DEEP PACKET INSPECTION FOR
PHYSICS-BASED ANOMALY DETECTION IN
INDUSTRIAL CONTROL SYSTEMS

by

David Ignacio Urbina Fuentes

APPROVED BY SUPERVISORY COMMITTEE:

Alvaro A. Cárdenas, Chair

Latifur Khan

Kamil Sarac

Bhavani Thuraisingham
To the three great women of my life..., 
my loving wife Sara, my mother Carmen, and my grandmother Josefa.
Whose continuous love, support, and teachings ignited in me 
the fire that gave birth to this work...
DEEP PACKET INSPECTION FOR
PHYSICS-BASED ANOMALY DETECTION IN
INDUSTRIAL CONTROL SYSTEMS

by

DAVID IGNACIO URBINA FUENTES, BS, MS

DISSERTATION
Presented to the Faculty of
The University of Texas at Dallas
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN
SOFTWARE ENGINEERING

THE UNIVERSITY OF TEXAS AT DALLAS
December 2016
ACKNOWLEDGMENTS

The work presented in this dissertation was only possible because of the hard work and dedication of many.

My sincere appreciation to my supervisor, Dr. Alvaro A. Cárdenas, for his invaluable guidance and support during my Ph.D. program. His passion and dedication to advance the state-of-the-art became a north star to follow. I certainly learned with him that only great leaders can turn coals into diamonds.

To my partner in crime, Dr. Jairo Giraldo, for the sleepless nights helping me to run experiments, and the invaluable answers for all my questions about control theory, allowing me to fill in my knowledge gaps.

I would also like to extend my thanks to Dr. Aditya P. Mathur, Dr. Nils O. Tippenhauer, and Dr. Justin Ruths from the iTrust - Centre for Research in Cyber Security at the Singapore University of Technology and Design, who received me as Visitor Researcher and so openly provided me with access to the Secure Water Treatment testbed. I want to specially thank Dr. Tippenhauer for his guidance and advise during my research time at iTrust.

Finally, my appreciation to Keith A. Stouffer, Richard Candell, CheeYee Tang, and Timothy Zimmerman at the National Institute of Standards and Technology for the access to the Tennessee Eastman and Cooperative Robotic Assembly testbeds.

This research was supported, in part, by the National Institute of Standards and Technology – U.S. Department of Commerce under awards 70NANB14H236 and 70NANB16H019, and by the National Science Foundation under award CNS-1553683.

The views and conclusions contained in this dissertation are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NIST, NSF, or the U.S. Government or any of its agencies.

October 2016
In the past couple of years we have seen an emerging field of research focusing on using the intrinsic physical properties of an Industrial Control System process for anomaly detection; however, these efforts have been mostly disconnected, finding little common ground between each other to create a foundation from which other researchers can build improvements.

In this dissertation, we review previous work based on a unified taxonomy that allows us to identify limitations, unexplored challenges, and new solutions. In particular, we propose a new adversary model and a way to compare previous work with a new evaluation metric based on the trade-off between false alarms and the negative impact of undetected attacks, which defines the worst-case adversary model for detection mechanisms based on models of the physical world. We use the metric to compare design choices for detecting anomalies, and design choices for modeling the “physics” of the system. We also show the advantages and disadvantages of three experimental scenarios to test the performance of attacks and defenses: real-world network data captured from a large-scale operational facility, a fully-functional testbed that can be used operationally for water treatment, a simulation of a chemical process, and a simulation of a frequency control in the power grid.

We also discuss practical attacks applied to a room-sized water treatment testbed. We implement scenarios in which the attacker manipulates or replaces sensor data as reported
from the field devices to the control components. As a result, the attacker can change the system state vector as perceived by the controls, which will cause incorrect control decisions and potential catastrophic failures. We discuss practical challenges in setting up Man-In-The-Middle attacks on the Field Communications Network of Industrial Control Systems, and how the attacker can overcome them.

Finally, we analyze the problem of security monitor placement in industrial control networks, and show that there are locations that allow to detect low-level attacks. Based on our analysis, we design a novel low-level security monitor that is able to directly observe the Field Communications Networks.
TABLE OF CONTENTS

ACKNOWLEDGMENTS ........................................................................ v
ABSTRACT ....................................................................................... vi
LIST OF FIGURES ............................................................................. xii
LIST OF TABLES ................................................................................ xv

CHAPTER 1 INTRODUCTION ................................................................. 1
  1.1 Motivation .................................................................................. 2
  1.2 Goal, Research Questions, and Approach .................................. 4
  1.3 Contributions and Dissertation Outline ................................. 5

CHAPTER 2 SYSTEMATIZATION OF KNOWLEDGE ............................. 9
  2.1 Introduction ................................................................................ 9
  2.2 Scope of Our Study ................................................................... 11
  2.3 Background and Taxonomy ....................................................... 12
    2.3.1 Background on Control Systems ....................................... 12
    2.3.2 Taxonomy ......................................................................... 17
    2.3.3 State Estimation ............................................................... 21
  2.4 Survey of Previous Work .......................................................... 22
    2.4.1 Power Systems .................................................................. 22
    2.4.2 Industrial Control Systems .............................................. 24
    2.4.3 Control Theory ............................................................... 28
    2.4.4 Miscellaneous Domains .................................................. 30
  2.5 Discussion and Suggestions ..................................................... 32
    2.5.1 General shortcomings ...................................................... 34
    2.5.2 Recommendations ........................................................... 34
  2.6 Conclusions .............................................................................. 37

CHAPTER 3 ATTACKING FIELD COMMUNICATIONS NETWORKS ........ 39
  3.1 Introduction .............................................................................. 39
## LIST OF FIGURES

1.1 Dissertation Outline. ......................................................... 5
2.1 Layered architecture of ICS. .................................................. 13
2.2 Anomaly Detection Architecture. ............................................. 14
2.3 Compromised sensor signals. ............................................... 15
2.4 Compromised actuation signals. .......................................... 16
2.5 Compromised controller. ................................................... 16
2.6 Central control attacks. ..................................................... 17
2.7 Detection module. .......................................................... 18
2.8 State estimation. ............................................................ 21
3.1 SWaT physical process overview. ......................................... 41
3.2 SWaT network architecture. ............................................... 42
3.3 Example ring topology in SWaT. .......................................... 43
3.4 EtherNet/IP encapsulation. ................................................ 47
3.5 CIP object-oriented packet structure. .................................... 48
3.6 Analog Input module. ....................................................... 50
3.7 Scaling 4-20mA to water flow. ............................................ 51
3.8 Detection mechanism and Man-In-The-Middle attack for SWaT. ... 52
3.9 Water level attack detection. .............................................. 53
3.10 Stealthy attacks in SWaT. ................................................ 53
4.1 Stage controlling the pH level. ............................................ 58
4.2 Water pH safe levels. ....................................................... 59
4.3 Water pH attacks. .......................................................... 60
4.4 Adaptive attack. ........................................................... 61
4.5 Proposed tradeoff metric. .................................................. 62
4.6 ROC vs. proposed metric. .................................................. 63
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.19</td>
<td>Multi-variable detection at Tennessee Eastman</td>
<td>103</td>
</tr>
<tr>
<td>5.20</td>
<td>Multi-Variable Detection architecture</td>
<td>104</td>
</tr>
<tr>
<td>5.21</td>
<td>Attack on PST sensor</td>
<td>105</td>
</tr>
<tr>
<td>5.22</td>
<td>Optimal stealthy attack on PST sensor</td>
<td>106</td>
</tr>
<tr>
<td>5.23</td>
<td>Detection of optimal stealthy attack on PST sensor</td>
<td>107</td>
</tr>
<tr>
<td>6.1</td>
<td>Number of packets per minute (April 23)</td>
<td>112</td>
</tr>
<tr>
<td>6.2</td>
<td>Number of packets per minute (May 5)</td>
<td>112</td>
</tr>
<tr>
<td>6.3</td>
<td>Finding periodicity</td>
<td>114</td>
</tr>
<tr>
<td>6.4</td>
<td>Feature vector space</td>
<td>118</td>
</tr>
<tr>
<td>6.5</td>
<td>PCA with 2 components retaining 98% of variance</td>
<td>121</td>
</tr>
<tr>
<td>6.6</td>
<td>PCA of clustered packets with k=6</td>
<td>121</td>
</tr>
<tr>
<td>A.1</td>
<td>Modbus stack</td>
<td>128</td>
</tr>
<tr>
<td>A.2</td>
<td>Modbus packet structure</td>
<td>129</td>
</tr>
<tr>
<td>A.3</td>
<td>CIP stack</td>
<td>130</td>
</tr>
</tbody>
</table>
LIST OF TABLES

2.1 Taxonomy of related work. .................................................. 33
3.1 RIO I/O modules. .............................................................. 50
4.1 Advantages and disadvantages of different evaluation setups. ............. 66
5.1 Detectability of Attack Depending on Trust in Components. ................. 85
5.2 Modules for each PLC. ....................................................... 91
6.1 Datasets collected in 30,000-meter AMI network. ............................. 111
6.2 Address information collected at layers 3 and 7 over a 24-hour period, and
differences with a 24-hour trace collected 10 days later. .......................... 116
6.3 Number of unique connections among end points collected at layers 3 and 7 over
a 24-hour period, and differences with a 24-hour trace collected 10 days later. . 116
6.4 Distribution of RCF flag among all unsolicited messages (first message of a session). 118
6.5 Components vs. Retained Variance. ....................................... 120
6.6 Mapping between intrusion detection techniques and detectable attacks despite
encryption. .............................................................................. 122
CHAPTER 1
INTRODUCTION

Industrial Control Systems (ICS) have been at the core of critical infrastructures, manufacturing and industrial plants for decades, and yet, there have been few confirmed cases of cyber-attacks. Control systems, however, are now at a higher risk to computer attacks because their vulnerabilities are increasingly becoming exposed and available to an ever-growing set of motivated and highly-skilled attackers: control systems are increasing in size and scope, they are introducing new embedded devices (such as wireless sensors spread in large areas) and increasing their connectivity; furthermore, there are more and more groups with the skills and motives to launch successful attacks.

There have been many computer-based incidents in ICS. Computer-based accidents can be caused by any unanticipated software error, like the power plant shutdown caused by a computer rebooting after a patch [76]. Non-targeted attacks are incidents caused by the same attacks that any computer connected to the Internet may suffer, such as the Slammer worm infecting the Davis-Besse nuclear power plant [141], or the case of a controller being used to send spam in a water filtering plant [42].

However, the biggest threat to control systems are targeted attacks. These attacks are the ones where the miscreants know that they are targeting control systems, and therefore, they tailor their attack strategy with the aim of damaging the physical system under control. Targeted attacks against control systems are not new. Physical attacks —for extortion and terrorism— are a reality in some countries [109]. Cyber-attacks are a natural progression to physical attacks: they are cheaper, less risky for the attacker, are not constrained by distance and geopolitical boundaries, and are easier to replicate and coordinate.

One of the first reported targeted attacks to ICS was the attack on Maroochy Shire Council's sewage control system in Queensland, Australia [129]. There are many other reported targeted attacks [118, 120, 2, 1, 75, 84, 50], including a high-profile attack against Saudi Aramco [121]; however, no other attack has demonstrated the potential threats that control systems are subject to as well as the Stuxnet worm [44, 80]. Stuxnet has made clear that there are groups with the motivation and skills to mount sophisticated computer-based attacks to critical infrastructures, and that these attacks are not just speculations or belong only in Hollywood movies.
Not only can Stuxnet-like attacks have devastating consequences, but they are also very difficult to detect. Because Stuxnet used zero-day vulnerabilities, anti-virus software would not have prevented the attack. In fact, the level of sophistication of the attack is such that some well known security companies, such as Kaspersky, were not initially able to detect it [116]. In addition, victims attempting to detect modifications to their embedded controllers would not see any rogue code as Stuxnet hides its modifications with sophisticated PLC rootkits, and validated its drivers with trusted certificates.

Finally, because ICS systems perform vital functions in national critical infrastructures, such as electric power distribution, oil and gas refining, and water treatment and distribution, the disruption of control systems could have a significant impact on public health, safety, and lead to large economic losses. Securing ICS in critical infrastructures is thus of national priority [143, 62].

1.1 Motivation

While research on intrusion detection systems has extended for more than 3 decades [39], less than a decade ago, one of the first works to consider intrusion detection in industrial control networks was Cheung et al. [33]. 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 Information Technology (IT) networks.

The limited memory and processing power of widely-deployed control devices, and the processing real-time requirements of control systems, constraint the application of Host-based detection approaches. Moreover, even if the computing resources are available, control devices undergo a special certification process that would limit the modification of their firmware to include intrusion detection functionality. Also, as industrial control networks evolve, new network attack vectors that could be exploited to get unauthorized access are being opened: control networks are growing in size and in covered distances —achieved by the introduction of long-distance communication protocols (such as Ethernet IEEE 802.3, WLAN IEEE 802.11, etc.)—, and are being interconnected with enterprise networks and even with the Internet —by the adoption of the TCP/IP stack—. For these reasons, in this work we explore a Network-based approach.

The combination of Commercial-Of-The-Shelf (COTS) technologies, state-of-the-art embedded devices, with legacy technologies —originally designed without concern for security—, have increased the complexity of industrial control networks. This makes them specially susceptible to the presence of zero-day vulnerabilities which would defeat commonly known
*misuse-based* intrusion detection approaches. See Debar et al. [38] for a description of *misuse-based* and *anomaly-based* detection approaches.

Anomaly-based intrusion detection systems have been proposed for smart grid networks [6, 24] and in general for ICS systems [100]; however, as Hadziosmanovic et al. [52] showed, intrusion detection systems that fail to incorporate domain-specific knowledge and the context in which they are operating, will still perform poorly in practical scenarios. Even worse, an attacker that has obtained control of a sensor, an actuator, or a PLC can send manipulated sensor or control values to the physical process while complying to typical IT traffic patterns (Internet Protocol (IP) addresses, protocol specifications with finite automata or Markov models, connection logs, etc.).

One of the fundamentally unique properties of industrial control—when compared to general IT systems—is that the physical evolution of the state of a system has to follow immutable *Laws of Physics*. Moreover, no matter the sophistication, resourcefulness, and determination of a cyber-attacker, any deviation imposed over the industrial process will still abide to the *Laws of Physics*. Therefore, as Cárdenas et al. [30] first pointed out, the physical properties of chemical systems (thermodynamics), water systems (fluid dynamics), or the power grid (electromagnetism), for example, can be used to create time series models that we can then use to confirm that the control commands sent to the field were executed correctly and that the information coming from sensors is consistent with the expected behavior of the system: if we open an intake valve we should expect that the water level in the tank should rise, otherwise we may have a problem with the control, actuator, or the sensor (this anomaly can be either due to an attack or a faulty device).

Finally, in contrast to work in intrusion detection that focuses on monitoring such low-level IT observations, in this dissertation we expand on the recent and growing literature in computer security conferences (e.g., CCS’15 [128], CCS’09 [88], ACSAC’13[95], ACSAC’14 [53], ASIACCS’11 [29], and ESORICS’14 [146]) studying how Deep Packet Inspection of sensor values from physical observations, and control signals sent to actuators, can be used to build physical models that allow for the detection of attacks in industrial control networks.

While the growing number of publications in the last couple of years shows the growing importance of leveraging the physical properties of control systems for security, we have found that most of the works focusing on this topic are presented independently, with little context to related work. For that reason, research results are presented with different models, different evaluation metrics, and different experimental scenarios. That disjoint presentation of ideas is a limitation for creating the foundations necessary for discussing results in this field and for evaluating new proposals.
1.2 Goal, Research Questions, and Approach

Our goal is to exploit the intrinsic relationship between ICS networks and the physical properties of the controlled process in order to develop physics-based anomaly detection techniques that are suited for these networks. Specifically, we address the following research questions:

Main RQ: How can Deep Packet Inspection be leveraged in order to achieve Physics-based Anomaly Detection in ICS? What are the implementation challenges? How can we evaluate physics-based anomaly detection in the presence of stealthy attacks?

