is the question: What is meant by "network" for an ICS? In our simple ICS network in Fig. 11.1, we have two generally different types of networks (primary and secondary buses). The primary bus is networked (perhaps IP-based), or serial. However, the devices in the secondary buses are arbitrarily diverse. The secondary buses of a complex ICS could use hundreds or thousands of different vendor-proprietary methods of communication.

Network communication protocols in ICSs seldom if ever use any authentication or encryption, which is a critical difference from protocols in ICT networks.

Thus, NIDS methods developed for ICT systems already face a serious challenge if they are to be ported to ICSs. In our simple ICS architecture in Fig. 11.1, the Primary Bus is really the only network that would make sense to apply a NIDS. The Secondary Bus components may not be globally addressable as IP network components are. They may consist of a large number of primitive copper cables, similar to the electrical cords you might find throughout your house. While it is certainly possible that ICT network protocols are used in unique parts of the secondary bus for specific controls, the ICS "network" is uniquely different from an ICT network. Humans may also serve as an essential component in the ICS "network" as they are often a vital component of the underlying ICS process.

We assume throughout the rest of this chapter that the types of IDS systems developed for ICT networks used in ICSs will be NIDS (not HIDS), and those NIDS will be use on the Primary Bus. This ICS IDSs would monitor network traffic between control workstations and ICS hardware devices such as PLCs or RTUs (see Chap. 2). Hereafter the term "ICS network" will imply the Primary Bus network, unless otherwise stated.

11.4.3.1 Signature-Based Intrusion Detection Methods in ICSs

Signature-based ID methods usually aim to find a known bit pattern in network traffic for documented malicious code. Snort "rules" meant for ICT network traffic can easily be ported and used on ICS networks. Since the communication protocols are different on ICS networks, some modification are in order.

Starting approximately 2008, Digital Bond began a DHS-sponsored program called Quickdraw to generate ideas on how to generate security log events for PLCs (see http://www.digitalbond.com/ and http://www.digitalbond.com/tools/quickdraw/). These ideas were developed into rules and pre-processors for Snort signature detection. IDS signatures for ICS protocols BACnet, DNP3, Modbus, Modicon, Niagara Fox, and Siemens S7 became available for anyone wishing to use Snort rules for ICS IDSs.

Digital Bond Snort modifications for IDSs have been used extensively for signature detection. However, signature-based methods have been argued to be far from sufficient in protecting IDS protocol traffic. Due to the inherent lack of authentication and encryption in ICS protocols, unauthorized network access to devices is possible without using known malware. Relatively complex and poorly documented (i.e., unknown) attacks occur on ICSs and these attacks can easily evade signature-based detection methods.

As a result, ICS security researchers often favor a combination of signature-based ID methods and non-signature based ID methods when designing efficient IDSs for ICSs (see, e.g. Verba and Milvich 2008).

11.4.3.2 Non-Signature-Based Intrusion Detection Methods in ICSs

Early Examples (Before 2010)

At this point, we have converged to the ID methods category that seems to be the best fit for ICS networks—non-signature based ID methods on ICS Primary Bus traffic (see Fig. 11.1). Much effort has been devoted to developing efficient non-signature-based ID systems for ICS in the last 10–15 years.

Most of the early works (published before 2010) described or implemented anomaly-based IDSs, with the intention of providing better security protection for inherently highly-vulnerable ICS systems. Physical security methods used previously were no longer sufficient for protecting systems once they are connected to external networks. Some example methods of non-signature based ID methods for ICSs are given below to illustrate progress in solving the issues.

A biologically inspired heuristic model based on ant colony clustering was shown to be feasible for precise clustering and thus accurate, unsupervised anomaly detection (Tsang and Kwong 2005). This model was subjected to some testing with ICT-based IDS attack data, but mostly provided momentum for follow-up studies on anomaly-based IDS methods for ICSs. Normal traffic in ICS networks should consist of only a few regular requests and responses, and the volume should be much lower than that in ICT systems. For example, a PLC likely receives periodic information requests on short regular intervals from the Human Machine Interface (HMI), and other workstations in the ICS produce similarly short periodic network traffic. Occasionally the human running the HMI may make manual requests or changes to field system device variables, but most of the network traffic is very repetitive and easily modeled.

Idaho National Laboratory (INL) funded an ID study that successfully used empirical nonparametric modeling of predetermined network features to compare with current network features, and probabilistically predicted anomalous system activity (Yang et al. 2006). Again, some lab testing was performed against this model-based anomaly ID method, but a full implementation and verification was not performed. The Yang et al. study showed that network anomaly detection had merit for ICSs.

A more complete set of model-based anomaly algorithms was tested at Sandia National Laboratories by Cheung et al. (2007). A Modbus protocol-level model, a model of expected communication patterns, and a learning-based model for system changes were used in the test. Curiously, Snort was utilized to flag the complement of the models, contrary to the normal expectation that Snort is used for signature-detection. All of these alerting methods, together with the Digital Bond Snort rules, were used on the EMERALD ID and correlation framework in the testbed experiments. This work demonstrated that a multi-algorithmic ID implementation not only seems mathematically promising, but functions very well in an real (testbed) environment. Verba and Milvich (2008) provided some further guidance based on their experience with ICS IDS systems at Idaho National Labs. They also describe a multi-algorithmic ID method in which intelligent packet inspection, tailored traffic flow analysis, and unique packet tampering detection are used to provide the much higher level of granularity needed for ICS IDSs. Their conclusion is that multiple methods should be used simultaneously to provide accurate alerting, including signature-based methods.

A solid implementation of an ICS IDS was eventually built and evaluated by Oman and Phillips (2008). This method was based on having all of the ICS device functionality and their configurations documented precisely so that failed logins, configuration changes, and non-compliant network traffic would provide accurate alerts. Testbed experiments showed that the method works well. In practice, however, obtaining and maintaining 100 % accurate device information may be difficult to implement.

These methods and other early (pre-2010) research on ID for ICSs revealed the following key ideas:

- Methods used for ICT networks do not simply apply to ICS networks in the same manner
- Signature-based methods are useful, but cannot be trusted by themselves
- A large number of non-signature-based methods are feasible, although since an operational system is being examined, their unsuitably high rate of false alarm is of concern for implementation
- Using multiple methods simultaneously is beneficial, especially when using both signature-based and non-signature-based methods simultaneously
- Performing real verification testing is very challenging since operational ICSs are not available for experimentation

Recent Examples (2010 or After)

While our cutoff date of 2010 for "recent" methods is somewhat arbitrary, it reasonably marks the time when ICS ID techniques began to diverge from their ICT Enterprise "parents" and develop unique and useful methodologies customized for ICSs. The idea of using multiple methods simultaneously continued, eventually even including unique multi-method prototyping for some less common communication protocols such as IEC 60870-5-104 (Yang et al. 2013). This author utilizes signature-based and model-based techniques together. Full implementation and

testing of the ICS ID techniques has not been easy to come by, most likely because few people had testbeds to perform such tests, and operational systems were not available for experimental testing.