The first step we take to answer this question is to perform an extensive background review in ICS networks. This includes ICS network architecture, protocols, and anomaly detection architectures for ICS. We then proceed with a deep literature review from security, control, smart/power grid, and others venues, where physics-based anomaly detection approaches have been proposed. We design a taxonomy that allows us to unify the highly segregated book of knowledge in this field, and to clearly depict the shortcomings and research opportunities yet to be exploited.

Second, we perform Deep Packet Inspection on traffic from multiple ICS setups, including: a room-size water treatment testbed, a real large-scale operational system managing more than 100 PLCs, and simulations of chemical plants and primary frequency control in the power grid. For each of these setups we implement attacks against the control signals, and derive physics-based models that can be leveraged for anomaly detection. In particular, we study and measure the impact of stealthy attacks on the physical process, and proposed new metrics that allow for the evaluation of such impact in the presence of different physics-based anomaly detectors.

Supplemental RQ: How to perform anomaly detection when in presence of encrypted ICS traffic?

A variety of techniques to enable IDSes to monitor encrypted traffic in ICS networks, including sharing keys with IDS sensors, leveraging partial encryption, or applying traffic analysis techniques. To answer this research question, we investigate how some of those approaches could work in practice by applying them experimentally on real traffic captured at a large operational utility AMI network.
We discuss the effectiveness of each of these approaches in detecting suspicious activity in the context of AMI traces from that we were able to collect. The main contributions of this work are to offer a detailed view of the internal network communications found on a large AMI and to provide insights on the challenges and possible solutions to identify intrusions despite the deployment of encryption.

1.3 Contributions and Dissertation Outline

In the following, we present the outline for this dissertation (see Figure 1.1) and short summary of each chapter, including the publications used as basis for it:

![Dissertation Outline](image)

**Figure 1.1. Dissertation Outline.**

**Chapter 2 - Systematization of Knowledge.** Monitoring the “physics” of control systems to detect attacks is a growing area of research. In its basic form a security monitor creates time-series models of sensor readings for an industrial control system and identifies anomalies in these measurements in order to identify potentially false control commands or false sensor readings. Despite a growing interest in this area, previous efforts have been
mostly disconnected, finding little common ground between each other. In this chapter, we review previous work based on a unified taxonomy that allows us to identify limitations, unexplored challenges, and new directions. This chapter comprehends a compilation of the following publications:


Chapter 3 - Attacking Field Communications Networks. The study of cyber-attacks in industrial control systems is of growing interest among the research community. Nevertheless, restricted access to real industrial control systems that can be used to test attacks has limited the study of their implementation and potential impact. In this chapter, we discuss practical attacks applied to a room-sized water treatment testbed. The testbed includes a complete physical process, industrial communication systems, and supervisory controls. We implement scenarios in which the attacker manipulates or replaces sensor data as reported from the field devices to the control components. As a result, the attacker can change the system state vector as perceived by the controls, which will cause incorrect control decisions and potential catastrophic failures. We discuss practical challenges in setting up Man-In-The-Middle attacks on fieldbus communications in the industrial EtherNet/IP protocol and topologies such as Ethernet rings using the Device-Level-Ring protocol. We show how the attacker can overcome those challenges, and insert herself into the ring. Once established as a Man-in-the-Middle attacker, we launched a range of attacks to modify sensor measurements and manipulate actuators. We show the efficacy of the proposed methodology in two experimental examples, where an adversary can intelligently design attacks that remain undetected for a typical bad-data detection mechanism. This chapter corresponds to the contributions of the following publication:

Chapter 4 - Limiting The Impact of Stealthy Attacks. While attacks on information systems have for most practical purposes binary outcomes (information was manipulated/eavesdropped, or not), attacks manipulating the sensor or control signals of Industrial Control Systems (ICS) can be tuned by the attacker to cause a continuous spectrum in damages. Attackers that want to remain undetected can attempt to hide their manipulation of the system by following closely the expected behavior of the system, while injecting just enough false information at each time step to achieve their goals. In this chapter, we first study if attack-detection can limit the impact of such stealthy attacks. We propose a new metric to measure the impact of stealthy attacks and how they relate to our selection on an upper bound on false alarms. We finally show that the impact of such attacks can be mitigated in several cases by the proper combination and configuration of detection schemes. We demonstrate the effectiveness of our algorithms through simulations and experiments using real ICS testbeds and real ICS systems. This chapter comprehends the contributions of the following publication:


Chapter 5 - Improving Visibility of Network Monitoring. Monitoring the sensor and control values of control systems to identify abnormal behavior and attacks is a growing area of research. 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. The location of the monitor is particularly important, because an attacker could bypass the attack detection by carefully restricting her attacks to unobserved network segments, and ensuring that the detection monitor observes unchanged data. In this chapter, we analyze the problem of security monitor placement in industrial control networks, and show that there are locations that allow to detect low-level attacks. Based on our analysis, we design a novel 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. This chapter comprehends the contribution of the following publication:


Chapter 6 - Analysis of Encrypted Signals. Encryption is a key ingredient in the preservation of the confidentiality of network communications but can also be at odds with the mission of intrusion detection systems (IDSes) to monitor traffic. This affects Advanced Metering Infrastructures (AMIs) too where the scale of the network and the sensitivity of communication make deploying IDSes along with encryption solutions mandatory. In this chapter, we study four different approaches for reconciling the twin goals of confidentiality and monitoring by investigating their practical use on a set of real-world packet-level traces collected at an operational AMI network. This chapter comprehends the contribution of the following publication:


Chapter 7 - Conclusions and Future Work. This chapter closes this dissertation by summarizing our main contributions and suggestions for future work directions.
CHAPTER 2
SYSTEMATIZATION OF KNOWLEDGE

2.1 Introduction

Embedded computers and networks are monitoring and controlling systems that interact with the physical world. Examples include the smart grid, transportation systems, medical devices, building controls, manufacturing, and industrial control systems. While some of these infrastructures have used sensing and control for more than a century—the steam governor was introduced in 1788, and the first programmable controller (the Modicon 0844) was released in 1969—it is only in the last decade that new technological advances, combined with drastic reductions in the cost of deploying (1) wireless sensors (e.g., wirelessHART, ISA-100), (2) embedded computers, and (3) communication networks have enabled the ubiquitous use of networking and embedded devices in diverse sectors. This close integration creates new opportunities to improve efficiency and functionality in traditional control systems (e.g., power grid or industrial control systems) and also enables new use-cases (adaptive cruise control or new medical devices).

While the ubiquitous use of embedded smart sensors and controllers is needed for productivity and growth, they also introduce the security risks associated with information technology. Just a few decades ago most of these control systems were analog and controlled with technologies such as relay panels. To change the logic of a controller engineers had to be physically present, rewire the systems, and physically introduce new components such as relays and contacts. Now, new embedded controllers and sensors are basically computers that can be reprogrammed to execute arbitrary logic. In addition, several of the interactions of these embedded systems with the physical world can be labeled as safety-critical: their failure can cause irreparable harm to the physical system being controlled and to the people

---


who depend on it. The security of these systems is therefore, a challenge that needs to be addressed to achieve the full economic benefits of these technologies.

One of the fundamentally unique properties of devices monitoring and controlling the physical world—when compared to general Information Technology (IT) systems—is that the physical evolution of the state of a system has to follow immutable laws of nature. For example, the physical properties of water systems (fluid dynamics) or the power grid (electromagnetics) can be used to create time series models that we can then use to confirm that the control commands sent to the field were executed correctly and that the information coming from sensors is consistent with the expected behavior of the system. For example, if we open an intake valve we should expect that the water level in the tank should rise, otherwise we may have a problem with the control, actuator, or the sensor; this anomaly can be either due to an attack or a faulty device.

The idea of using physics-based attack detection has been presented in an increasing number of publications appearing in security conferences. Applications include water control systems [53], state estimation in the power grid [88, 89], boilers in power plants [146], chemical process control [29], capturing the physics of active sensors [128], electricity consumption data from smart meters [93], and other control systems [95].

The growing number of publications in the last couple of years clearly shows the growing importance of leveraging the physical properties of control systems for security; however, we have found that most of the works focusing on this topic are presented independently, with little context to related work. Therefore, research results are presented with different models, different evaluation metrics, and different experimental scenarios. This disjoint presentation of ideas is a limitation for creating the foundations necessary for discussing results in this field and for evaluating new proposals.

In particular, we found in our literature review that the works published in security conferences cite at most three other physics-based attack detection works (those in Table 2.1); in contrast, in this chapter we include over 45 physics-based attack detection works. This is, as far as we are aware, the first attempt to try to present these diverse and large set of works in a unified way.

Because of the interdisciplinary nature of this field, our survey includes works from fields that do not usually interact, such as control theory journals, information security conferences, and power system journals. We identify the relationships and trends in these fields to facilitate interactions among researchers of different disciplines. One of our goals is to provide basic foundations so that state of the art research in physics-based attack detection in control
systems or power system can be readily understood by information security practitioners and vice versa.

Our taxonomy is based on four main aspects: (1) model for physical system, (2) trust model, (3) detection mechanism proposed, and (4) evaluation metrics. To introduce these systems in a unified language, we had to introduce new terminology in an attempt to characterize unique differences between works and proposals, such as the difference between stateless vs. stateful anomaly detection tests.

Based on our review of the work from different domains, we present an analysis of the implicit assumptions made in works and the trust placed on embedded devices, and a logical detection architecture that can be used to elucidate hidden assumptions, limitations, and possible improvements to each work. We also show trends on which models, statistics and metrics are being used in this field, discuss advantages and disadvantages of each, and propose new research directions.

The remainder of this chapter is organized as follows: The scope of this chapter is presented explicitly in § 2.2. In § 2.3, we provide an introduction to control systems, and present the taxonomy we will use in this chapter to classify related work. We apply our taxonomy to a comprehensive set of related work in § 2.4. In § 2.5, we summarize our findings from related work, point out common shortcomings, and propose new directions of research. We conclude the chapter in § 2.6.

2.2 Scope of Our Study

There is a growing literature on the security of Cyber-Physical Systems (CPS), including the verification of control code by an embedded system before it reaches the Programmable Logic Controller (PLC), Remote Terminal Unit (RTU), or Intelligent Electronic Device (IED) [98], security of embedded devices [83], the automatic generation of malicious PLC payloads [96], security of medical devices [124], vulnerability analysis of vehicles [32, 71, 61], and of automated meter readings [123, 6]. There is also ongoing research on CPS privacy including smart grids [63], vehicular location monitoring [57], and location privacy [127]. We consider those works related, but complementary to our work.

One of the first works to consider intrusion detection in industrial control networks was Cheung et al. [33]. 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 work has been done in smart grid networks [6, 24] and in general CPS systems [100]; however, as Hadvziosmanovic et al. showed [52], intrusion
detection systems that fail to incorporate domain-specific knowledge and the context in which they are operating, will still perform poorly in practical scenarios. Even worse, an attacker that has obtained control of a sensor, an actuator, or a PLC can send manipulated sensor or control values to the physical process while complying to typical traffic patterns such as Internet Protocol (IP) addresses, protocol specifications with finite automata or Markov models, connection logs, etc.

In contrast to work in intrusion detection that focuses on monitoring such low-level IT observations, in this chapter we systematize the recent and growing literature in computer security conferences (e.g., CCS’15 [128], CCS’09 [88], ACSAC’13[95], ACSAC’14 [53], ASI-ACCS’11 [29], and ESORICS’14 [146]) studying how monitoring sensor values from physical observations, and control signals sent to actuators, can be used to detect attacks. We also systematize similar results by other fields like control theory conferences with the goal of helping security practitioners understand recent results from control theory, and control theory practitioners understand research results from the security community. Our selection criteria for including a work in the survey is to identify all the works (that we are aware of) where the system monitors sensor and/or control signals, and then raises an alert whenever these observations deviate from a model of the physical system.

2.3 Background and Taxonomy

We now briefly introduce control systems, common attacks, and countermeasures proposed in the literature. Then, we present the taxonomy that we will apply to review related work in § 2.4.

2.3.1 Background on Control Systems

Industrial Control Systems (ICS) are organized in a layered hierarchy [148], with the highest layer consisting of the Supervisory Control Network (SCN) and the lowest layer, the Field Communications Network (FCN), focusing on the direct interaction with the physical system. At the FCN, actuators (pumps, valves, etc.) and sensors (thermometers, tachometers, etc.) communicate with a controller—PLC, a Remote Terminal Unit (RTU) or an Intelligent Electronic Device (IED) via time-critical communication protocols. This communication was traditionally done with serial or analog communications; however, as advances in embedded systems and wireless communications continue, new protocols connecting field devices to local controllers continue to emerge. There is a large variety of communication standards used in
Figure 2.1. Layered architecture of ICS systems. A control center coordinates the operation of a large-scale process through a Supervisory Control Network. Field devices such as PLCs, RTUs and IEDs interact directly with the plant, through the Field Communications Network, sending control signals $u$ to keep the state of the system $y$ in the condition desired by the control center.

different parts of a typical control network; creating thus an highly fragmented market. For a recent survey of computer communications in industrial control systems see Gaj et al. [46]. At the SCN, controllers (PLC, RTU, EID) are usually connected to other controllers and to Supervisory Control And Data Acquisition (SCADA) centers with Master Terminal Units (MTU) or Human Machine Interfaces (HMI) through other non-time-critical communications systems. A layout of these layers of the ICS architecture is shown in Figure 2.1. Any of these devices (actuators, sensors, PLCs, or even the control center) can be compromised.

Four main components are part of a general anomaly-detection architecture: (1) the physical phenomena of interest (sometimes called the “plant”), (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 am is 120KV), (3) based on the sensor measurements received $y_k$, the controller 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).

A general security monitoring architecture for control systems that looks into the “physics” of the system needs an anomaly detection system that receives as inputs the sensor measurements $y_k$ from the physical system and the control commands $u_k$ sent to the physical
system, and then uses them to identify any suspicious sensor or control commands is shown in Figure 2.2. We assume that the detection system is trusted, and that it can recognize unexpected behaviors and potentially take counter measures (we discuss limitations on the detectability of an attack in later sections).

Tapping the control and sensor signals being sent to the plant can be done at the SCN, or between field devices and the plant at the FCN. Although where to monitor these values will have important consequences on our trust model as we will explain later.

The idea of monitoring sensor measurements $y_k$ and control commands $u_k$ and to use them to identify problems with sensors, actuators, or controllers is not new. In fact, this is what the literature of fault-detection in dynamical systems has investigated for more than four decades [149, 48, 60]. Fault Detection, Isolation, and Reconfiguration (FDIR) methods are diverse, and encompass research on hardware redundancy (e.g., adding more sensors to detect faulty measurements, or adding more controllers and decide on a majority voting control) as well as software (also known as analytical) redundancy [60]. While fault-detection theory provides the foundations for security-related work, fault-detection systems were designed to detect and respond to equipment failures, random faults, and accidents, not attacks, and it is important to understand how these systems fail and how we can improve upon them.

The anomaly detection architecture of Figure 2.2 can detect a variety of attacks coming from compromised sensors, controllers or actuators. Figure 2.3 shows an attack in the sensor, where the real state of the system $z_k$ is not reported back to the controller, and instead a false
Figure 2.3. When one or more sensor signals are compromised (e.g., the sensor itself is compromised or the controller receives and accepts a sensor measurement from an untrusted entity) the sensor measurement used as an input to the control algorithm will be different from the real state of the measured variables \( y_k \neq z_k \).

measurement \( y_k \) is sent, which allows the attacker to deceive the controller about the real state of the plant. The underlying idea of the detection algorithm in this case is to identify that the state of the system received \( y_k \) does not correspond to the state that is expected based on the control command \( u_k \).