A neural network approach for ID was proposed and tested by Gao et al. (2010) at the Mississippi State University. Promising results were shown for most attacks, although the false alarm rate was still high enough to be worrisome. Reply attacks were not well detected, with a quoted testbed accuracy of only 12 %.

Automatic machine learning gave rise to the concept of state analysis, starting with a description of a signature-based and state-analysis system using a special rule language for MODBUS and DNP3 (Fovino et al. 2010), designed to express critical states in the system. At this point, the focus of IDSs started to change from "detecting intruders, or events signifying intruders," which would seem to be of concern for ICT systems, to "detecting changes to the underlying process, affecting process availability." This was a significant step forward as this is in fact one of the main difference between ICT and ICS system security. Fovino's embryonic work was more fully developed by Carcano et al. (2011), who tested a prototype system with a more elaborate Critical State Analysis engine and a State Proximity indicator. A multidimensional metric provided the parametric measure of the distance between a given state and any of the defined critical states. As Carcano et al. mention, the processes and critical states in an ICS system are generally well-known and limited in complexity, which is not normally the case for ICT systems. Goldenberg and Wool (2013) use a somewhat similar approach, modeling the current and critical states with Deterministic Finite Automation (DFA) techniques. These authors find that a multiple-DFA method would be superior to the single-DFA method they describe, but leave verification for future work.

A key observation for these "recent" ICS ID techniques is that they are attempting to measure the actual process values as much as possible from the network traffic, and determine anomalous behavior as disturbances in the process.

An n-gram anomaly detection technique described in Tylman (2013) shows that ICT IDS tools can still be used reliably. They developed a Snort pre-processor to detect anomalous communication between devices using the MODBUS RTU protocol. While this technique only applies to a unique scenario (MODBUS communication, which we already widely confront in ICS security), it does demonstrate that implementation does not necessarily mean building new tools from scratch.

Oman and Phillips (2008) described a full implantation of an ID system based on automated gathering, logging, and comparison of RTU device settings. Device settings are stored in XML format and comparisons are made with on-line monitor data. This is one of the most empirical based methods that has been described, and again, applies to a unique scenario (specific RTU devices). One could utilize this method with any ICS hardware configuration, provided one could exactly account for all device configurations and all possible anomalous states for those configurations.

The following research ideas have been established from these "recent" ID methods for ICSs:

- Again, using multiple methods simultaneously is beneficial, especially when using both signature-based and non-signature-based methods simultaneously

- Being able to model the state of the system (or underlying process) is very important, especially since availability of that process is of upmost importance
- Being able to model critical states and proximity to those critical states with various mathematical techniques seems to be a very promising way forward
- While ideas and prototypes are useful, there is not sufficient real testing and verification of many of the published methods
- Detailed and complete accounting of all specific hardware configurations in one's unique ICS provides useful and perhaps vital knowledge for ID

11.5 Process-Oriented Intrusion Detection

11.5.1 Overview

The design intent of an ICS is intended to (1) establish appropriate process values to produce desired output and (2) to allow operators to observe aspects of the plant to assure proper operation and safety and quality conditions. The sole purpose and only capability of ICS network traffic control messages is to support the synchronization of the PLC registers and to provide a local, HMI-side copy of these registers, to effect control of the plant processes. ICT network traffic has a much wider variety of uses, but is not generally used for process control. While both ICS and ICT computers have registers, only an ICS network can change and read register values. Register values directly affect process parameters and hence, the process. Since ICS security is ultimately for safeguarding the process variables and not the network traffic itself, process-oriented designs for monitoring and ID became of interest.

In this section, we discuss two current ID methods that focus on monitoring the underlying process in the control system rather than monitoring network traffic. The first method (Hadžiosmanović et al. 2014) attempts to model process variable excursions beyond their appropriate ranges using machine-learning techniques. The second method, which is based on ongoing ICS research at the Army Research Laboratory, requires plant personnel input to define critical process variable limits. Semantic modeling of plant control variables is used in both methods.

Semantic Security Monitoring (SSM) uses analysis of control-bus traffic messages to construct a 3rd copy of the plant-PLC registers for a new purpose: to detect events that suggest that plant operations may be out of specification, out of compliance, or out of a desired safety range. These events form the basis for a cyber-security monitoring capability. The change in emphasis is necessary; ICS networks are intended to control plant processes to produce quality output. Input sensors are queried at rates and with precision sufficient to accomplish control to maintain quality output. These rates, precision, and monitored parameters may not be appropriate or sufficient for security and safety monitoring operations.

11.5.1.1 Semantic Security Modeling from Network Traffic Data

Hadžiosmanović et al. (2014) describe a novel network monitoring approach that utilizes process semantics by (1) extracting the value of process variables from network traffic, (2) characterizing types of variables based on the behavior of time series, and (3) modeling and monitoring the regularity of variable values over time.

Their prototype system measures MODBUS traffic using scripts written in the policy language of the Bro Intrusion Detection Platform (www.bro.org) and custom C++ code. Data characterization is achieved by using heuristic algorithms to compare project file information with the network traffic information. Their results show good matches (>90 % matched) for constant process variables, but poor matches (20-70 % matched) for attribute and continuous variables. This information is used to create a "shadow" memory map of the ICS process variables.

During a training period, deviations are measured, and a rolling forecasting procedure is used to cross validate the model. A control-limits model and/or an autoregression model are then used to model all of the process variables, providing the basis for alerting.

Approximately 98 % of the process control variables used in real-world plans are reliably monitored by this process (Hadžiosmanović et al. 2014). The remaining 2 % of the variables remain challenging to model with this approach.

11.5.1.2 ARL Collaborative Modeling using SME Input, Network Traffic Data, and Process Monitoring Data

The novel approach by Hadziosmanovic et al. demonstrates that process variables can successfully be modeled for ID. However, as they mention, additional work is needed if all of the process variables are to be monitored reliably. Our ICS ID research at the Army Research Laboratory (ARL) is based on the assumption that all of the process variables do not need to monitored for alerting. Rather, there are critical process variables that need to be monitored for alerting, but abnormal values of the remaining variables are not significant enough to harm underlying plant process. We argue that identifying the critical values and determining the allowed ranges of those critical values is extremely difficult if only network traffic data is used. We use a collaborative modeling approach which uses plant operator or plant Subject Matter Expert (SME) input and out-of-band (OOB) sensor data in addition to data from network packets.

In Table 11.1, we describe general differences between the ARL Collaborative method and SSM model of Hadziosmanovic et al.

Our model recognizes that, just as in ICT ID, reference information from plant sensors, configurations, semantics, and policies (acceptable security/safety value ranges) must be captured, maintained, shared, and made available to the security/safety monitoring analysts in timely, orderly, and priority-relevant means to enhance decision-making. However, it also recognizes that ICS process sampling methods and process control methods (e.g. MODBUS) were never intended to feed security/safety analyses. Thus, as stated earlier, many process parameters seen in network

Table 11.1 General differences between the Hadziosmanovic et al. SSM model and the ARL collaborative model

Step	SSM method (Hadziosmanovic et al.)	Collaborative method (ARL)
Identify variables	• Extract values of process variables from network traffic	• Utilize critical process variable information specifically identified and described by plant process engineer (SMEs)
		• Assemble and update security and safety-relevant models based on SME input
Characterize process	• Characterize types of variables based upon observed behavior of time series (Train Model)	• Identify units, possible ranges, typical ranges, change characteristics, required sampling rates, and extreme-value conditions based on SME input
		• Refine model characterization using network traffic data
Model and alert	• Model the regularity of variable values over time using machine learning approaches to signal anomalous behavior, or significant drift/divergence of the process	• Refine model using network traffic data and OOB data
		• Alert based on excursions of critical process variables beyond SME-provided input

traffic may not be relevant, or may not be sampled at sufficient rate or fidelity. Moreover, there may be other process variables that are indeed critical, but they are not represented in network traffic, i.e. they are out of band. In this case, independent sensing of these parameters would be needed to create sufficient uplift in timeliness, accuracy, and relevance to the security/safety monitoring mission. In the ARL model, the SME defines the critical security model variables based on his knowledge and analysis of the plant processes, and the IDS security engineer implements the appropriate security model.

We refer to this model as "collaborative" since the security engineer utilizes human input from the plant operator/SME for constructing the IDS security model.

11.5.2 ARL Collaborative Intrusion Detection: A Case Study of a Sample Plant

In this section, we describe a case study implementation of the ARL Collaborative ID model. We also illustrate specific operational scenarios in which the collaborative model offers significant advantages over other methods. We end with a description of an effective alerting infrastructure which employs three ID methods, the first of which is the collaborative method outlined here.

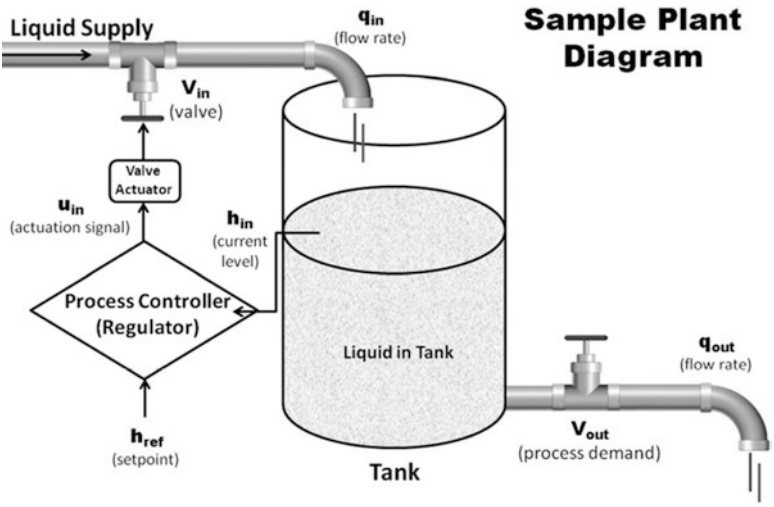

Fig. 11.2 Diagram of a sample plant

11.5.2.1 Background: Description of a Plant

Physical Plant Model

We show a plant diagram of our sample case study plant in Fig. 11.2. The plant uses a process controller (regulator) to maintain liquid level in a tank to the level value prescribed by a set point parameter entered from the HMI.

Output flow, represented by $\mathbf{q}_{\text{out}}$, reflects typical outflow to the process. A regulator monitors the current liquid level $\mathbf{h}_{\text{in}}$ in the tank, and adjusts the actuation signal $\mathbf{u}_{\text{in}}$ to an inflow supply valve $\mathbf{V}_{\text{in}}$ to maintain the level at the set point value $\mathbf{h}_{\text{ref}}$. The regulator design is intended to maintain the liquid level during anticipated process loading.

Much of the process control for this sample plant is performed by the process controller, or regulator. According to Astrom (2002) more than 95 % of regulators in control loops are of the PID (Proportional-Integral-Differential) or PI (Proportional-Integral) type. In PID type regulator, a sensor measures the process variable and the regulator compares that value with the set point value to provide the current error as a function of time [*error(t)*]. A proportional term provides a contribution to change the output (actuation) directly, according to the magnitude and direction of the error. An integral term accumulates a weighted sum of all past error values and is usually needed to allow for convergence of the process value to the setpoint value. A derivative term can improve stability and reduce oscillatory behavior by responding more to increased changes in the error value. The actuator output $\mathbf{u}_{\text{in}}$ is a weighted average of these three terms:

$$\text{Actuator Output } \mathbf{u_{in}} = W_{proportional} * error(t) + W_{integral} * \int error(t)\,dt + W_{derivative} * \frac{d[error(t)]}{dt}$$

Most plants encounter unanticipated load changes called disturbances that often present as step changes in the output load. Implementation of this load in the actual plant process can reveal previously unknown issues which can have an impact on plant monitoring data.

Implementation: Electronic Plant Model

For this case study, we implement the plant model (Fig. 11.2) by replacing the fluid mechanic components by electronic counterparts. A diagram illustrating our electronic implementation for the sample plant is shown in Fig. 11.3.

In the electronic implementation, electric current flow represents the liquid supply flow $\mathbf{q_{in}}$ to the tank. The reservoir or tank is equivalent to a capacitor. Valves $\mathbf{V_{in}}$ and $\mathbf{V_{out}}$ (or $\mathbf{a_{out}}$) control flow rates are modeled using a potentiometer and/or a bipolar-junction transistor operating in its linear range.