Figure 2.4 shows an attack on the actuator, which modifies the control command send to the plant. Note how the state of the system gets “tainted” by this attack on the actuator, resulting thus in a sensor signal \( y_k \) that will again be unexpected by the anomaly detection algorithm (since it does not match the expected state of the system given the control command \( u_k \)). Finally, the controller can be compromised as well, giving the attacker potentially unlimited control to manipulate the plant (see Figure 2.5). In this case the anomaly detector needs to have analytical redundancy to identify a control signal that does not match the current state of the system.

This last figure also captures the threat model from a malicious field controller (e.g., a PLC) and also a malicious control command or set point sent from the control center as seen in Figure 2.6: While the implementation might be different–one monitor is placed in the SCN and the other monitor on the FCN–the logical architecture–what the anomaly detection application sees–will be the same.

The detection block in Figures 2.2 and 2.5 is expanded in Figure 2.7 to illustrate several alternative algorithms we found in the literature. There are two blocks that are straightforward
Figure 2.4. When one or more actuation signals are compromised (e.g., the actuator itself is compromised or it receives and accepts a control command from an untrusted entity) the actuation to the plant will be different to the intended action by the controller: \( v_k \neq u_k \). This false actuation will in turn affect the measured variables of the plant \( z_k \) which in turn affect the sensor measurements reported back to the controller: \( y_k = z_k \).

Figure 2.5. When the controller is compromised, it will generate a control signal that does not satisfy the logic of the correct control algorithm: \( u_k \neq \mathcal{K}(y_k) \).
to implement: (1) The Control Logic block in Figure 2.7 is a redundant control algorithm (i.e., in addition to the controller of Figure 2.2) that checks if the controller is sending the appropriate $u_k$ to the field, and (2) The Safety Check block is an algorithm that checks if the predicted future state of the system will violate a safety specification (e.g., the pressure in a tank will exceed its safety limit).

Two particular blocks (the red and purple blocks and their associated equations—which are discussed in the next section) in Figure 2.7 feature significantly in our survey of related work and will require special attention:

1. **Prediction** (Physical Model): given input features (e.g., sensor $y_k$ and control commands $u_{k+1}$, or a history of previous sensor values) a model of the physical system will predict a future expected measurement $\hat{y}_{k+1}$.

2. **Anomaly detection** (Statistical Test): Given a time series of residuals $r_k$ (the difference between the received sensor measurement $y_k$ and the predicted/expected measurement $\hat{y}_k$), the anomaly detection test needs to determine when to raise an alarm.

### 2.3.2 Taxonomy

We now present our taxonomy for related work, based on four aspects: (1) model for physical system, (2) trust model, (3) detection mechanism proposed, and (4) evaluation metrics.

In the following, we assume that the control commands $u_k$, the sensor measurements $y_k$ and the hidden states of the system $x_k$ (variables of interest to the control operator, but that cannot be seen directly from sensor measurements) are vector quantities.

**Physical System Model: LDS or AR**

The model of how a physical system behaves can be developed from physical equations (Newton’s laws, fluid dynamics, or electromagnetic laws) or it can be learned from observations.
In the top, the Control Logic block is a redundant control (i.e., in addition to the controller of Figure 2.2) that can be used to check if the control commands are appropriate given the current state of the system. The middle row (Prediction, Residual Generation, and Anomaly Detection blocks) focuses on looking at the sensor values and raising an alarm if they are different to what we expect/predict ($\hat{y}_k$). In the final row, the Safety Check block focuses on identifying if a given state or control command violates a safety limit and then raises an alert.
through a technique called system identification [91, 15] (like machine learning but focusing on learning the physical properties of a system). In system identification one often has to use either Auto-Regressive Moving Average with eXogenous inputs (ARMAX) or linear state-space models. Two popular models used by the works we survey are Auto-Regressive (AR) models (e.g., used by Hadziosmanovic et al. [53]) and Linear Dynamical State-Space (LDS) models (e.g., used by PyCRA [128]). AR models are a subset of ARMAX models but without modeling external inputs or the average error and LDS are a subset of state space models.

If we only have output data (sensor measurements $y_k$), regression models like AR, ARMA, or ARIMA are a popular way to learn the correlation between observations. Using these models we can predict the next outcome. For example, for an Auto-Regressive (AR) model, the prediction would be

$$\hat{y}_{k+1} = \sum_{i=k-N}^{k} \alpha_i y_i + \alpha_0$$

(2.1)

where $\alpha_i$ are the coefficients learned through system identification and $y_i$ are the last $N$ sensor measurements—where the amount of parameters to learn $N$ can be also estimated to prevent over-fitting of the model using tools like Akaike’s Information Criteria (AIC). It is possible to obtain the coefficients $\alpha_i$, by solving an optimization problem that minimizes the residuals (e.g., least squares) [90].

If we have inputs (control commands $u_k$) and outputs (sensor measurements $y_k$) available, we can use subspace model identification methods, producing the following model:

$$x_{k+1} = Ax_k + Bu_k + \epsilon_k$$

$$y_k = Cx_k + Du_k + e_k$$

(2.2)

where A, B, C, and D are matrices modeling the dynamics of the physical system. Most physical systems are strictly causal and therefore typically $D = 0$. The control commands $u_k \in \mathbb{R}^p$ affect the next time step of the state of the system $x_k \in \mathbb{R}^n$ and sensor measurements $y_k \in \mathbb{R}^q$ are modeled as a linear combination of these hidden states. $\epsilon_k$ and $\epsilon_k$ are sensor and perturbation noise, and are are used to model uncertain perturbations to the system, or noisy measurements. To make a prediction, we i) first need $y_k$ and $u_k$ to obtain a state estimate $\hat{x}_{k+1}$ and ii) use the estimate to predict the next sensor measurement as $\hat{y}_{k+1} = C\hat{x}_{k+1}$ (if $D$ is not zero we also need $u_{k+1}$). Some communities adopt models that employ the observation equation from (2.2) (the second line) without the dynamic state equation (i.e., without considering how the control inputs affect the hidden state $x_k$). We refer to this special case of LDS as Static Linear State-space (SLS) model.
Trust Model

To evaluate attack detection schemes, it is important to explicitly state which components in the control loop (or complete system) need to be trusted in order to correctly detect attacks. We call such explicit assumptions a trust model, and summarize such explicit or implicit assumptions for the related work. The trust model is related to attacker models, that often explicitly specify which components can be compromised (or not). Devices that cannot be compromised are trustworthy, so both model views are certainly related. The attacker model is more focused on the attacker, and the trust model more focused on the system under attack. We discuss trust assumptions in § 2.5.2.

Detection Mechanism: Stateless or Stateful

Based on the observed sensor or control signals up to time $k$, we can use models of the physical system (e.g., AR or LDS) to predict the expected observations $\hat{y}_{k+1}$ (note that $\hat{y}_{k+1}$ can be a vector representing multiple sensors at time $k + 1$). The difference $r_k$ between the observations predicted by our model $\hat{y}_{k+1}$ and the sensor measurements received from the field $y_{k+1}$ is usually called a residual. If the observations we get from the sensors $y_k$ are significantly different from the ones we expect (i.e., if the residual is large), we can generate an alert. In a Stateless test, we raise an alarm for every single significant deviation at time $k$: i.e., if

$$|y_k - \hat{y}_k| = r_k \geq \tau$$  \hspace{1cm} (2.3)

where $\tau$ is a threshold.

In a Stateful test we compute an additional statistic $S_k$ that keeps track of the historical changes of $r_k$ (no matter how small) and generate an alert if $S_k \geq \tau$, i.e., if there is a persistent deviation across multiple time-steps. There are many tests that can keep track of the historical behavior of the residual $r_k$ such as taking an average over a time-window, an exponential weighted moving average (EWMA), or using change detection statistics such as the non-parametric CUMulative SUM (CUSUM) statistic.

The theory behind CUSUM assumes we have a probability model for our observations $r_k$ (the residuals in our case); this obscures the intuition behind CUSUM, so we focus on the non-parametric CUSUM (CUSUM without probability likelihood models) which is basically a sum of the residuals. In this case, the CUSUM statistic is defined recursively as $S_0 = 0$ and $S_{k+1} = (S_k + |r_k| - \delta)^\dagger$, where $(x)\dagger$ represents $\max(0, x)$ and $\delta$ is selected so that the expected value of $|r_k| - \delta < 0$ when there is no attack (i.e., $\delta$ prevents $S_k$ from increasing consistently under normal operation). An alert is generated whenever the statistic is greater than a previously defined threshold $S_k > \tau$ and the test is restarted with $S_{k+1} = 0$. 
Evaluation Metric

The evaluation metric is used to determine the efficacy of the proposed detection scheme. Ideally, the metric should allow for a fair comparison of different schemes that are targeting the same adversarial model for comparable settings. Common evaluation metrics are the number of false alerts, and the probability of detecting attacks. A parametric curve illustrating the trade-off of these two quantities is the Receiver Operating Characteristic (ROC) curve. A specific combination of these two metrics into a single quantity is the accuracy (correct classification) of the anomaly detector. Some works also show the performance of the residuals $r_k$ under attack to show that they behave differently as when there is no attack.

2.3.3 State Estimation

Figure 2.8. Whenever the sensor measurements $y_k$ do not observe all the variables of interest from the physical process, we can use state estimation to obtain an estimate $\hat{x}_k$ of the real state of the system $x_k$ at time $k$ (if we have a model of the system). State estimates can then be used for the control logic, for prediction (and therefore for bad data detection), and for safety checks.

Before we start our survey we also need some preliminaries in what state estimation is. Whenever the sensor measurements $y_k$ do not observe all the variables of interest from the physical process, we can use state estimation to obtain an estimate $\hat{x}_k$ of the real state of the system $x_k$ at time $k$ (if we have a model of the system). State estimates can then be used for the control logic, for prediction (and therefore for bad data detection), and for safety checks.

Recall that Equation (2.2) gives us the relationship between the observed sensor measurements $y_k$ and the hidden state $x_k$. The naive approach would assume the noise $e_k$ is zero and then solve for $x_k$: $x_k = C^{-1}(y_k - Du_k)$; however, for most practical cases this is not possible...
as the matrix $C$ is not invertible, and we need to account for the variance of the noise. The exact solution for this case goes beyond the scope of this chapter, but readers interested in finding out how to estimate the state of a linear dynamical system are encouraged to read about Luenberger observers [133] and the Kalman filter [147], which are used to dynamically estimate the system’s states without or with noise, respectively.

State estimates can be used for the following blocks of Figure 2.7 Control Logic, for Prediction (and therefore for bad data detection as well), and for Safety Checks, as illustrated in Figure 2.8.

2.4 Survey of Previous Work

In this section, we survey previous work and relate it to the general framework we have introduced.

2.4.1 Power Systems

Attacks on bad data detection. One of the most popular lines of work within the scope of our work is the study of false-data injection attacks to avoid being detected by bad data detection algorithms for state estimation in the power grid. In the power grid, operators need to estimate the phase angles $x_k$ from the measured power flow $y_k$ in the transmission grid. These bad data detection algorithms were meant to detect random sensor faults, not strategic attacks, and as Liu et al. [88, 89] showed, it is possible for an attacker to create false sensor signals that will not raise an alarm (experimental validation in software used by the energy sector was later confirmed [137]). Model of the Physical System: It is known that the measured power flow $y_k = h(x_k) + e_k$ is a nonlinear noisy measurement of the state of the system $x$ and an unknown quantity $e_k$ called the measurement error. Liu et al. considered the linear model where $y_k = Cx_k + e_k$, therefore this model of the physical system is the sensor measurement SLS model described by Equation (2.2), where the matrix $D$ is zero and without the dynamic state equation. Detection: the mechanism they consider is a stateless anomaly detection test, where the residual is $r_k = y_k - \hat{C}\hat{x}_k$, the state estimate is defined as $\hat{x}_k = (C^TW^{-1}C)^{-1}C^TW^{-1}y_k$ and $W$ is the covariance matrix of the measurement noise $e_k$. Note that because $r_k$ is a vector, the metric $|\cdot|$ is a vector distance metric, rather than the absolute value. This test is also illustrated in the middle row of Figure 2.8. Trust Model: The sensor data is manipulated, and cannot be trusted. The goal of the attacker is to create false sensor measurements such that $|r_k| < \tau$. Evaluation Metrics: The work focuses on how hard it is for the adversary to find attacks such that $|r_k| < \tau$. 


There has been a significant amount of follow up research focusing on false data injection for state estimation in the power grid, including the work of Dán and Sandberg [36], who study the problem of identifying the best \( k \) sensors to protect in order to minimize the impact of attacks (they assume the attacker cannot compromise these sensors). Kosut et al. [72] consider attackers trying to minimize the error introduced in the estimate, and defenders with a new detection algorithm that attempts to detect false data injection attacks. Liang et al. [85] consider the nonlinear observation model \( y_k = h(x_k) + e_k \). Further work includes [26, 126, 136, 68, 49, 145, 119].

**Automatic Generation Control.** Control centers in the power grid send Area Control Error (ACE) signals to ramp up or ramp down generation based on the state of the grid. Sridhar and Govindarasu [134] consider an ACE signal that cannot be trusted. **Model of the Physical System:** A historical model of how real-time load forecast affects ACE. **Detection:** The ACE computed by the control center (ACE\(_R\)) and the one computed from the forecast (ACE\(_F\)) are then compared to compute the residual. They add the residuals for a time window and then raise an alarm if it exceeds a threshold. **Trust Model:** The load forecast is trusted but the ACE signal is not. **Evaluation Metric:** False positive and false negative rates.

**Active monitoring.** While most of the works we consider in this survey use passive monitoring (they do not interfere with normal operation unless there is an alarm and a reconfiguration is triggered), the works of Morrow et al. [108] and Davis et al. [37] consider active monitoring, that is, they use the optional reconfiguration signal we defined in Figure 2.2 to change the system periodically, even if there are no indicators of attacks. The intuition behind this approach is that changing the input to the system at random will tell us if the adversary is simply replaying old measurements and is not aware of our random changes. The idea of active monitoring has also been proposed in other domains [102, 128, 144].

While the idea of perturbing the system to reveal attackers that don’t adapt to these perturbations is intuitively appealing, it also comes with an operational cost: the deviation of a system from an ideal operational state just to test if the sensors have been compromised might not sound appealing to control engineers and asset owners whose livelihood depends on the optimal operation of a system. However, there is another way to look at this idea: if the control signal \( u_k \) is intrinsically random (e.g., the control signal for frequency generators in the power grid need to react to constant changes in the power demand from consumers), then the system might already be intrinsically better suited to detect attacks via passive monitoring because the attacker cannot simply replay or otherwise ignore the control signals sent to the system.
2.4.2 Industrial Control Systems

**Real-world Modbus-based Detection.** Hadziosmanovic et al. [53] give us a good example of how to use Modbus (an industrial protocol) traces from a real-world operational system to detect attacks by monitoring the state variables of the system, including: constants, attribute data, and continuous data. We focus on their analysis of continuous data because this focuses on physics-based attack detection. **Model of the Physical System:** To model the behavior of continuous sensor observations $y_k$ like the water level in a tank or the water pressure in a pipe, the authors use an AR model as we described in Equation (2.1). **Detection:** The scheme raises an alert if (1) the measurement $y_k$ reaches outside of specified limits (this is equivalent to the Safety Check box in Figure 2.7) or (2) $y_k$ produces a deviation in the prediction $\hat{y}_k$ of the autoregressive model (noting that $r_k = y_k - \hat{y}_k$), this is the stateless statistical test from Figure 2.7. **Trust Model:** It is not clear where in the control architecture the real-world data trace was collected. Because deploying a large-scale collection of a variety of devices in a control network is easier at the supervisory control network, it is likely that the real-world traffic monitors data exchanged between the control centers and the PLCs. In this case the PLC must be trusted, and therefore the adversary must attack the actuators or the sensors. **Evaluation Metrics:** The work focuses on understanding how accurately their AR system models the real-world system and identifying the cases where it fails. They mention that they are more interested in understanding the model fidelity rather than in specific true/false alarm rates, and we agree with them because measuring the true positive rate would be an artificial metric. Understanding the model fidelity is implicitly looking at the potential of false alarms because deviations between predictions and observations during normal operations are indicators of false alarms. While this is a good approach for the exploratory data analysis done in the work, it might be misunderstood by future proposals. After all, the rule never raise an alert will have zero false alarms (but it will never detect any attack). We discuss this further in § 2.5.