In our electronic implementation, the regulator is replaced by an Arduino computer programmed to function as a PID process controller. Digital signals are converted to

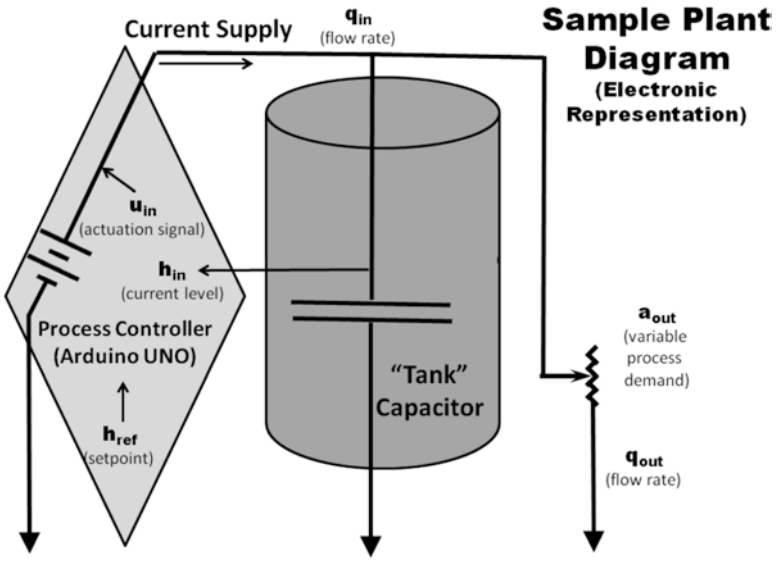

Fig. 11.3 Electronic representation of sample plant

continuous current flow by integrating the digital signal into a resistor-capacitor circuit. The time-based behavior of the physical plant can then be matched in the electrical model system. Thus, an electronic PID controller will appropriately regulate the electronic circuits in the electronic plant model.

Plant Control Network

The network diagram of the plant control network in Fig. 11.4 shows a PLC (PID controller), an HMI, a network traffic monitor (ICS Security monitor), and a high-speed sensor (Independent LEVEL sensor) that monitors the tank level.

Human Machine Interface (HMI)

In this implementation, MODBUS TCP messages are sent from the HMI to the PLC over the control network at regular polling-rate intervals of approximately one second. The HMI also affords monitoring of PLC variables representing the state of the plant process. There are roughly 10 different MODBUS commands which are limited to reading one or more register values, and setting one or more register values. Some commands pertain to 'coils' which are a binary type of register, which can take values of ON or OFF. Some PLC registers are used as inputs into its control algorithms. For example, sensor values obtained from interrogation by the PLC may be made available via MODBUS queries using a specific register. The HMI can also be used to write specific control parameters to the PLC, using MODBUS commands over the network.

The function of the HMI is to periodically query the PLC to obtain and often display important PLC control register values that are indicative of important process variables. The HMI output is then monitored regularly (usually visually) by a human plant operator.

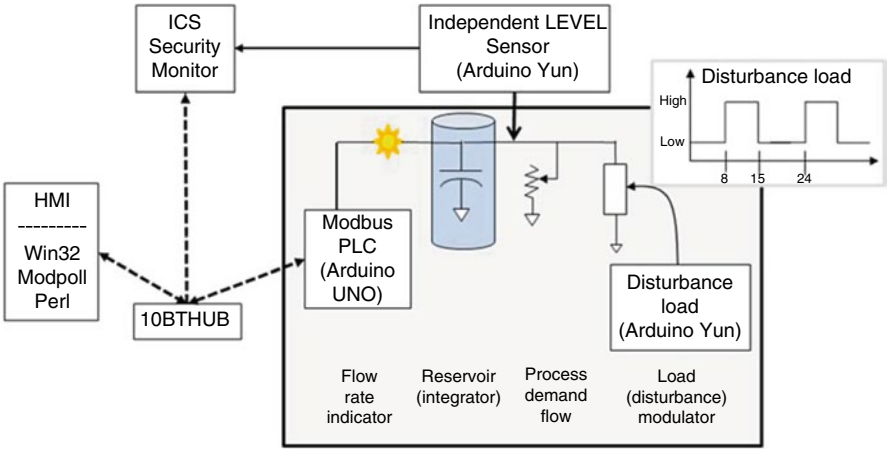

Fig. 11.4 Network diagram of electronic implementation of sample plant

PLC/Regulator (PID Controller)

The Arduino UNO PID controller regulates incoming supply flow $\mathbf{q_{in}}$. (The Ardino UNO is labeled "Modbus PLC (Arduino UNO)" in Fig. 11.4). The controller supports MODBUS TCP communications over a wired Ethernet network. Register and coil number assignments are made arbitrarily by plant engineers, which we will see is an important observation in Section 11.5.2.2.1.

The Arduino PLC implementation diagram for our plant process is shown in Fig. 11.5. We show the use of one coil register (C[1]) as an on/off or RUN/STOP switch which must be set by the HMI controller interface.

During operation, a level sensor (10-bit analog input A/D converter) provides current level values $\mathbf{h_{in}}$ to the controller which update register R[2]. The desired set point value $\mathbf{h_{ref}}$ is stored in R[1] using a MODBUS write operation from the HMI. The PID algorithm evaluates the error (difference between set point and current level) and provides a copy of this value to update R[9] as the current error value. The PID algorithm calculates a new output actuation signal value $\mathbf{u_{in}}$, and sends the actuation signal to an output actuator (using a pulse-width modulated signal) and provides the corresponding numeric output value to R[3].

In this manner, MODBUS read-register commands can obtain values from all PID parameters including set point, current level, current error, current output, as well as other parameters exposed to the MODBUS registers.

Network Traffic Monitor

To implement network monitoring for intrusion detection, a network traffic sensor is added to the plant control network to enable passive monitoring of all MODBUS TCP traffic. This sensor is labeled "ICS Security monitor" in Fig. 11.4.

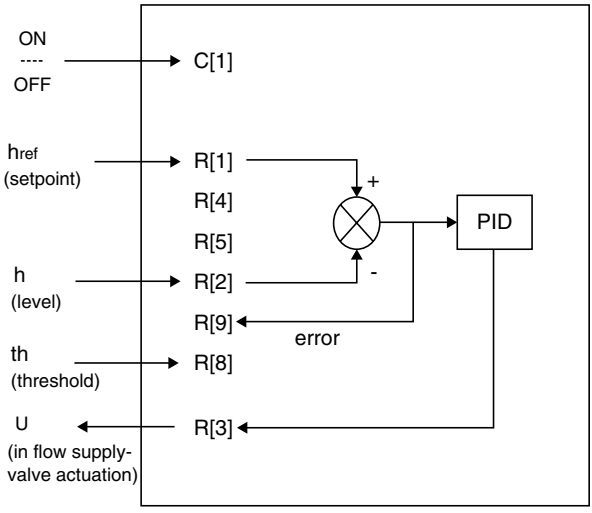

Fig. 11.5 Arduino Uno PLC implementation diagram

Independent High-Speed Sensor

We also add a high-frequency sensor to the plant which also monitors critical process parameters. This sensor is labeled "Independent LEVEL sensor" in Fig. 11.4.

This high-speed sensor serves two purposes. First, it helps the SME to better characterize some typical behaviors that are not critical, but may appear as such when sampled at the lower rates used by the control system. Second, the sensor provides an independent (out of band) physical measurement that can validate data from the network traffic monitor.

For our specific plant model we use an inexpensive analog sensor to sample the specific critical process parameter 'liquid tank level.' The high speed sensors provides 1500 integer samples during a 1-s interval. This sampling rate and interval were determined experimentally to best illustrate typical oscillatory behavior in response to set point and step disturbances; this practice is also typically performed by SMEs during process control implementations and recipe change testing.

11.5.2.2 Configuration of Plan Security Monitoring Model

In this section we describe how our security model is configured using the collaborative method. We demonstrate that even if the variables captured from 1-s network polling of the PLC are used to populate a decision model, more information is needed to accurately define the security model. One still requires (1) an SME control engineer to verify the typical and over/under limit range bands thus describing the semantics of the observed value time-series, and (2) the SME to categorize these excursion events to allow un-ambiguous labeling of them to use in the semantic message for each corresponding alert.

Although there are many other process variables that could be monitored, we selected the "tank" level as our sole critical process variable. In our collaborative method, this critical process variable will receive priority in monitoring, and a common understanding of typical and alarm values will be determined. Semantics and notification actions to be implemented are achieved by collaborative discussions between plant personnel and the security monitoring analysts. We discuss these aspects in the subsections below.

Inference of Critical Values from Network Traffic Data

A first step in defining the security model is to examine the network MODBUS traffic itself. In our case study implementation, an ICS network traffic security monitor collects MODBUS TCP traffic between the HMI and the Arduino PLC. We use this passive monitoring data to attempt to construct a valid parallel model of the critical plant operating parameters. In the lower right section of Fig. 11.6, we show actual configuration and register content logs from the running plant. The traffic monitor is able to receive and parse all MODBUS traffic, and provide register-labels for each of the values received, but the underlying configuration is not clear.

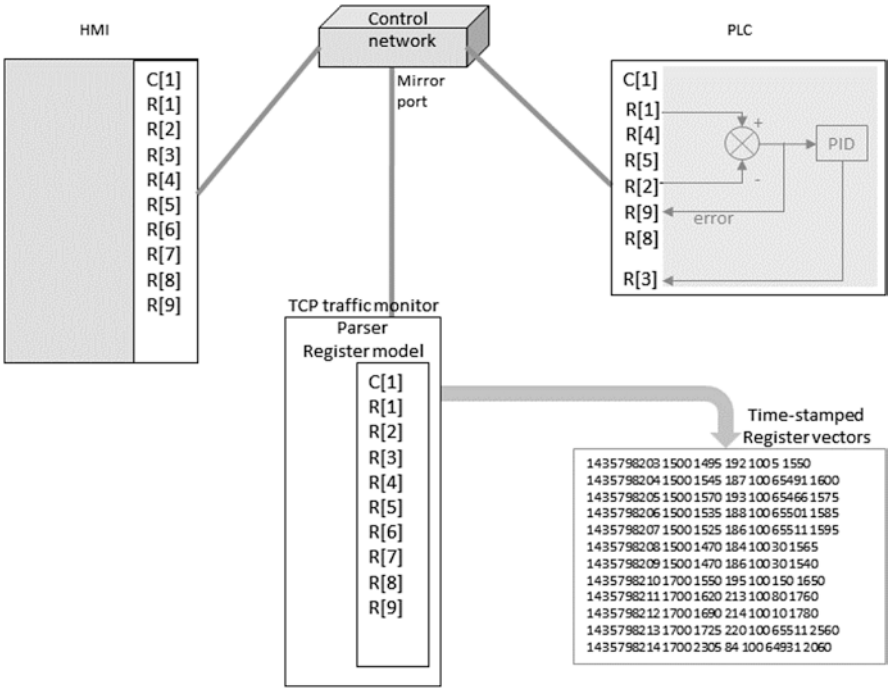

Fig. 11.6 Observed plant parameters from ICS security monitor

Without knowledge of the plant or the monitored process, it is difficult to infer (1) which process variable is represented by each register, and (2) the semantic of each register. Is the register an input, or an output, or a set-point value, or unused, or even more importantly, not used today for this recipe and batch but may be used in a different recipe or batch run?

While machine learning can certainly be applied to this type of model, it will only identify the degree of 'normality' of current observations compared to past observations. A machine learning approach really does not allow inference of the purpose, configuration, or semantic of these parameters. These details and even more useful information concerning the criticality and acceptable value-ranges of the important (critical) values must be obtained from the plant operator/SME.

Determination of Critical Values from SME Input and Network Traffic Data

A next step in defining the security model is to consult with plant personnel on the exact usage of PLC registers and other plan process configuration details that can be monitored. In our case study plant, after consultation with the plant operator/SME, we are able to better understand plant operating processes.

First, we find that the R[2] register values report our critical value of the tank level. Further information is provided in the form of upper- and lower-alarm bands that should be monitored for the R[2] register.

Second, our plant operator/SME also advises that the system is frequently subjected to a disturbance (unpredicted change in load) which presents as a significant step-change to the process. Tank levels may traverse the specified alarm bands during this time and this should not be considered a course for alarm.

Finally, disturbances and set-point changes perturb the controller in similar manners causing a damped oscillatory deviation in the tank level having a frequency of about 5Hz. Since the MODBUS HMI polls every 1000 ms, various peak values from such oscillations may be represented in the logs and model. This information can be used to verify and validate SME input for the allowed ranges of the tank level.