**Attack Localization.** State Relation-based Intrusion Detection (SRID) [146] attempts to detect attacks, and then find the root cause of the attack in an industrial control system. SRID is an outlier in our survey, while most other works leverage system identification tools to learn the model of the system, and anomaly detection tests that look at residuals, SRID proposes new heuristics for modeling the system and detecting attacks. **Model of Physical System:** Instead of using a traditional and well-understood system identification approach to learn a model of the boiler simulator they study, they propose a set of heuristics they
name feedback correlations and forward correlations; however, we were not able to find a good justification as to why these heuristics are better than traditional system identification methods and fault localization tools. We recommend that when researchers propose new (previously untested tools), they explain why traditional tools are not suitable. Part of our motivation for writing this survey work is to help future researchers understand the large amount of related work, and the tools they can leverage for developing novel physics-based attack detection algorithms. Detection: SRID does not specify if they use control and sensor measurements for their anomaly detection, but from the description it appears they use only sensor measurements. SRID proposes a new bad data detection based on alternation vectors, which basically tracks the history of measured variables going up or down. If this time series is not an allowable trend (not previously seen) the detection test generates an alert. Trust Model: The sensors cannot be trusted, but the attacker sends arbitrary data that falls within the sensor’s valid range. Therefore, this attacker is not strategic and it behaves as a random fault in the sensor. Evaluation Metrics: SRID measures the successful attack detection rate and the false alarm rate.

Attack-Detection and Response. Cárdenas et al. [29] consider a chemical industrial control system. Model of the Physical System: The authors approximate the nonlinear dynamics of the chemical system with an input/output linear system, as we defined in Equation (2.2). Therefore this model captures the correlations among multiple different observations $y_k$ (with the matrix C) but also the correlation between input $u_k$ and output $y_k$ and is therefore a model that can match the fidelity of observations very closely. Detection: The authors use the linear system to predict $\hat{y}_k$ given the previous input $u_{k-1}$ and the previous measurement $y_{k-1}$ and then test whether or not the prediction is close to the observed measurement $r_k = y_k - \hat{y}_k$. They raise an alert if the CUSUM statistic (the stateful test of Figure 2.7) is higher than a threshold. Trust Model: One or more sensors are compromised, and cannot be trusted. The goal of the adversary is to violate the safety of the system: i.e., an attacker that wants to raise the pressure level in the tank above 3000kPa and at the same time remain undetected by the test. The actuators and the control logic are assumed to be trusted. Evaluation Metrics: The work proposes a control reconfiguration whenever an attack is detected, in particular a switch to open-loop control, meaning that the control algorithm will ignore sensor measurements and will attempt to estimate the state of the system based only on the expected consequences of its own control commands. As a result, instead of measuring the false alarm rate, the authors measure the impact of a reconfiguration triggered by a false alarm on the safety of the system—in other words, a false alarm must never drive
the system to an unsafe state (a pressure inside the tank greater than 3000kPa). To evaluate
the security of the detection algorithm, the authors also test to see if an attacker that wants
to remain undetected can drive the pressure inside the tank above 3000kPa.

**Detecting Safety Violations and Response.** Another work that proposes attack-
response (or control reconfiguration) is McLaughlin [95]. This chapter tackles the problem
of how to verify that control signals $u_k$ will not drive the system to an unsafe state, and if
they do, to modify the control signal and produce a reconfiguration control that will prevent
the system from reaching an unsafe state. The proposed approach, $C^2$, mediates all control
signals $u_k$ sent by operators and embedded controllers to the physical system. **System Model:**
$C^2$ considers multiple systems with discrete states and formal specifications, as such this
approach is better suited for systems where safety is specified as logical control actions instead
of systems with continuous states (where we would need to use system identification to learn
their dynamics). **Detection:** This threat model corresponds to the attack on control signals in
Figure 2.5. Their focus is not to detect if $u_k \neq K(y_k)$, but to check if $u_k$ will violate a safety
condition of the control signal or not; therefore their attack-detection algorithm corresponds
to the Safety Check block we introduced in Figure 2.7. **Trust Model:** McLaughlin mentions
that “$C^2$ mitigates all control channel attacks against devices, and only requires trust in
process engineers and physical sensors,” as such $C^2$ assumes trusted sensors and trusted
actuation devices (specifically stating trusted actuators is a missing trust assumption in their
model). $C^2$ is related to traditional safety systems for control like safety interlocks, and not
necessarily malicious attacks as there does not seem to be a difference between preventing
an unsafe accidental action to an unsafe malicious action. **Evaluation Metrics:** There are
three main properties that $C^2$ attempts to hold: 1) safety (the approach must not introduce
new unsafe behaviors, i.e., when operations are denied the ‘automated’ control over the plant
should not lead the plant to an unsafe state), 2) security (mediation guarantees should hold
under all attacks allowed by the threat model), and 3) performance (control systems must
meet real time deadlines while imposing minimal overhead).

**Detecting Malicious Control Commands.** Further work trying to understand the
consequences of potentially malicious control commands from the control center, and as such
they correspond to the attack on control signals in Figure 2.6 [113, 74, 87]. Their goal is to
understand safe sequences of commands, and commands that might create problems to the
system. For example, Lin et al. [87] consider contingency analyses to predict consequences of
control commands. A different line of work looks at the reachability of dynamical systems
given potentially malicious control commands (Figure 2.5). For example, Mitra et al. [101] combine the dynamics of the system with discrete transitions (finite state machines) such as interruptions. They show it is possible to determine the set of safe states, the set of reachable states, and invariant sets; therefore, if there is not an input that can drive the states out of the safety set, the model is safe. Finding these sets requires some relaxations and a good knowledge of the behavior and limitations of the system.

**Critical State Analysis.** Carcano et al. [28] propose a safety monitoring system similar to $C^2$ but without mediating control commands (and using the control command $u_k$ to predict the next state $\hat{y}_k$ to see if it violates a safety condition) or proposing any reconfiguration when a safety issue is detected. The proposed concept is to monitor the state of a system and raise alerts whenever it is in a critical state (or approaching a critical state). *Model of the Physical System:* the approach measures the distance of sensor measurements $y_k$ to a critical state $\hat{y}^c$: $d(y_k, \hat{y}^c)$. They do not learn the dynamics of the physical system and this can have serious consequences as for example the power grid can change the distance to a critical state almost immediately whereas chemical processes such as growing bacteria in anaerobic reactors can take days to drive a system state to an unsafe region. *Detection:* They raise an alert whenever the system is in a critical state and also log the packets that led the system to that state for forensic purposes. They only monitor $y_k$ not $u_k$, which as we will argue, is a suboptimal approach. *Trust Model:* Because the authors monitor Modbus commands, it is likely that their sniffer is installed at the Supervisory Control Network of Figure 2.1, and as we will show, this assumes a trusted PLC. They also assume trusted sensors. The simulated attacks consist of control commands that drive the system to unsafe states. *Evaluation Metrics:* they monitor the number of false alarms and the true positive rate. The detection algorithm can have missed positives (when an attack happened and was not detected) because of packet drops but it is not clear what a false alarm is in their case.

**Clustering.** Another approach to detect attacks in process control systems is to learn unsupervised clustering models containing the pair-wise relationship between variables of a process, and then identify potential attacks as anomalies that do not fit these clusters [78, 69]. These approaches are non-parametric, which have the advantage of creating models of the physical process without a priori knowledge of the physics of the process; however, a non-parametric approach does not have the fidelity to the real physics of the system as an LDS or AR model will have, in particular when modeling the time-evolution of the system or the evolution outside of a steady state.
2.4.3 Control Theory

There is a significant body of work in attack detection from the control theory community [79, 17, 99, 58, 18]. While the treatment of the topic is highly mathematical (a recent special issue of the IEEE Control Systems Magazine provides an accessible introduction to the topic [59]), we attempt to extract the intuition behind key approaches to see if they can be useful for the computer security community.

Most control works we reviewed look at models of the physical system satisfying Equation (2.2) because that model has proven to be very powerful for most practical purposes. In addition, most of the control theory works we reviewed assumed a stateless detection as in Equation (2.3). We think this bias towards the stateless test by the control theory community stems from the fact that the stateless test allows researchers to prove theorems and derive clean mathematical results. In contrast, providing such thorough theoretical analysis for stateful tests (e.g., CUSUM) can become intractable for realistic systems. We believe that this focus on strong analytical results prevents the use of stateful tests that effectively perform better in many practical cases.

Zero-Dynamics Attacks. One of the novel attacks proposed by the literature in control theory is the so called zero dynamics attacks. These attacks are interesting because they show that even without compromising sensors, attackers can mislead control systems into thinking they are at a different state. The attacks require the attacker to compromise the actuators, that the anomaly detection system monitors the legitimate control signal $u_k$ and the legitimate sensor signal $y_k$ (but it does not observe the compromised $v_k$), and a plant vulnerable to these attacks as such the threat model is the one described in Figure 2.4. (Zero dynamic attacks can also be launched by sensors if the plant is controlled via feedback control, but this is a technicality outside the scope of our work). Not all systems are vulnerable to these attacks, but certain systems like the quadruple tank process [64] can be (depending on the specific parameters).

One of the fundamental properties control engineers ask about Equation (2.2) is whether or not the system is Observable [133]. If it is observable, then we know that we can obtain a good state estimate $\hat{x}_k$ given the history of previous control inputs $u_k$ and sensor measurements $y_k$. Most practical systems are observable or are designed to be observable. Now, if we assume an observable system, then we can hypothesize that the only way to fool a system into thinking it is at a false state, is by compromising the sensors and sending false sensor readings. Zero-dynamics attacks are an example that this hypothesis is false [138, 139, 114].
Not all systems are vulnerable to these attacks, but certain systems like the quadruple tank process [64] can be (depending on the specific parameters).

Though zero-dynamics attacks are interesting from a theoretical point of view, most practical systems will not be vulnerable to these attacks (although it is always good to check these conditions). First, if the sensors monitor all variables of interest, we won’t need state estimation (although this might not be possible in a large-scale control system with thousands of states); second, even if the system is vulnerable to zero-dynamics attacks, the attacker has to follow a specific control action from which it cannot deviate (so the attacker will have problems achieving a particular goal—e.g., move the system to a particular state), and finally, if the system is minimum phase, the attacker might not be able to destabilize the system. In addition, there are several recommendations on how to design a control system to prevent zero-dynamic attacks [139].

**Combined use of Cyber and Physical Attacks.** Control theory works have also considered the interplay between physical attacks and cyber-attacks. In a set of works by Amin et al. [9, 10] the attacker launches physical attacks to the system (physically stealing water from water distribution systems) while at the same time it launches a cyber-attack (compromised sensors send false data masking the effects of the physical attack). We did not consider physical attacks originally, but we then realized that the actuation attacks of Figure 2.4 account for physical attack, as it is equivalent to the attacker inserting its own actuators, and therefore the real actuation signal \( v_k \) will be different from the intended control command \( u_k \). To detect these attacks, they propose the use of unknown input observers. If the attackers control enough actuation and sensor measurements, there is nothing the detector can do as the compromised sensors can always send false data to make the detector believe the system is in the state the control wanted it to go. These covert attacks have been characterized for linear [130] and nonlinear systems [131].

**Active Monitoring.** The idea of reconfiguring the control system by sending unpredictable control commands and then verifying that the sensor responds as expected is referred in our survey as active monitoring (see § 2.4.1). The work of Mo et al. [102, 103, 104] considers embedding a watermark in the control signal that can be checked by the sensor observations. This is useful for systems that remain constant for long periods of time (if they are in a steady state) and where there is a risk of replay attacks. By randomly perturbing the system, an analyst can see if the sensor values respond appropriately. The threat models of active monitoring usually assume that the control input cannot be seen by the attacker. More
precisely, their threat model does not consider an attacker that knows the dynamics of the system and that can also observe the control commands, because such an attacker can then craft an appropriate false sensor response that will not be detected by the security analyst.

**Energy-Based Attack Detection.** Another novel attack-detection mechanism using control theoretic ideas was proposed by Eyisi and Koutsoukos [43]. The main idea is that the energy properties of a physical system can be used to detect errors or attacks. Unlike observer-based detection (used by the majority of control works), their work uses concepts of energy (or passivity), which is a property of systems which consume but not produce net energy. In other words, the system is passive if it dissipates more energy than it generates. To use this idea for detecting attacks, the monitor function estimates the supplied energy (by control commands) and compares it to the energy dissipated and the energy stored in the system (which depend on the dynamics of the system). While the idea is novel and unique, it is not clear why this approach might be better than traditional residual-based approaches, in particular given that any attack impersonating a passive device would be undetected, and in addition, the designer needs more information. To construct an energy model, a designer needs access to inputs and outputs, the model of the system in state space (as in Equation (2.2), and functions that describe the energy dissipation of a system in function of the stored energy (energy function) and the power supply (supply function).

### 2.4.4 Miscellaneous Domains

There is a growing interest in using the physics-based attack detection in a variety of domains.

**Active Monitoring for Sensors.** Active monitoring has also been used to verify the freshness and authenticity of a variety of sensors [128] and video cameras [144]. PyCRA [128] uses an LDS model to predict the response of sensors and to compute the residual $r_k$, which is then passed to a stateful $\chi^2$ anomaly detection statistic. The attacker in PyCRA has a physical actuator to respond to the active challenge. The evaluation of the proposal focuses on computing the trade-off between false alarms and probability of detection (i.e., ROC curves).

Another active monitoring approach suggests *visual challenges* [144] in order to detect attacks against video cameras. In particular a trusted verifier sends a visual challenge such as a passphrase or Quick Response (QR) code to a display that is part of the visual field of the camera, and if the visual challenge is detected in the video relayed by the camera, the
footage is deemed trustworthy. The work considers an adversary model that knows all the
details of the system and tries to forge video footage after capturing the visual challenge.
The authors use the CUSUM statistic to keep track of decoding errors.

**Automated Vehicles.** Kerns et al. [67] consider how Global Positioning System (GPS)
spoofing attacks can take control over unmanned aircrafts. They use an LDS as a model of
the physical system, and then use a *stateless* residual (also referred to as innovations) test to
detect attacks. They show two attacks, one where the attacker is detected, and another one
where the attacker manages to keep all the residuals below the threshold while still changing
the position of the aircraft. Sajjad et al. [125] consider the control of cars in automated
platoons. They use LDS to model the physical system and then use a *stateful* test with a
fixed window of time to process the residuals. To evaluate their system they show that when
attacks are detected, the cars in the platoon can take evasive maneuvers to avoid collisions.