Model Refinement and Verification using Network Traffic Data

Time-stamped log records from the traffic monitor provide input to our semantic security model. In Fig. 11.7, we show the R[2] tank level in column 3 of the log records. With further knowledge from the plant operator/SME, we identify the high-speed sensor tank level measurements as column 7 in f the log records(see Fig. 11.7).

Typically, plant personnel can specify more precise conditions needed to justify issuance of a semantic security monitoring alert by providing an additional qualifier describing the extent or severity of the alarm condition. In the case of our case study, we may be informed that during disturbances, the tank level will often travel outside of the alarm bands, significantly so, but, will not maintain these extreme values for 'longer than 1 s' (as the control algorithm applies output actuation to bring this controlled parameter back into desired range).

Plant personnel provide the information that the level set-point $\mathbf{h}_{\mathbf{ref}}$ is indicated by the value in R[1]. Referring to the above annotated model, the set-point appears as column 2 in the register vector log. We notice it starts at 1500 and after 8 s, changes to 1700. The process engineer recalls that for this recipe, the batch starts at 1500 and then does an automatic set-point change to 1700 at 8-s. Thus, the alarm bands need to be adjusted in the detection model to reflect this planned event.

As an example of a semantic security monitoring language, we provide a sample implementation of a windowed alarm band integrator to calculate a moving area-under-curve value to use to identify routine excursions as distinct from more severe alarms where the tank level maintains an alarm value for a longer time period and thus justifying alarming and notification.

Now equipped with a level monitoring capability with independent sensor input, we focus on analyses which can determine the magnitude and severity of alarm events. The observation of one alarm is not usually sufficient justification for declaration of a significant event. In process control it is often the case that an alarm condition must persist for a minimum time duration after which it becomes a concern. Many power-conversion processes specify a maximum duration for delivery

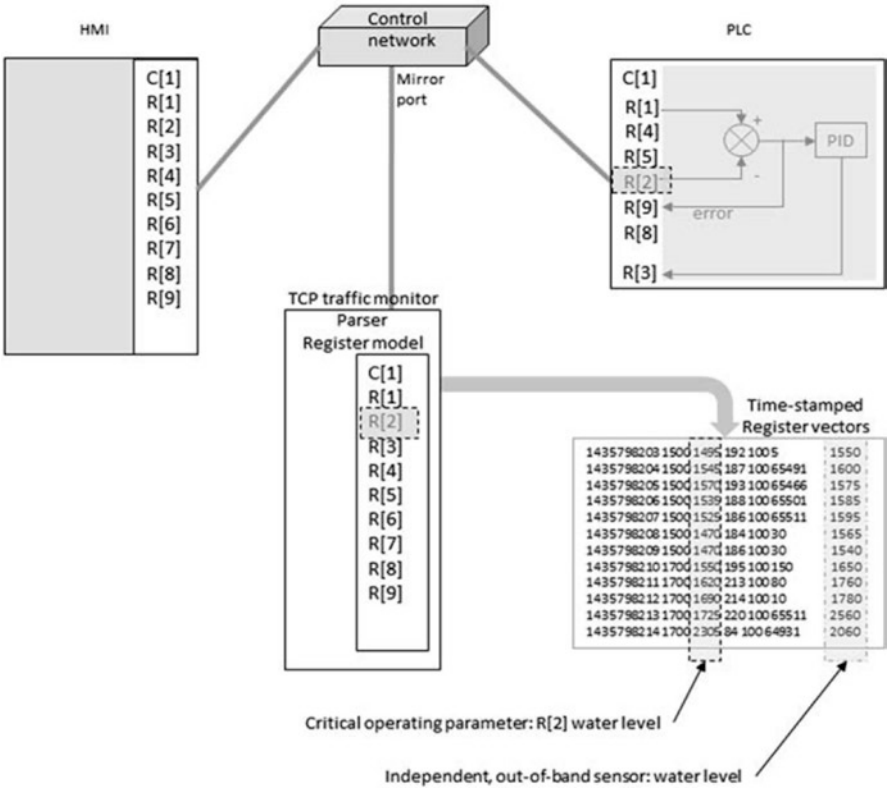

Fig. 11.7 Observed plant parameters with collaborative input from plant personnel

of power at some value above its nominal rating. For example, an aircraft turbine engine might be rated at 90 % power continuous duty, but 5-min at 100 % and 1-min at 120 % power. Our analytic alerting method needs to allow expression of these various alarm bands and provide means to calculate accumulated values over various time-durations.

To illustrate this capability, we implemented a windowed, area-under-curve (AUC) function and applied it to the tank level log data (see Fig. 11.8). The function accumulates net area above the alarm-high level as well as net area below the alarm-low level using trapezoidal integration.

We are then informed by the plant personnel that if the AUC stays above 1000 (unit-secs) for longer than 20 s, then it is an alarm condition that requires notifications. Above, we implement an AUC integrator with sample length of 30, calculated each new sample-time (1000 ms) using alarm bands of 1800 and 1300. Negative excursions are accumulated along with positive excursions providing a cumulative value shown above. Thus, although AUC for the level parameter exceeds the alarm level, it does not persist longer than 6 s in the sample above and no notifications are needed.

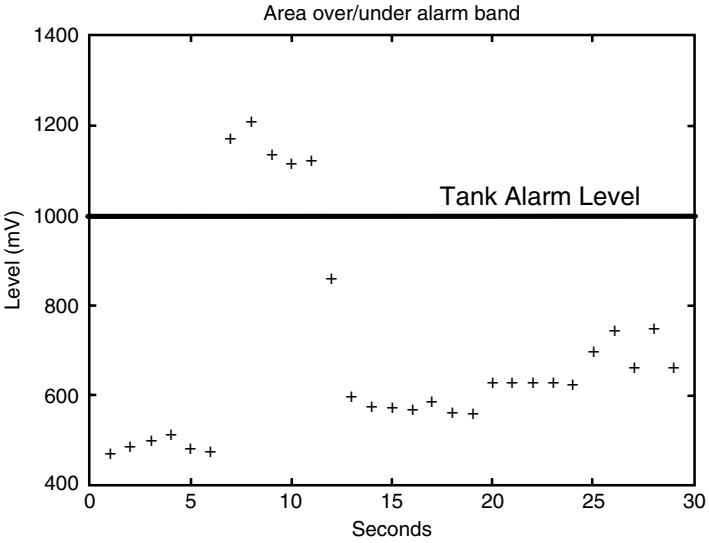

Fig. 11.8 ICS security monitor data showing tank level excursions above tank alarm level

The tank level alarm process described here is somewhat complex, but it represents exactly what the plant operator/SME desire. We argue that our collaborative security model is more effective than one based solely on network packet analysis.