**Physics-based forensics.** Conotter et al. [34] propose to look at the geometry and physics
of free-falling projectiles to check if the motions of a moving object in videos are realistic or
fake. To detect implausible trajectories of objects they first, describe a simplified 3D physical
model of the expected trajectory and a simplified 2D imaging model. Then, they determine
if the image of the trajectory of a projectile motion is consistent with the physical model. A
contribution of the work is to show how a 3D model can be directly created from the 2D
video footage. Once a 3D model is created, it can be used to check against the physical model
to detect any deviations. The attacker is someone who uses sophisticated video editing tools
to manipulate a video of, for example, a person throwing a basketball to create a perfect,
spectacular shot. In this case, the forger has access to the 2D video footage and can re-process
it. The work does not focus on how the forgery is done, but assumes that a video can be
either fake or real, and the goal of the proposed approach is to determine the authenticity
of each video. However, note that only naive attackers were considered here. If the forger
is aware of such detection mechanism, it will try to manipulate the 2D image to conform
to the real 3D model. The evaluation metric computes the mean error between the pair of
representations of the projectile motion using Euclidean distance; so it is a stateful test. The
reason for using this “offline” test (and not quick detection statistics like the CUSUM) stems
from the fact that forgery detection does not need to be done in real-time, but it is mostly
done after the fact.
Electricity theft. There is also work on the problem of electricity theft detection by monitoring electricity consumption from deployed smart meters [93]. To model the electricity consumption the authors use ARMA models, which are output-only models similar to those in Equation (2.1). Since their detection is not done online (similar to the video forensics case), the detection test is not stateless but stateful (an average of the residuals), where the detector can collect a lot of data and is not in a rush to make a quick decision. The attacker has compromised one sensor (the smart meter at their home) and sends false electricity consumption. The evaluation metric is the largest amount of electricity that the attacker can steal without being detected.

Medical devices. Detection of attacks to medical devices is also a growing interest [54, 55]. Hei et al. [54] study overdose attacks/accidents for insulin pumps and employ a supervised learning approach to learn normal patient infusion patterns with dosage amount, rate, and time of infusion. The model of their physical system is done through a Support Vector Regression (SVR). Again, similar to all the works reviewed in this miscellaneous section focusing on off-line anomaly detection, the detection test is an average of the residuals. More specifically, they use the Mean Squared Error measuring the difference between the predicted and the real value before raising an alert.

2.5 Discussion and Suggestions

We apply our taxonomy to previous work in Table 2.1. We arrange works by conference venue (we assigned workshops to the venue that the main conference is associated with). We also assigned conferences associated with Cyber-Physical Systems Week (CPSWeek) to control conferences because of the overlap of attendees to both venues. We make the following observations: (1) the vast majority of prior work uses stateless tests; (2) most control and power grid venues use LDS (or their static counterpart SLS) to model the physical system, while computer security venues tend to use a variety of models; several of them are non-standard and difficult to replicate by other researchers; (3) there is no consistent metric used to evaluate proposed attack-detection algorithms; (4) most works focus on describing attacks to specific devices (i.e., devices that are not trusted) but they do not provide a fine-grain trust model that can be used to describe what can be detected and what cannot be detected when the adversary is in control of different devices; and (5) no previous work has validated their work with all three options: simulations, testbeds, and real-world data.
Table 2.1. Summary of taxonomy of related work on physics-based attack detection in control systems.

<table>
<thead>
<tr>
<th>Detection</th>
<th>Control</th>
<th>Smart/Power Grid</th>
<th>Security</th>
<th>Misc.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AR</td>
<td>●</td>
<td>❖</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>SLS</td>
<td>❖</td>
<td>❖</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>LDS</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>other</td>
<td>❖</td>
<td>❖</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>impact</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>statistic</td>
<td>❖</td>
<td>❖</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>TPR</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>PPR</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Not Trusted</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>sensors</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>actuators</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>controllers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Validation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>simulation</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>real data</td>
<td>❖</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>tested</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>Monitoring</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
</tbody>
</table>

Legend: ●: feature considered by authors, ❖: feature not explicitly stated or exhibits ambiguity, ◆: a windowed stateful detection method is used, ◇: passive monitoring, ◆: active monitoring, †: attacks are made on the communication layer, ‡: also considers physical attacks, * Evaluation options have been abbreviated in the table: Attack Impact, Statistic Visualization, True Positive Rate, False Positive Rate.
2.5.1 General shortcomings

1. **No Consistent Evaluation.** There is no common evaluation metric used across multiple works. Some works [146, 128] measure the accuracy of their anomaly detector by looking at the trade-off between the false alarm rate and the true positive rate (metrics that are commonly used in machine-learning, fault-detection, and some intrusion detection works), while others [53] argue that measuring the true positive rate is misleading because we do not have a variety of real-world attack samples, and therefore they argue that we should focus only on measuring the fidelity of their models (i.e., minimizing the false alarms). In addition, most works focusing on false data injection for state estimation in the power grid and most works in control theory tend to focus on developing new undetected attacks, and ignore the number of false alarms in their metrics.

2. **No Comparison among Different Models and Different Tests.** There is no systematic publication record that builds upon previous work. While previous work has used different statistical tests (stateless vs. stateful) and models of the physical system to predict its expected behavior (AR vs. LDS), so far they have not been compared against each other, or if a given combination of physical models with the appropriate anomaly detection test is the best fit.

3. **Lack of Trust Models.** Most works do not describe their trust models with enough precision. Information exchanged between field devices (sensor to controller and controller to actuator in Figure 2.2) is communicated through a different channel from information that is exchanged between controllers or between controller and the supervisory control center. works that monitor network packets in the supervisory control network [53] implicitly assume that the controller (PLC) they monitor is trusted, otherwise the PLC could fabricate false packets that the monitor expects to see, while at the same time sending malicious data to actuators (what Stuxnet did). Thus, we need to monitor the communication between field devices in order to identify compromised PLCs in addition to monitoring supervisory control channels to identify compromised sensors or actuators.

2.5.2 Recommendations

We now provide some recommendations and new research directions to address some of the general shortcomings we identified.
Metrics and Evaluation. One of the key observations of our literature review is that while there are a large number of similarities in the way multiple researchers from many different CPS domains and fields of expertise have proposed mechanisms to detect attacks; there is a wide variety of evaluation metrics used to measure the performance of systems. Other fields like machine learning, and even intrusion detection, tend to look at the trade-off between false alarms, and attack detection from real-world attacks. Even alternative metrics for evaluating the classification accuracy of intrusion detection systems can be shown to be a multi-criteria optimization problem between the false alarm rate, and the true positive rate [31], and all of them depend on the ability of a system to detect some attacks.

However, one of the differences between detecting attacks in control systems when compared to detecting attacks in general IT systems is that researchers do not have readily available data from attacks in the wild. Even if we test our algorithms on the few known examples (like Stuxnet), they are domain specific and it is not clear they will give insights into the evaluation other than to show that we can detect Stuxnet (which can be easily detected \textit{ex post}). For that reason, researchers need to generate novel attacks in their works, and the question we would like to address in this section is how to create attacks that are general enough to be applicable across multiple industrial control domains but that will also allow us to define an evaluation metric that is fair (and that is not biased to detect the specific attacks from the researchers).

It is clear that if we use in this field the true positive rate, then we need to generate an attack that will be detected. It is not clear if there can be a principled way of justifying the generation of an attack that will be detected as this implies our attacker is not adaptive and will not attempt to evade our detection algorithms. Some publications using the true positive rate [146, 28] generate their attacks as random signals (e.g., a sensor reporting random values instead of reporting the true state of the physical system). This type of non-strategic random failure is precisely what the fault-detection community has been working on for over 40 years [149]; with those attacks we are not advancing the state of the art on attack-detection, but rather reinforcing the fact that fault-detection works when sensor or control signals fail randomly.

On the other hand, most publications in Table 2.1 focus on metrics trying to show the performance of the system under attack (the impact of the attack, a figure of the historical behavior of the anomaly detection statistic, and the True Positive Rate (TPR)), but they ignore the performance of the anomaly detection system without attacks (i.e., the false positive rate (FPR)). We need to emphasize that the performance of a system is usually a trade-off between multiple competing objectives; at the end of the day, the impact of attacks, or
the performance statistic, or the TPR, will produce perfect scores if we simply use the following anomaly detection algorithm: *raise an alarm for everything*. As such, measuring only metrics focusing on attack detection risk being misleading on the eventual performance in a real-world setting where attacks are rare, and most of the events will be produced by the normal operation of the system.

As a summary, we need to design new metrics that capture both, (1) the ability to mitigate attacks, and (2) keep a low false alarm rate. Our recommendation is to focus on measuring the impact of the attack (to avoid the pitfalls of designing attacks that will be detected) while at the same time keeping track of the false positive rate.

**Comparisons Between Models.** The lack of a commonly-used metric is hurting progress in this field because without it we cannot compare if different proposals for models of the physical system or for statistical tests, perform better or not. Previous work has left the identification of a suitable metric as an afterthought, focusing instead on applying new detection algorithms to different datasets, network protocols (DNP3, Modbus, etc.) or simulation models.

In particular, we notice that for modeling the system, most control theory works focus on LDS, most power grid works on SLS as state estimation in the power grid is not done considering a dynamical system (for the vast majority of cases), and works published in security conferences have a wide variety of models being proposed. It would be interesting to see if these various models are indeed justifiably better than LDS (or its static SLS counterpart), or if the use of these more complicated models stems from the fact that in security conferences, works are not reviewed mostly for their new theorems and mathematical derivations, but for their practical impact.

Similarly, it is not clear is stateless anomaly detection statistics are better than stateful statistics. While the vast majority of previous work has focused on stateless statistics for simplicity, we believe that keeping track of the historical performance of the system should improve the performance of attack-detection, and therefore we would like to encourage more work looking at stateful statistics such as the CUSUM, EWMA, sliding windows, etc.

**Trust Models.** Recall that most control systems have in general a layered hierarchy [148], with the highest levels consisting of the Supervisory Control Network (SCN) and the lowest levels focusing on the Field Communications Network (FCN) with the physical system, as shown in Figure 2.1. While several of the interactions at the field-level are still analog
If we were to deploy our anomaly detection system in the SCN (which typically has network switches with mirror ports making it the easy choice), then a compromised PLC can send manipulated data to the FCN, while pretending to report that everything is normal back to the SCN. 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., $u^a \neq u$) to a field device. 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 in the field communications layer, the compromised PLC reported a false value $y_a \neq y$ to the control center (through the SCN) claiming that devices were operating normally.

By deploying our network monitor at the SCN, we are not able to detect compromised PLCs (unless we are able to correlate information from other trusted PLCs), or unless we receive (trusted) sensor data directly.

A number of works we analyzed did not mention where the monitoring devices will be placed, which makes it difficult to analyze the author’s trust model. For example, analyzing the DNP3 communications standard [87, 86] implicitly assumes that the monitoring device is placed in the SCN, where DNP3 is most commonly used, and this security monitor will thus miss attacks that send some values to the SCN, and others to the FCN (such as Stuxnet). Therefore, such works implicitly assume that the PLC is reporting truthfully the measurements it receives, and the control commands it sends to actuators. This weak attacker model limits the usefulness of the intrusion detection tool.

To mitigate such restrictions, we argue that anomaly detection monitors should (also) be used at the FCN to detect compromised PLCs, actuators, and sensors. Assuming the monitor is placed in the FCN, the selection of trusted components determines the kind of attacks that can be detected. Our analysis in §5 (see Table 5.1) shows that as long as you trust two components in the loop, it is possible to detect an attack on the remaining component.

2.6 Conclusions

In this chapter, we have identified a general anomaly detection framework that captures most of the proposals available in the literature. We also provide a comprehensive taxonomy of related work, discuss general shortcomings we identified, and proposed new directions of research.
As we discussed in § 2.1, we found in our survey in § 2.4 that the works from Table 2.1 published in security conferences cite at most three other physics-based attack detection works from the same table. This presentation of physics-based attack detection ideas without contextualizing them with the large body of previous work is a big limitation to progress in this field. To address this problem, and to help researchers identify related work, in our survey we include over 45 physics-based attack detection works presented under a unified taxonomy. We hope that our work can advance the discussion in this space, and help other researchers from different fields (security or control) understand what their peers are developing in their respective conferences and journals, and facilitate the collaboration of security and control engineers to develop the theoretical foundations, the common language, and the tools for physics-based attack detection.

There are many challenges for future research: as we discussed we need better metrics, a comparison between competing methods, and better trust models. Furthermore, we have focused on detection as the main goal in this chapter, but in practice an attacker might sacrifice detection for achieving a desired malicious objective, so we need to find new ways to prevent unsafe actions by attackers. Finally, while some works in our survey cover attack-response (i.e., reconfiguration of the system) this is an area that has received relatively less focus. How to respond to alerts will become a vital area of research once we agree on some of the attack detection fundamentals.
CHAPTER 3
ATTACKING FIELD COMMUNICATIONS NETWORKS

3.1 Introduction

In recent years, security threats to industrial control systems (ICS) have received an increasing amount of attention [5, 3, 77, 8, 19]. One area that has received particular attention are attacks against the integrity of sensor and control data in the system, also known as false data injection attacks [29, 89, 150, 73]. If the attacker manipulates or replaces sensor data reported from the field devices, the control algorithm will take actions based on an incorrect perception of the world, which will cause incorrect control decisions and potential catastrophic failures.

So far, most of the analysis on false data injection has focused on a wide range of suspected theoretical attacks and simulation studies, and only limited work has been published on the practical challenges for launching such attacks as access to real-world ICS is usually hard to obtain for security researchers.

In this chapter, we discuss practical attacks applied to a room-sized water treatment testbed (the SWaT testbed). The Secure Water Treatment (SWaT) testbed includes a complete physical process, the related industrial communication infrastructure, a Field Communications Network (FCN), and a Supervisory Control Network (SCN). This chapter is an extension of our previous work [14], where we launched Man-in-the-Middle (MitM) attacks at the SCN of the SWaT testbed. In that work, we attacked the physical system using Ettercap spoofing on the SCN level, which enabled us to modify the sensor information. In addition, we noted that Ettercap-based spoofing can easily be detected by more complex networking devices such as SDN-capable switches and their controllers.

In contrast to Antonioli and Tippenhauer [14], our proposed methodology attacks the FCN directly, which enables the adversary to have total control of the system operation, without taking into account any command coming from the higher levels such as the Human-Machine Interface (HMI). Such attacks on field communications are challenging due to the

---

ring topology and the device specific messages used in the field, requiring more specific knowledge of the network operation. We discuss practical challenges in setting up MitM attacks on a FCN using the EtherNet/IP protocol over an Ethernet ring (maintained with the Device-Level-Ring (DLR) protocol). Once established as a MitM attacker, we are able to launch a range of sensor and actuator attacks and demonstrate their efficacy.

We summarize our contributions as follows:

- We study field communications and implement a prototype Man-in-the-Middle (MitM) attack.
- We show how a MitM attacker can obtain sensor readings from eavesdropped packets, and craft her own spoofed sensor and actuator command traffic. We also provide details on data types and conversions required (e.g., from 4-20mA signal to physical measurements).
- We combine the MitM attack and detailed understanding of the EtherNet/IP protocol to demonstrate practical attacks on SWaT, and show their impact on the physical process.

This chapter is structured as follows: in § 3.2, we state the problem setting and attacker model. In § 3.3, we present our practical attacks and discuss them in detail. We conclude the chapter in § 3.4.

3.2 Background and Problem Formulation

In this section, we introduce Industrial Control Systems (ICS), their field networks, and the specific testbed we use for our experiments. Finally, we introduce our attacker model.

3.2.1 The SWaT Physical Process

The SWaT testbed is a water treatment plant which consists of 6 main processes to purify raw water. Each process possesses a PLC that receives the information from the sensors and compute the control actions to the actuators. SWaT is set up to have two different communication channels for many links: either wired (over IEEE 802.3 Ethernet) or wireless communications (using IEEE 802.11).

The main physical processes SWaT can be described as follows (see Figure 3.1):
Raw water (P1) In this process, raw water is stored. P1 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) system’s tank.

Pre-treatment (P2) While the water from P1 is pumped to the UF system, water quality properties are evaluated and pre-treated. Conductivity, pH, and ORP are measured to determine the activation of chemical dosing to maintain the quality of the water within desirable limits.

Ultra Filtration (P3) The ultra-filtration process 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.

Dechlorinization (P4) After the small residuals are removed by the UF system, the remaining chlorines are destroyed in the ultraviolet chlorine destruction unit and by dosing a solution of sodium bisulphite.

Reverse Osmosis (P5) The RO system is designed to reduce inorganic impurities by pumping the filtrated and dechlorinated water with a high pressure through semipermeable membranes.