Model Refinement and Verification using Out-of-Band Data (High speed sensor)

Process variables may not be sampled at a sufficient rate or precision to inform security/safety decisions. Our case study plan can clearly show this distinction. It is customary in process control to provide parameter information only as frequently as is needed to control the plant. It is however quite common for many process variables to exhibit wild or oscillatory excursions at much faster rates. This can occur in response to disturbances (changing load conditions) or set-point changes (changing value requirements). In a properly designed plant control environment, such excursions would be anticipated and means provided to diminish these behaviors before any damage is caused. Unfortunately, since the control design can often prevent adverse effects by dampening such excursions, a cyber monitoring system may only receive sampled parameter values which could include some of these extreme parameter values that are clearly out of range. This we anticipate will be the source of most false alarms, or false-positive alerts.

If the (few) monitored parameters are described as critical variables by the SME, then it may be justifiable to augment normal network monitoring of ICS sensors with additional, high-speed out-of-band sensors. Gathering independent measurements of

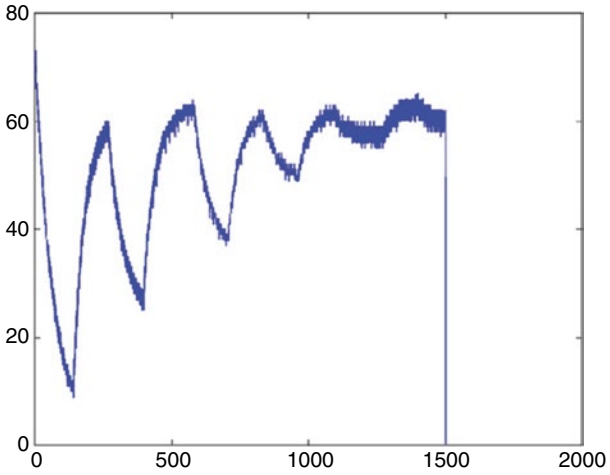

Fig. 11.9 One-second (1500 samples) duration high-speed sensor sampling of tank level during system disturbance. Within 1500 samples (1 sample of the Independent LEVEL sensor), significant deviations are observed

such critical parameters will supply significant uplift to data and decision quality. In our case study implementation, an independent level sensor that polls the tank level at high speeds, in synchronization with the ICS Security monitor.

To more fully understand the excursions in the tank-level values in response to step change disturbances, we obtain a trace from a high-speed sensor. In Fig. 11.9, we show 1500 samples for a 1-s interval triggered by the increased step load disturbance. This oscillatory behavior occurs in response to each load change. Analysis of these samples using a DFT (discrete Fourier transform) confirm the principal component of 5-Hz along with low frequency and DC components. Monitoring of the 1-s MODBUS samples of the tank level with the ICS Security monitor will not represent this signal consistently.

Knowledge of this oscillatory behavior allows us to better interpret the meaning of the tank level when sampled at 1-s intervals. Frequent excursions above and below the alarm bands can be observed for the critical variable of the tank level. It is obvious that one-second network sampling is not adequate to alert reliably on the behavior. Actual value ranges and durations should again be determined in consultation with plant personnel to provide realistic ranges compatible with the analysis, decision, and alert notification time-frames.

11.5.2.3 Intrusion Detection Alerting

We define ‘alerting’ as automatic information generation to be sent to a human analyst for further consideration. We define an alarm as a determination of a possible compromise or other insecure situation as determined by the human analyst, based on alerting information that was provided by the intrusion detection system.

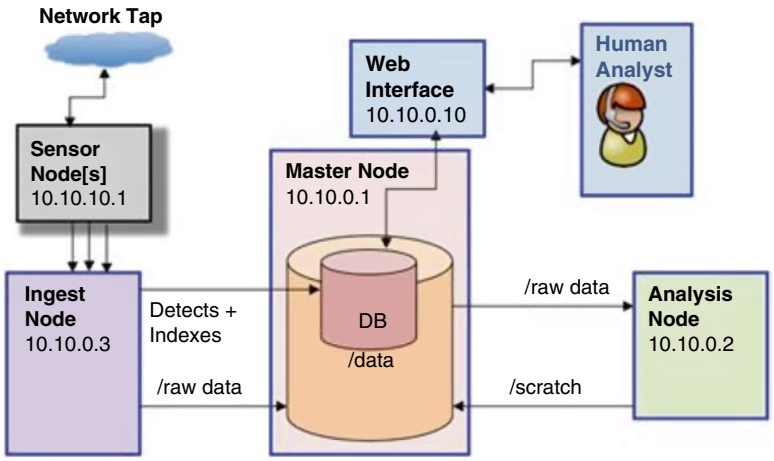

Fig. 11.10 Sample generic intrusion detection architecture

Our collaborative intrusion detection model is implemented at ARL in a live testbed. We report on general findings from our testbed experiments. In Fig. 11.10, we show the implemented IDS architecture in our testbed. A network tap (e.g. SPAN port on a switch) provides network capture data to one or more sensor nodes. Some of the data are pre-processed on the sensor nodes into 'detects' (detect/alert information) and index data. The Ingest node then forwards that data to a master node, which stores raw data and provides indexed information for analyst web tools. More complicated analytics are executed by the Analysis Node, which again places results back on the Master Node for the web interface to display. The Web Interface contains an HTTP web server with web analytics and web links for execution of additional analysis tools. The Human Analyst then examines alerting information that resides in the system using various analytical tools.

In our testbed implementation, IDS alerting by the Sensor Node is generated in one of three methods:

1. Critical Process Values: This is the collaborative method just described. Critical Process variables are those that define whether the control system is successfully operational or not. Sensor nodes are modified specifically to monitor the value of all critical process variables that have been defined collaboratively between the ARL security engineer and the site operations SME. Nominal values, and upper and lower limits for critical values are programmed into the sensor node. The example of a critical value for our case study is the tank level. Danger can incur if the tank overflows.
2. Network Packet Reporting Values: Anomalies identified by deep inspection of the network packets captured by the network traffic monitor are reported by the sensor node. These network values are not necessarily critical process variables, but if they are not within reasonable range, they may indicate a minor issue or

warn of a future significant network intrusion. Collaborative discussions between the ARL security engineer and the plant operator/SME will determine nominal values and ranges, and the ARL security engineer will use network captures to refine the model.