RO final product (P6) The last part of the water treatment process consists on storing the RO product i.e., cleaned water ready to distribute. In the SWaT case, treated water is transferred again to the raw water tank in order to reuse it.

The SWaT testbed features distributed controls among the different process stages. Each stage is controlled by a PLC (with hot-redundant counterpart), and all PLCs and the
SCADA system are connected together through a common network (which we call Level 1 (L1) network). In addition, the PLCs are connected to local sensors and actuators through individual field rings called Level 0 (L0). We now give details of this network architecture.

### 3.2.2 Industrial Control Network

A modern Industrial Control System typically consists of several layers of networks. In the SWaT case of study, the industrial control network is illustrated in Figure 3.2. The physical process is measured by distributed sensors, and manipulated by actuators. These sensors and actuators in SWaT operate by receiving and sending analog signals (4mA). The analog signals are converted into digital signals by Remote Input/Output (RIO) modules. The digital signals are then encapsulated over field communication protocols (EtherNet/IP in our case over the L0 network Figure 3.2), and sent back and forth from PLCs. PLCs in turn communicate with a centralized Supervisory Control and Data Acquisition (SCADA) system with the L1 network in Figure 3.2. This central system contains the HMI and Historian. In this chapter, we want to show a systematic methodology to deploy cyber-attacks on ICS, with a focus on Field Communications Network.

The reliability of such field networks is of great concern to the plant operator. For that reason, ring topologies are a popular choice to implement these topologies. The ring topology can be seen in Figure 3.2 at the L0 network, where there is a ring between the RIO and a primary and a backup PLC. In particular, rings can tolerate faults such as the loss of a single ring segment, without losing connectivity between any of the participating devices. If the

![Figure 3.2. SWaT network architecture.](image-url)
communication uses Ethernet as medium, rings can be constructed using the device-level-ring (DLR) protocol.

In the context of an attacker who tries to insert itself as MitM, such ring topologies have interesting properties. In particular, if the attacker cuts the ring to insert her own device, the ring will automatically stop transmitting data through the “lost” segment. As a result, the attacker will not receive any traffic to eavesdrop on. In order to successfully complete the attacker, the attacker must “close” the ring again. We discuss that further in § 3.3.3.

3.2.3 Field Communications Network

![Diagram of SWaT's ring topology with an attacker inserted as Man-in-the-Middle.](image)

In this configuration, the attacker can eavesdrop and manipulate all traffic between RIO and primary PLC.

SWaT’s ring topology in the Field Communications Network contains the following four main devices (see Figure 3.3).

**Programmable Logic Controller (PLC)** This is the control device which receives sensors readings and emits control commands to the actuators. SWaT’s PLCs correspond to a chassis conformed by 6 components: 1756-PA2 Control Logix AC Power Supply Unit; 1756-L71 Control Logix 2MB Controller; 1756-EN2T Ethernet I/P Module (for communications at the Supervisory Control Network level); 1756-EN2TR Dual Port Ethernet I/P Module (for field communications at the ring level); 1756-RM2 Control Logix Redundancy Module; and a 1756-RMC1 Control Logix Redundancy Fiber Optic Cable. Each SWaT’s ring presents a redundant PLC, and on power up the system selects one as Primary, actively controlling
the physical process, and other as Secondary, shadowing the memory state of the Primary. From that moment on, the Primary is responsible for the monitoring and control of the field communications and the relaying/receiving of data from the SCADA network. If the Primary fails, the system automatically switches over the control to the Secondary.

**Remote I/O (RIO)** This device is the responsible for the translation of 4-20 mA signals to/from the actuators/sensors to a stream of bytes where each byte (or bit depending on the resolution of the I/O module) corresponds to an I/O signal of the system. For the communications between the PLC and RIO, the stream is encapsulated following the Common Packet Format of the EtherNet/IP specification [111] and transported through a wired connection. The RIO is modular, and in SWaT it consists of 4 components: 1794-AENTR Flex Dual Port Ethernet I/P Adapter (for field communications at the ring level); 1794-IB32 Flex 32 Points Digital Input Module (1-bit resolution per signal); 1794-OB16 Flex 16 Points Digital Outputs Module (1-bit resolution per signal); and 1794-IE12 Flex 12 Points Analog Input Module (16-bit 2’s complement per signal).

**Wireless Remote I/O (WRIO)** SWaT features a manual switch to turn parts of the system into “wireless mode”. If the field communications at the ring level are set to wireless mode, this WRIO takes over the responsibility of scanning the analog sensors and sending updates to the PLC. Its is important to highlight that in SWaT this wireless transmission contains exclusively the analog input signals and not the digital I/O signals. The Digital input and output signals are always transmitted through the wired ring network between the RIO and PLC.

**Wireless Access Point (WAP)** In wireless mode, the WRIO connects to a wireless access point connect to the EtherNet/IP ring thought a 1783-ETAP 3-ports switch.

**3.2.4 Attacker Model**

**Objective** The objective of an attacker depends on how much damage she wants to cause to the system. Our proposed methodology can be used to a) manipulate sensor readings that are reported from the sensors to the PLCs, and b) to manipulate control messages that are sent from the PLC to the actuators. Therefore, a wide number of attacks can be deployed, starting with a simple eavesdropping to sophisticated integrity persistent and stealthy attacks.
Resources  The attacker is assumed to either a) being able to physically access the network connecting the PLCs, sensors, and actuators, or b) able to fully compromise one of the devices attached to that network. In both cases, the attacker will have a device attached to the network, and can program the device to transmit arbitrary messages to the network, and to process any message it receives.

3.3 Attacking field Communications in SWaT

Based on the details we provided on SWaT, its field topologies, and the EtherNet/IP protocol, we now show results of practical MitM attacks. We start by introducing tools we used, and among them our custom *SWaT Assault* tool.

3.3.1 Tools

We used several tools to launch attacks against the field communications at the SWaT testbed:

**SWaT Assault**  We developed a command-line interpreter (CLI) application which includes a library of attack modules capable of launching diverse spoofing and bad-data-injection attacks against the sensor and actuator signals of the SWaT testbed. The attack modules can be loaded, configured, and run independently of each other, allowing the attack of sensors and actuators separately. Attack modules also can be orchestrated and assembled in teams in order to force more complex behaviors over the physical process, while maintaining a normal operational profile on the HMI. SWaT Assault consists of 439 lines of Python\(^2\) code and its only external dependencies are Scapy and NetFilterQueue.

**Scapy**  Making use of the Scapy\(^3\) packet manipulation program we developed a new protocol parser for the Rockwell Automation proprietary message protocol used for signal communication between the RIO and the PLC, and for the EtherNet/IP Common Packet Format wrapper that encapsulates it. This parser (which we chose to call SWaT message parser) is specific for the SWaT’s deployment (the SWaT Ring implementation makes use of User Datagram Protocol (UDP) for the transport of EtherNet/IP I/O implicit messages among ring devices) and its implementation follows SWaT’s Control Panels and Electrical

\(^2\)Python Language. Version 2.7. https://docs.python.org/2/

Drawings manual. Scapy was also used to sniff sensor readings from the EtherNet/IP Ring and to inject manipulated data on both, sensor readings and actuation commands. Our tool also automatically recomputes the data integrity checksums used by the Transport Layer protocol to match the false-data injection attack values.

**NetFilterQueue** In order to avoid duplication of packets and/or race conditions between original and injected packets, we employed the NetFilterQueue\(^4\) Python bindings for libnetfilter_queue to redirect all the EtherNet/IP I/O messages between PLC and RIO to a handling queue defined on the *mangle* table of the Linux firewall *iptables*. The queued packets are later modified using Scapy and the previously mentioned SWaT message parser, and finally released to reach their original destination i.e., PLC or RIO. 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 ring devices.

The command we use to queue packets for modification is the following:

```
iptables -t mangle -A PREROUTING -p udp --dport <port> -j NFQUEUE
```

**Wireshark** We used Wireshark\(^5\) to understand the nature of the communication between devices in the ring. We also used Wireshark together with the SWaT’s Control Panel and Electrical Drawings manual, to derive the exact structure of the EtherNet/IP-wrapped messages used in SWaT.

**Ettercap** We used Ettercap\(^6\), a Man-In-The-Middle attack suite, on our attempts to launch wireless attacks.

3.3.2 Differences Between Parsing Supervisory Network Packets vs. Field Packets

While the SWaT networks use EtherNet/IP at the supervisory as well as the field level, the encapsulated protocol is different at each level: the CIP protocol (see § A.3) is used as main

---

\(^4\) Python bindings for libnetfilter_queue. [https://github.com/fqrouter/python-netfilterqueue](https://github.com/fqrouter/python-netfilterqueue)

\(^5\) Wireshark Network Protocol Analyzer. [https://www.wireshark.org/](https://www.wireshark.org/)

\(^6\) Ettercap Project. [https://ettercap.github.io/ettercap/](https://ettercap.github.io/ettercap/)
data payload for device communications at the Supervisory Control Network layer, while a EtherNet/IP (see § A.3.1) device-dependent I/O implicit message payload is employed at the field Communications layer (see Figure 3.4).

Parsing and injection of manipulated data at the Supervisory Control level using CIP messages or at the field Communications level using EtherNet/IP device-dependent I/O implicit messages introduce different challenges and requirements to the attacker.

In one hand, CIP messages contain much more rich semantic information about the information being exchanged in the network, which can facilitate the understanding of the process by the attacker. CIP messages follow an object-oriented format, allowing for the transmission of a variable number of data with distinct types, which translates into highly structured packets of variable lengths (see Figure 3.5). The attacker must dissect each particular packet to extract the CIP object attributes containing the sensor or actuator data, which may be set at different offsets depending on the number and type of objects targeted by the packet.

On the other hand, at the field level of SWaT, EtherNet/IP device-dependent I/O implicit messages follow non-standard formats of fixed lengths, partially defined by the vendor and by the control system designer, and where the analog sensor signals are encoded using 4-20 mA measurements. The attacker therefore must have detailed knowledge and understanding of the system design a priori and implementation decisions, i.e., she must have access to the

Figure 3.4. EtherNet/IP encapsulation for CIP and custom SWaT Messages for Supervisory Network and field communications, respectively.
devices specifications, electrical drawings, and installation layouts in order to understand the information exchanged and manipulate the sensor readings and control commands.

3.3.3 MitM of field Communications

We attempt attacks on the wired and wireless mode field communications in SWaT. We now present results for both cases.

Wireless MitM

On our first try to sniff the field communications in the SWaT testbed we attempted a wireless MitM attack between the PLC and the WRIO. We set the field communications into wireless mode and use a laptop we connected to the WAP. Using Wireshark, we realized that we could already see one multicast EtherNet/IP connection, stacking EtherNet/IP over UDP, with the WRIO’s IP as source. This multicast connection corresponded to the analog input signal which, as by SWaT’s design, is the only signal transported in wireless mode. After switching on and off the wireless mode multiple times, we verified that the multicast address range corresponded to the Organizational Local Scope [51], as expected. The assigned UDP destination port was 2222, also defined in SWaT’s design.

It is important to highlight that the attacks presented in § 3.3.5 could be achieved with minimum technical requirements through a wireless MitM if the ICS’s design accounts for a complete wireless transmission of digital and analog signals. Unfortunately, the digital input and output are not reported via the wireless links, and thus cannot be compromised using the wireless MitM attack. In order to cope with this characteristic of SWaT’s design, we resorted to performing a wired MitM directly in the EtherNet/IP ring.
Wired MitM

We assume an attacker who is an insider or who is able to set a physical device at any point of the EtherNet/IP ring, between the PLC and the RIO. In our experiments, for the implementation of the wired MitM, we intercepted the field communications by adding a laptop with two Ethernet ports in a segment of the EtherNet/IP ring.

DLR must be taken into consideration when attempting a MitM in the EtherNet/IP ring. If the attacker uses a DLR-unaware device, as we did in our experiments, she must disable MAC learning and Spanning Tree Protocol when bridging both Ethernet ports. Failing in carefully addressing this requirement will result in the isolation of the EtherNet/IP ring segment, as the DLR supervisor will recognize it as broken, and ultimately, in the inability to sniff and inject data into the ring messages.

Our configuration for the Ethernet ports and bridge for launching the attacks is the following:

```
auto eth0             # Port 0
iface eth0 inet manual

auto eth1             # Port 1
iface eth1 inet manual

# Bridge between Port 0 and Port 1
auto br0
iface br0 inet manual
    bridge_ports eth0 eth1
    bridge_stp off     # Disabling STP
    bridge_ageing 0    # Disabling MAC learning
```

3.3.4 Parsing and injection of manipulated data in EtherNet/IP ring packets

In SWaT, we identify three different device-dependent I/O implicit messages: one for each I/O module conforming the RIO. Figure 3.6 shows the I/O implicit message for the analog input module. It consists of a stream of 24 bytes, corresponding to 12 analog inputs channels of 16-bits. The spare channels are not in use by SWaT’s current deployment. The digital input and output modules emit and receive bit streams, 32 bits and 16 bits respectively, where each bit corresponds to one digital signal. See Table 3.1 for details on RIO modules.
Figure 3.6. RIO’s Analog Input Module 12 input signals (16-bits 2’s complement per analog signal).

### Table 3.1. RIO I/O modules.

<table>
<thead>
<tr>
<th>Module</th>
<th>Signal size (bits)</th>
<th># signals</th>
<th>Avg. Freq. (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Input</td>
<td>1</td>
<td>32</td>
<td>50</td>
</tr>
<tr>
<td>Digital Output</td>
<td>1</td>
<td>16</td>
<td>60</td>
</tr>
<tr>
<td>Analog Input</td>
<td>16</td>
<td>12</td>
<td>80</td>
</tr>
</tbody>
</table>

The I/O implicit messages representing the analog signals are sent by the RIO to the PLC with an average frequency of 80 milliseconds. They transport the numeric representation of the 4-20 mA signals measured by the analog sensors. In order to scale back and forth the 4-20 mA signal to the real physical measurement we use Equation (3.1), which is a typical linear transformation (scaling and bias shift) to change the analog signal (4-20mA) into a physical meaningful quantity (e.g., the height of the water level in a tank being 0.5m). The constant values ($\text{RawMin, RawMax, EUMax, EUMin}$) depend on the deployment and the physical property being measured (we obtained the specific values for each constant from the HMI software of the testbed). Figure 3.7 shows an example for the scaling of the water flow in SWaT.
\[ Out = (In - RawMin) \times \frac{EUMax - EUMin}{RawMax - RawMin} + EUMin \] (3.1)

Figure 3.7. Scaling from 4-20 mA signals to water flow. The 4-20mA signal is scaled by the RIO to another value (7790 in this case) and this value sent over the network is the one we capture and convert to physical observations using Equation (3.1) with the respective constants for each signal.

3.3.5 Example: Stealthy Sensor Attack

We illustrate the feasibility of our proposed methodology by deploying a stealthy sensor attack in the first stage of the plant i.e., raw water storage. The raw water process consists of a storage tank with its water level sensor \( h_1 \), one valve that opens when \( h_1 < 0.5 \) m and closes when \( h_1 > 0.8 \) m, and one pump whose action depends on the UF process. As a safety mechanism, if the water level in tank 1 is below 0.25 m, the pump is immediately Off. The attacker’s goal is to overflow the water without being detected by a typical behavior-based detection mechanism using the “physics” of the system under control to identify anomalies. Using the methodology described above, we gain access to the ring and we are able to modify the sensor and actuator information by constructing appropriate packets.

Attacker Action

Figure 3.8 depicts how an attacker can gain access to the sensor measure \( h_i(k) \) and actuator command \( u_{i}^{nom}(k) \) packets and modify them. In this example, the attack consists on injecting false information to the level sensor in tank 1. In particular, data injected is computed using \( h_i^a(k) = h_i(k) + \delta(k) \). As result of this attack, the level of the water is decreasing all the time, i.e., \( \delta(k) - \delta(k - 1) = \Delta < 0 \) is the rate at which the sensor information is modified.
Figure 3.8. Detection mechanism and Man-In-The-Middle attack for SWaT. $h_1^{(k)}$ is the attacked water level measure, $u_1^{\text{nom}}(k)$, $u_1^{a}(k)$ are the real and attacked control command, respectively, for time instance $k$.