3. Network Traffic Pattern Anomalies: Traffic on a control system network is very minimal and is usually very periodic. Methods similar to the non-signature based ICS anomaly detection discussed earlier in this chapter are used to provide alerting. As we note, we do not regard this alerting information as something the analyst should use exclusively to report an alarm condition. These alerts are expected to have a high enough false-alarm rate that more information should be used to draw a conclusion about reporting an alarm. Metrics and limits to those metrics will need to be defined. These metrics will be specific to the site and will define when anomalous network traffic is indicated. This calibration process should be done carefully to ensure that false alarm (alert) information is not passed up to the Interrogator Analyst. Network traffic pattern anomaly alerts are not necessarily critical in that they do not indicate that the system is not functional. However, they may indicate that an intrusion is in progress, and that the critical process variables are threatened. Providing this alert information to the analyst can help avoid a critical process failure.

11.6 Summary and Conclusions

One of the central themes of IDS research and development for ICSs is whether one can easily transfer intrusion detection methods developed and implemented for early and current ICT systems. ICSs are not merely a collection of networked computer servers, network switches, and routers, such as you might find in an ICT network. ICSs often simultaneously employ many different intrusion detection and alerting methods, including signature-based, anomaly-based, and specification-based methods.

An ICS can be described in terms of a Primary Bus and Secondary Bus architecture, where the Secondary Bus connects field devices to PLCs and other ICS hardware. The Primary Bus is often connected via standard network communication protocols and perhaps even to a corporate network or the Internet, but the Secondary Bus may not have network-addressable components. We assume ICS IDSs will only connect to the Primary Bus.

Digital Bond's Quickdraw adaptation for signature-based methods using Snort was used on ICSs beginning approximately 2008. ICS (and ICT) security researchers however often favor a combination of signature-based ID methods and non-signature based ID methods. Starting approximately 2010, ICS ID techniques began to diverge from their ICT Enterprise "parents" and developed unique and useful methodologies that focused more on characterizing the actual ICS process values, even to the point of creating and monitoring a distinct copy of the ICS plant PLC registers. These register states can be configured automatically via network polling, or with collaborative assistance from plant personnel.

As we demonstrate in our sample plant case study, all of the PLC registers do not need to be monitored for alerting. If PLC register values obtained from polling the network are used to populate a decision model, more information is needed to accurately define a sensible security model. In some cases, higher time precision is also needed to accurately determine the alarming method. Network polling of register values and other network traffic can be used for identifying anomalous behavior once the plant variables are defined. Anomaly detection in ICSs is more effective than for ICT networks due to the small volume and high regularity of ICS network traffic. ARL testbed research on intrusion detection methods for ICSs proposes a three-method approach: collaborate monitoring of specific critical plant process variables, a process-oriented anomaly-based technique based on network polling, and non-signature base anomaly detection technique for network traffic patterns. Alerts from all three methods are sent to a human analyst who investigates further and decides if plant operational personnel should be notified.

References

Anderson, J. P. (1980). *Computer security threat monitoring and surveillance*. Fort Washington, PA: James P. Anderson.

Astrom, K. (2002). Chapter 6. PID control. In *Control System Design Lecture Notes for ME 155A*. UCSB, CA.

Axelsson, S. (2000). The base-rate fallacy and the difficult of intrusion detection. *ACM Transactions on Information and System Security (TISSEC), 3*(3), 186–205.

Carcano, A., Coletta, A., Guglielmi, M., Masera, M., Fovino, I., & Trombetta, A. (2011). A multidimensional critical state analysis for detecting intrusions in SCADA systems. *IEEE Transactions on Industrial Informatics, 7*(2), 179–186.

Cheung, S., Dutertre, B., Fong, M., Lindqvist, U., Skinner, K., & Valdes, A. (2007, January). Using model-based intrusion detection for SCADA networks. In *Proceedings of the SCADA Security Scientific Symposium* (Vol. 46, pp. 1–12).

Denning, D. E. (1987). An intrusion-detection model. *IEEE Transactions on Software Engineering, 2*, 222–232.

Denning, D. E., & Neumann, P. G. (1985). Requirements and model for IDES—A real-time intrusion detection expert system. *Document A005, SRI International*, *333*.

Fovino, I., Carcano, A., Murel, T., Trombetta, A., & Masera, M. (2010). Modbus/DNP3 state-based intrusion detection system. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*.

Gao, W., Morris, T., Reaves, B., & Richey, D. (2010). On SCADA control system command and response injection and intrusion detection. In *2010 ECrime Researchers Summit*.

Goldenberg, N., & Wool, A. (2013). Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems. *International Journal of Critical Infrastructure Protection, 6*(2), 63–75.

Hadžiosmanović, D., Sommer, R., Zambon, E., & Hartel, P. (2014). Through the eye of the PLC. In *Proceedings of the 30th Annual Computer Security Applications Conference on—ACSAC '14*.

Liao, H. J., Lin, C. H. R., Lin, Y. C., & Tung, K. Y. (2013). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications, 36*(1), 16–24.

Long, K.S. (2004). Catching the cyber spy: ARL's interrogator, *Army Research Lab Technical Report*, (DTIC ACC. NO. ADA432198).

Oman, P., & Phillips, M. (2008). Intrusion detection and event monitoring in SCADA networks. In *IFIP International Federation for Information Processing Critical Infrastructure Protection* (pp. 161–173).

Tsang, C., & Kwong, S. (2005). Multi-agent intrusion detection system in industrial network using ant colony clustering approach and unsupervised feature extraction. In *2005 IEEE International Conference on Industrial Technology*.

Tylman, W. (2013). SCADA intrusion detection based on modelling of allowed communication patterns. In *Advances in Intelligent Systems and Computing New Results in Dependability and Computer Systems* (pp. 489–500).

Verba, J., & Milvich, M. (2008). Idaho national laboratory supervisory control and data acquisition intrusion detection system (SCADA IDS). In *2008 IEEE Conference on Technologies for Homeland Security*.

Yang, Y., Littler, T., Wang, H., Mclaughlin, K., & Sezer, S. (2013). Rule-based intrusion detection system for SCADA networks. In *2nd IET Renewable Power Generation Conference (RPG 2013)*.

Yang, D., Usynin, A., & Hines, J. W. (2006, November). Anomaly-based intrusion detection for SCADA systems. In *5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (npic&hmit 05)* (pp. 12–16).

Zhu, B., & Sastry, S. (2010, April). SCADA-specific intrusion detection/prevention systems: A survey and taxonomy. In *Proceedings of the 1st Workshop on Secure Control Systems (SCS)*.