**Detection Mechanism**

The detection mechanism also known as bad-data detection uses the residuals $r_i(k)$, which consists on the difference between the sensor measure $h_i(k)$ and its estimated $\hat{h}_i(k)$, such that $r_i(k) = |h_i(k) - \hat{h}_i(k)|$. For a sensor attack occurring at a time instant $k^*$, the residuals are then $r_i(k) = |h_i^{a}(k) - \hat{h}_i(k)|$ for all $k \geq k^*$. In our work, the estimated states are obtained by obtaining a mathematical approximation of the system behavior with a Luenberger observer (we refer to [89, 92] for more details on system estimation and the bad-data detection method, which are out of the scope of this chapter). When $r_i(k) > \tau_i$ for $\tau_i > 0$, an alarm is triggered indicating the presence of an attack. The main property of this type of detection is that it is based on the physical properties of the system and it can detect changes that violate those physical properties. For instance, Figure 3.9 depicts how the detection mechanism is able to trigger alarms for $\tau_1 = 0.1$ when an attack induces a sudden change in the sensor measurements, $h_1^{a}(k^*) = 0.1$ m, which yields to $r_1(k) > 0.1$. However, an intelligent adversary can remain stealthy by causing small changes in the sensor information.

Figure 3.10 illustrates the effect of the stealthy sensor attack. Before the attack is launched, the valve was closed and the pump was ON, so the water level was decreasing. As soon as the attack starts, the sensor measurement received keeps indicating that the water level is always decreasing (a little bit faster because the pump is ON), such that the valve opens when it reaches its minimum level $h_1^{a}(k) = 0.5$ m. At that instant the pump continues pumping water such that the real water level remains constant for some time. When the pump stops ($h_1^{a}(k) < 0.25$ m), the tank starts filling up with water (the water level $h(k)$ starts increasing) but the false sensor reading keeps indicating low water levels. The valve never closes and eventually it will yield to an overflow. Due to the small rate of change, the bad-data detection with a threshold $\tau_1 = 0.01$ never detects the attack. Smaller thresholds can detect the attack but also increase the number of false alarms. Besides, even if $\tau_1$ is smaller, the attack can be designed to remain stealthy for small $\Delta$. 
Figure 3.9. Sensor attack in the water level $h_1(k)$. The attack is detected when $r_1(k) > 0.1$.

Figure 3.10. Effects of a stealthy sensor attack in the SWaT testbed. The PLC is connected to an intelligent verifier that analyzes the sensor measure and detects attacks based on sudden changes in the physical behavior. The attack was designed such that the bad-data detector with $\tau_1 = 0.01$ could not detect the attack.
3.4 Conclusions

In this chapter, we discussed practical MitM attacks on ICS field communications. In particular, we provided details on the typical topologies (in particular, DLR) and protocols used in such a setting (EtherNet/IP). We also discussed practical challenges in setting up MitM attacks and how to overcome them, and demonstrated results of our practical attacks. We have shown that, although EtherNet/IP can be used as the overall encapsulation protocol, the protocol selection, and therefore, parsing and injection of manipulated data on the encapsulated message depends on the control system layer. SWaT presents CIP for Supervisory Control Network communications and a device-specific I/O messages for field communications.

The MitM of an EtherNet/IP ring also presents several challenges: The attacker must verify that the attacking device incorporated into the ring does not break the ring permanently, as these events could potentially be monitored, i.e., the attacking device must not interfere with the DLR protocol. An attacker who successfully deploys a MitM device into a EtherNet/IP ring established for field communications must be assumed to have access to all digital and analog signals. She could inject manipulated data in all sensors and actuators monitored and controlled by the ring’s PLC at any time. Therefore, the adversary could effectively isolate and manipulate the physical process disregarding control actions sent by the control room or the PLCs. Using the proposed methodology to launch attacks, different detection mechanisms can be tested and improved. In addition, more robust communication infrastructures can be designed in order to decrease an attacker’s impact over the system.
CHAPTER 4

LIMITING THE IMPACT OF STEALTHY ATTACKS

4.1 Introduction

One of the fundamentally unique and intrinsic properties of Industrial Control Systems (ICS)—when compared to general Information Technology (IT) systems—is that changes in the system’s state must follow immutable laws of physics. For example, the physical properties of water systems (fluid dynamics) or the power grid (electromagnetics) can be used to create prediction models that we can then use to confirm that the control commands sent to the field were executed correctly and that the information coming from sensors is consistent with the expected behavior of the system: if we opened an intake valve, we would expect the water tank level to rise, otherwise we may have a problem with the control, actuator, or the sensor.

The idea of using physics-based models of the normal operation of control systems to detect attacks has been used in an increasing number of publications in security conferences in the last couple of years. Applications include water control systems [53], state estimation in the power grid [88, 89], boilers in power plants [146], chemical process control [29], capturing the physics of active sensors [128], electricity consumption data from smart meters [93], and a variety of industrial control systems [95].

The growing number of publications in the last couple of years shows the importance of leveraging the physical properties of control systems for security; however, we have found that most of the works focusing on this topic have not tried to quantify the negative impact that a stealthy attacker can achieve, and which previously proposed attack-algorithms perform better against this powerful attacker. In particular, the problem we consider is one where the attacker knows the attack-detection system is in place and bypasses it by launching attacks imitating our expected behavior of the system, but different enough that over long periods of time it can drive the system to an unsafe operating state. This attacker is quite powerful and

---

can provide an upper bound on the worst performance of our attack-detection tools; however, we have found that this type of attacker is rarely used in previous work.

The remainder of this chapter is organized as follows: In § 4.2, we provide the scope of study. Based on our Survey in § 2.4, we summarize our findings from the related work, point out common shortcomings, and propose several improvements in § 4.3. We introduce our attacker model and the need for new metrics in § 4.4. We introduce a way to evaluate the impact of undetected attacks and attack-detection systems in § 4.5, and then we use this adversary model and metric to evaluate the performance of these systems in physical testbeds, real-world systems, and simulations in § 4.6. We conclude the chapter in § 4.7.

4.2 Scope of Our Study

This chapter focuses on the problem of using real-time measurements of the physical world to build indicators of attacks. Our work is motivated by false sensor measurements [88, 135] or false control signals like manipulating vehicle platoons [47], manipulating demand-response systems [135], and the sabotage Stuxnet created by manipulating the rotation frequency of centrifuges [81, 45]. The question we are trying to address is how to detect these false sensor or false control attacks in real-time.

Contributions. (i) We propose a strong adversary model that will always be able to bypass attack-detection mechanisms and propose a new evaluation metric for attack-detection algorithms that quantifies the negative impact of these stealthy attacks and the inherent trade-off with false alarms. Our new metric helps us compare in a fair way previously proposed attack-detection mechanisms.

(ii) We compare previous attack-detection proposals across three different experimental settings: a) a testbed operating real-world systems, b) network data we collected from an operational large-scale Supervisory Control and Data Acquisition (SCADA) system that manages more than 100 Programmable Logic Controllers (PLCs), and c) simulations.

(iii) Using these three scenarios we find the following new results: (a) while the vast majority of previous work uses stateless tests on residuals, stateful tests are better in limiting the impact of stealthy attackers (for the same levels of false alarms), (b) limiting the impact of a stealthy attacker can also depend on the specific control algorithm used and not only on the attack-detection algorithm.
4.3 Limitations of Previous Work

After analyzing the results of applying our previously defined taxonomy (in § 2.3.2) to a large variety of previous work (as summarized in Table 2.1), we derive the following observations: (1) the vast majority of prior work use stateless tests; (2) most control and power grid venues use LDS (or their static counterpart SLS) to model the physical system, while computer security venues tend to use a variety of models, several of them are non-standard and difficult to replicate by other researchers; (3) there is no consistent metric or adversary model used to evaluate proposed attack-detection algorithms; and (4) no previous work has validated their work with all three options: simulations, testbeds and real-world data.

The first three observations (1-3) are related: while previous work has used different statistical tests (stateless vs. stateful) and models of the physical system to predict its expected behavior, so far they have not been compared against each other, and this makes it difficult to build upon previous work (it is impossible to identify best practices without a way to compare different proposals). To address this problem we propose a general-purpose evaluation metric in § 4.5 that leverages our stealthy adversary model, and then compare previously proposed methods. Our results show that while stateless tests are more popular in the literature, stateful tests are better to limit the impact of stealthy attackers. In addition, we show that LDS models are better than AR models, that AR models proposed in previous work can be improved by leveraging correlation among different signals, and that having an integral controller can limit the impact of stealthy actuation attacks.

To address point (4) we conduct experiments using all three options: a testbed with a real physical process under control § 4.6.1, real-world data § 4.6.2, and simulations § 4.6.3. We show the advantages and disadvantages of each experimental setup, and the insights each of these experiments provide.

4.4 Motivating Example

The testbed we use for our experiments is a room-size, water treatment plant consisting of 6 stages to purify raw water. The testbed has a total of 12 PLCs (6 main PLCs and 6 in backup configuration to take over if the main PLC fails).

The general description of each stage is as follows: 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. This stage is illustrated in Figure 4.1 and will be used in our motivating example. 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. 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. Finally, in RO final product stage stores the RO product (clean water).

Attacking the pH level. In this process, the water’s pH level is controlled by dosing the water with Hydrochloric Acid (HCl). Figure 4.2 illustrates the normal operation of the plant: if the pH sensor reports a level above 7.05, the PLC sends a signal to turn On the HCl pump, and if the sensor reports a level below 6.95, it sends a signal to turn it Off. The wide oscillations of the pH levels occur because there is a delay between the control actions of the HCl pump, and the water pH responding to it.

To detect attacks on the PLC, the pump or the sensor, we need to create a model of the physical system. While the system is nonlinear, let us first attempt it to model it as time-delayed LDS of order 2. The model is described by \( \text{pH}_{k+1} = \text{pH}_k + u_{k-T_{\text{delay}}} \), where we estimate (by observing the process behavior) \( u_{k-T_{\text{delay}}} = -0.1 \) after a delay of 35 time steps after the pump is turned On, and 0.1 after a delay of 20 time steps after it is turned Off. We then compare the predicted and observed behavior, and compute the residual. We then apply
a stateless, and a stateful test, if either of these statistics goes above a defined threshold, we raise an alarm.

We note that high or low pH levels can be dangerous. In particular, if the attacker can drive the pH below 5, the acidity of the water will damage the membranes of the Ultra Filtration and Reverse Osmosis stages, the pipes, and even sensor probes.

We launch a wired Man-In-The-Middle (MitM) attack between the field devices (sensors and actuators) and the PLC by injecting a malicious device in the EtherNet/IP ring of the testbed, given that the implementation of this protocol is unauthenticated. A detailed implementation of our attack is given in our previous work [142]. In particular, our MitM intercepts sensor values coming from the HCL pump and the pH sensor, and intercept actuator commands going to the HCl pump, to inject false sensor readings and commands sent to the PLC and HCl pump.

Our attack sends false sensor data to the PLC, faking a high pH level so the pump keeps running, and thus driving the acidity of the water to unsafe levels, as illustrated in Figure 4.3 (left). Notice that both, stateless and stateful tests detect this attack (each test has a different threshold set to maintain a probability of false alarm of 0.01). We also launched an attack on the pump (actuator). Here the pump ignores Off control commands from the PLC, and sends back messages stating that it is indeed Off, while in reality it is On. As illustrated in Figure 4.3 (right), only the stateful test detects this attack. We also launched several random attacks that were easily detected by the stateful statistic, and if we were to plot the ROC curve of these attacks, we would get 100% detection rate.

**Observations.** As we can see, it is very easy to create attacks that can be detected. Under these simulations we could initially conclude that our LDS model combined with the stateful anomaly detection are good enough; after all, they detected all attacks we launched. However,
are these attacks enough to conclude that our LDS model is good enough? And if these attacks are not enough, then which types of attacks should we launch?

Notice that for any physical system, a sophisticated attacker can spoof deviations that follow relatively close the “physics” of the system while still driving the system to a different state. How can we measure the performance of our anomaly detection algorithm against these attacks? How can we measure the effectiveness of our anomaly detection tool if we assume that the attacker will always adapt to our algorithms and launch an undetected attack? And if our algorithms are not good enough, how can we design better algorithms? If by definition the attack is undetected, then we will always have a 0% true positive rate, therefore we need to devise new metrics to evaluate our systems.

4.5 A Stronger Adversary Model

We assume an attacker that has compromised a sensor (e.g., pH level in our motivating example) or an actuator (e.g., pump in our motivating example) in our system. We assume that the adversary has complete system knowledge, i.e., she knows the physical model we use, the statistical test we use, and the thresholds we select to raise alerts. Given this knowledge, she generates a stealthy attack, where the detection statistic will always remain below the selected threshold.
While similar stealthy attacks have been previously proposed [89, 88, 36], in this chapter we extend them for generic control systems including process perturbations and measurement noise, we force the attacks to remain stealthy against stateful tests, and also force the adversary to optimize the negative impact of the attack. In addition, we assume our adversary is adaptive, so if we lower the threshold to fire an alert, the attacker will also change the attack so that the anomaly detection statistic remains below the threshold. This last property is illustrated in Figure 4.4.

Notice that this type of adaptive behavior is different from how traditional metrics such as ROC curves work, because they use the same attacks for different thresholds of the anomaly detector. On the other hand, our adversary model requires a new and unique (undetected) attack specifically tailored for every anomaly detection threshold. If we try to compute an ROC curve under our adversary model we would get a 0% detection rate because the attacker would generate a new undetected attack for every anomaly detection threshold.

This problem is not unique to ROC curves: most popular metrics for evaluating the classification accuracy of intrusion detection systems (like the intrusion detection capability, the Bayesian detection rate, accuracy, expected cost, etc.) are known to be a multi-criteria optimization problem between two fundamental trade-off properties: the false alarm rate, and the true positive rate [31], and as we have argued, using any metric that requires a true positive rate will be ineffective against our adversary model launching undetected attacks.

**Observation.** Most intrusion detection metrics are variations of the fundamental trade-off between false alarms and true positive rates [31], however, our adversary by definition will
never be detected so we cannot use true positive rates (or variations thereof). Notice however that by forcing our adversary to remain undetected, we are effectively forcing her to launch attacks that follow closely the physical behavior of the system (more precisely, we are forcing our attacker to follow more closely our Physical Model), and by following closer the behavior of the system, then the attack impact is reduced: the attack needs to appear to be a plausible physical system behavior. So the trade-off we are looking for with this new adversary model is not one of false positives vs. true positives, but one between false positives and the impact of undetected attacks.

**New Metric.** To define precisely what we mean by impact of undetected attack we select one (or more) variables of interest (usually a variable whose compromise can affect the safety of the system) in the process we want to control—e.g., the pH level in our motivating example. The impact of the undetected attack will then be, how much can the attacker drive that value towards its intended goal (e.g., how much can the attacker lower the pH level while remaining undetected) per unit of time.

![Tradeoff Curve of Detector 1
Tradeoff Curve of Detector 2
Less deviation = More Secure
Longer time between false alarms = More Usable
Security Metric: Maximum deviation imposed by undetected attacks per time unit
Usability Metric: Expected time between false alarms](image)

Figure 4.5. Proposed tradeoff metric. The y-axis is a measure of the maximum deviation imposed by undetected attacks per time unit $\Delta X/TU$, and the x-axis represents the expected time between false alarms $E[T_{fa}]$. Anomaly detection algorithms are then evaluated for different points in this space.

Therefore we propose a new metric consisting of the trade-off between the maximum deviation per time unit imposed by undetected attacks (y-axis) and the expected time between false alarms (x-axis). Our proposed trade-off metric is illustrated in Figure 4.5, and its comparison to the performance of Receiver Operating Characteristic (ROC) curves against our proposed adversary model is illustrated in Figure 4.6.
Figure 4.6. Comparison of ROC curves with our proposed metric: ROC curves are not a useful metric against a stealthy and adaptive adversary.

Notice that while the y-axis of our proposed metric is completely different to ROC curves, the x-axis is similar, but with a key difference: instead of using the probability of false alarms, we use instead the expected time between false alarms $\mathbb{E}[T_{FA}]$. This quantity has a couple of advantages over the false alarm rate: (1) it addresses the deceptive nature of low false alarm rates due to the base-rate fallacy [16], and (2) it addresses the problem that several anomaly detection statistics make a decision (“alarm” or “normal behavior”) at non-constant time-intervals.

We now describe how to compute the y-axis and the x-axis of our proposed metric.

4.5.1 Computing the X and Y axis of Figure 4.5

Computing Attacks Designed for the Y-axis of our Metric. The adversary wants to maximize the deviation of a variable of interest $y_k$ (per time unit) without being detected. The true value of this variable is $y_k, y_{k+1}, \ldots, y_N$, and the attack starts at time $k$, resulting in a new observed time series $y^a_k, y^a_{k+1}, \ldots, y^a_N$. The goal of the attacker is to maximize the distance $\max_i |y_i - y^a_i|$. Recall that in general $y_k$ can be a vector of $n$ sensor measurements, and that the attack $y^a_k$ is a new vector where some (or all) of the sensor measurements are compromised.

An optimal greedy-attack $(y^a_*)$ at time $k \in [\kappa, \kappa_f]$ (where $\kappa$ and $\kappa_f$ are the initial and final attack times, respectively), satisfies the equation: $y^a_{k+1} = \arg \max_{y^a_{k+1}} f(y^a_{k+1})$ (where $f(y^a_{k+1})$ is defined by the designer of the detection method to quantify the attack impact) subject to not raising an alert (instead of max it can be min). For instance, if $f(y^a_{k+1}) = \|y_{k+1} - y^a_{k+1}\|$, the greedy attack for a stateless test is: $y^a_{k+1} = \hat{y}_{k+1} \pm \tau$. The greedy optimization problem for an attacker facing a stateful CUSUM test becomes $y^a_{k+1} = \max\{y^a_{k+1} : S_{k+1} \leq \tau\}$. 
Because $S_{k+1} = (S_k + r_k - \delta)$ the optimal attack is given when $S_k = \tau$, which results in $y_{k+1}^a = \hat{y}_{k+1} \pm (\tau + \delta - S_k)$. For all attack times $k$ greater than the initial time of attack $\kappa$, $S_k = \tau$ and $y_{k+1}^a = \hat{y}_{k+1} \pm \delta$.

Generating undetectable actuator attacks is more difficult than sensor attacks because in several practical cases it is impossible to predict the outcome $y_{k+1}$ with 100% accuracy, given the actuation attack signal $v_k$ in Figure 2.4. For our experiments when the control signal is compromised in § 4.6.3, we use the linear state space model from Equation (2.2) to do a reverse prediction from the intended $\hat{y}_{k+1}^a$ to obtain the control signal $v_k$ that will generate that next sensor observation.

Computing the X-axis of our Metric. Most of the literature that reports false alarms uses the false alarm rate metric. This value obscures the practical interpretation of false alarms: for example a 0.1% false alarm rate depends on the number of times an anomaly decision was made, and the time-duration of the experiment: and these are variables that can be selected: for example a stateful anomaly detection algorithm that monitors the difference between expected $\hat{y}_k$ and observed $y_k$ behavior has three options with every new observation $k$: (1) it can declare the behavior as normal, (2) it can generate an alert, (3) it can decide that the current evidence is inconclusive, and it can decide to take one more measurement $y_{k+1}$.

Because the amount of time $T$ that we have to observe the process and then make a decision is not fixed, but rather is a variable that can be selected, using the false alarm rate is misleading and therefore we have to use ideas from sequential detection theory [65]. In particular, we use the average time between false alarms $T_{FA}$, or more precisely, the expected time between false alarms $E[T_{FA}]$. We argue that telling security analysts that e.g., they should expect a false alarm every hour is a more direct and intuitive metric rather than giving them a probability of false alarm number over a decision period that will be variable if we use stateful anomaly detection tests. This way of measuring alarms also deals with the base rate fallacy, which is the problem where low false alarm rates such as 0.1% do not have any meaning unless we understand the likelihood of attacks in the dataset (the base rate of attacks). If the likelihood of attack is low, then low false alarm rates can be deceptive [16].

In all the experiments, the usability metric for each evaluated detection mechanism is obtained by counting the number of false alarms $n_{FA}$ for an experiment with a duration $T_E$ under normal operation (without attack), so for each threshold $\tau$ we calculate the estimated time for a false alarm by $E[T_{fa}] \approx T_E/n_{FA}$. Computing the average time between false alarms in the CUSUM test is more complicated than with the stateless test. In the CUSUM
case, we need to compute the evolution of the statistic $S_k$ for every threshold we test, because once $S_k$ hits the threshold we have to reset it to zero.

**Algorithm 1:** Computing Y axis.

1. Define $f(y_{k+1}^a)$
2. Select $\tau_{set} = \{\tau_1, \tau_2, \ldots\}$, $\kappa$, $\kappa_f$, and $K_{set} = \{\kappa, \ldots, k_f - 1\}$
3. $\forall (\tau, k) \in \tau_{set} \times K_{set}$, find $y_{k+1}^a(\tau) = \arg \max_{y_{k+1}^a} f(y_{k+1}^a)$ s.t. Detection Statistic $\leq \tau$

4. $\forall \tau \in \tau_{set}$, calculate $y - axis = \max_{k \in K_{set}} f(y_{k+1}^a(\tau))$

**Algorithm 2:** Computing X axis.

1. Observations $Y^{na}$ with no attacks of time-duration $T_E$
2. $\forall \tau \in \tau_{set}$, compute

Detection Statistic: $D_S(Y^{na})$

Number of false alarms: $nFA(D_S, \tau)$

$x - axis = E[T_{fa}(\tau)] = T_E/nFA$

Notice that while we have defined a specific impact for undetected attacks in our y-axis for clarity, we believe that designers who want to evaluate their system using our metric should define an appropriate worst case undetected attack optimization problem specifically for their system. In particular, the y-axis can be a representation of a cost function $f$ of interest to the designer. There are a variety of metrics (optimization objectives) that can be measured such as the product degradation from undetected attacks, or the historical deviation of the system under attack $\sum_i |y_i - \hat{y}_i^a|$ or the deviation at the end of the attack $|y_N - \hat{y}_N^a|$, etc. A summary of how to compute the y-axis and the x-axis of our metric is given in Algorithms 1 and 2.
Table 4.1. Advantages and disadvantages of different evaluation setups.

<table>
<thead>
<tr>
<th>Reliability of:</th>
<th>X-Axis</th>
<th>Y-Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Data</td>
<td>●</td>
<td>○</td>
</tr>
<tr>
<td>Testbed</td>
<td>○○○</td>
<td>○○○</td>
</tr>
<tr>
<td>Simulation</td>
<td>○</td>
<td>●</td>
</tr>
</tbody>
</table>

● = well suited, ○ = partially suitable, ○ = least suitable

4.6 Experimental Results

We evaluate anomaly detection systems under the light of our Stronger Adversary Model (see § 4.5), using our new metrics in a range of test environments, with individual strengths and weaknesses (see Table 4.1). As shown in the table, real-world data is useful to consider operational large-scale scenarios, and therefore it is the best way to test the x-axis metric $E[T_{fa}]$. Unfortunately, real-world data does not give researchers the flexibility to launch attacks and measure the impact on all parts of the system. Such interactive testing requires the use of a dedicated physical testbed.

A physical testbed on the other hand, limits the range of experimental attacks that could potentially be performed. Its physical components and devices may suffer damage by attacks that violate the safety requirements and conditions for which they were designed for. Moreover, attacks could also drive the testbed to states that endanger the operator’s and environment’s safety. Therefore, while a testbed provides more experimental interaction than real data, it introduces safety constraints for launching attacks.

Simulations on the other hand, do not have these constraints and a wide variety of attacks can be launched. So our simulations will focus on attacks to actuators and demonstrate settings that cannot be achieved while operating a real-world system because of safety constraints. Simulations also allow us to easily change the control algorithms and to our surprise, we found that control algorithms have a big impact on the ability of our attacker to achieve good results in the y-axis of our metric. However, while simulations allow us to test a wide variety of attacks, the problem is that the false alarms measured with a simulation are not going to be as representative as those obtained from real data or from a testbed.

4.6.1 Physical Testbed (EtherNet/IP packets)

In this section, we focus on testbeds that control a real physical process, as opposed to testbeds that use a Hardware-In-the-Loop (HIL) simulation of the physical process. A HIL testbed is similar to the experiments we describe in § 4.6.3.
We developed an attacker who has complete knowledge of the physical behavior of the system and can manipulate EtherNet/IP packets and inject attacks. We now apply our metric to the experiments we started in § 4.4.

**Attacking pH Level.** Because this system is highly nonlinear, apart from the simple physical model (LDS) of order 2 we presented in § 4.4, we also applied a system identification to calculate higher order system models: an LDS model of order 20 and two nonlinear models (order 50 and 100) based on wavelet networks [117]. Figure 4.7 shows the minimum pH achieved by the attacker after 4-minutes and against three different models. Notice that the nonlinear models limited the impact of the stealthy attack by not allowing deviations below a pH of 5, while our linear model (which was successful in detecting attacks in our motivating example) was not able to prevent the attacker from taking the pH below 5.

![Figure 4.7. pH deviation imposed by greedy attacks while using stateful detection (τ = 0.05) and LDS and nonlinear models.](image)

Figure 4.8 illustrates the application of our proposed metric over 10 different undetected greedy attacks, each averaging 4 minutes, to evaluate the three system models used for detection. Given enough time, it is not possible to restrict a deviation of pH below 5. Nevertheless, for all $E[T_{fa}](min)$, the nonlinear model of order 100 performs better than the nonlinear model of order 50 and the LDS of order 20, limiting the impact of the attack per minute $\Delta_{pH}/min$. It would take over 5 minutes for the attacker to deviate the pH below 5 without being detected using a nonlinear model of order 100, whereas it would take less than 3 minutes with the nonlinear of order 50 and the LDS of order 20.
Attacking the Water Level. Now we turn to another stage in our testbed. The goal of the attacker this time is to deviate the water level in a tank as much as possible until the tank overflows.

While in the pH example we had to use system identification to learn LDS and nonlinear models, the evolution of the water level in a tank is a well-known LDS system that can be derived from first principles. In particular, we use a mass balance equation that relates the change in the water level $h$ with respect to the inlet $Q_{\text{in}}$ and outlet $Q_{\text{out}}$ volume of water, given by $\text{Area} \frac{dh}{dt} = Q_{\text{in}} - Q_{\text{out}}$, where $\text{Area}$ 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, $Q_{\text{in}}$ or $Q_{\text{out}}$ remain constant if they are open, and zero otherwise. Using a time-discretization of 1 s, we obtain an LDS model of the form

$$h_{k+1} = h_k + \frac{Q_{\text{in}} - Q_{\text{out}}}{\text{Area}}.$$  

Note that while this equation might look like an AR model, it is in fact an LDS model because the input $Q_{\text{in}} - Q_{\text{out}}$ changes over time, depending on the control actions of the PLC (open/close inlet or start/stop pump). In particular it is an LDS model with $x_k = h_k$, $u_k = [Q_{\text{in}}^k, Q_{\text{out}}^k]^T$, $B = [-\frac{1}{\text{Area}}, -\frac{1}{\text{Area}}]$, $A = 1$, and $C = 1$.

Recall that the goal of the attacker is to deviate the water level in a tank as much as possible until the tank overflows. In particular, the attacker increases the water level sensor...
signal at a lower rate than the real level of water (Figure 4.9) with the goal of overflowing the tank. A **successful attack** occurs if the PLC receives from the sensor a High water-level message (the point when the PLC sends a command to close the inlet), and at that point, the deviation ($\Delta$) between the real level of water and the “fake” level (which just reached the High warning) is $\Delta \geq \text{Overflow} - \text{High}$. Figure 4.9 shows three water level attacks with different increment rates, starting from the Low level setting and stopping at the High level setting, and their induced maximum $\Delta$ over the real level. Only attacks $a_1$ and $a_2$ achieve a successful overflow (only $a_2$ achieves a water spill), while $a_3$ deviates the water level without overflow. In our experiment, High corresponds to a water level of 0.8 m and Low to 0.5 m. Overflow occurs at 1.1 m. The testbed has a drainage system to allow attacks that overflow the tank.

![Figure 4.9. Impact of different increment rates on overflow attack. The attacker has to select the rate of increase with the lowest slope while remaining undetected.](image)

Because it was derived from “first principles”, our LDS model is a highly accurate physical model of the system, so there is no need to test alternative physical models. However, we can combine our LDS model with a stateless test, and with a stateful test and see which of these detection tests can limit the impact of stealthy attacks.

In particular, to compute our metric we need to test stateless and stateful mechanisms and obtain the security metric that quantifies the impact $\Delta$ of undetected attacks for several thresholds $\tau$. We selected the parameter $\delta = 0.002$ for the stateful (CUSUM) algorithm, such that the detection metric $S_k$ remains close to zero when there is no attack. The usability metric is calculated for $T_E = 8$ h, which is the time of the experiment without attacks.
Figure 4.10. Comparison of stateful and stateless detection. At 0.3 m the tank overflows, so stateless tests are not good for this use case. $\tau_b, \tau_c$ correspond to the threshold associated to some $E[T_{fa}]$.

Figure 4.10 illustrates the maximum impact caused by 20 different undetected attacks, each of them averaging 40 minutes. Even though the attacks remained undetected, the impact using stateless detection is such that a large amount of water can be spilled. Only for very small thresholds is it possible to avoid overflow, but it causes a large number of false alarms. On the other hand, stateful detection limits the impact of the adversary. Note that to start spilling water (i.e., $\Delta > 0.3 \text{ m}$) a large threshold is required. Clearly, selecting a threshold such that $E[T_{fa}] = 170 \text{ min}$ can avoid the spilling of water with a considerable tolerable number of false alarms.

In addition to attacking sensor values, we would like to analyze undetected actuation attacks. To launch attacks on the actuators (pumps) of this testbed, we would need to turn them On and Off in rapid succession in order try to maintain the residuals of the system low enough to avoid being detected. We cannot do this on real equipment because the pumps would get damaged. Therefore, we will analyze undetected actuator attacks with simulations (where equipment cannot be damaged) in § 4.6.3.

4.6.2 Large-Scale Operational Systems (Modbus packets)

We were allowed to place a network sniffer on a real-world operational large-scale water facility in the U.S. We collected more than 200GB of network packet captures of a system
using the Modbus/TCP [105] industrial protocol (see § A.1). Our goal is to extract the sensor and control commands from this trace and evaluate and compare alternatives presented in the survey.

The network has more than 100 controllers, some of them with more than a thousand registers. In particular, 1) 95% of transmissions are Modbus packets and the rest 5% corresponds to general Internet protocols; 2) the trace captured 108 Modbus devices, of which one acts as central master, one as external network gateway, and 106 are slave PLCs; 3) of the commands sent from the master to the PLCs, 74% are Read/Write Multiple Registers (0x17) commands, 20% are Read Coils (0x01) commands, and 6% are Read Discrete Inputs (0x02) commands; and 4) 78% of PLCs count with 200 to 600 registers, 15% between 600 to 1000, and 7% with more than 1000.

We replay the traffic traces in packet capture (pcap) format and use Bro\textsuperscript{2} [115] to track the memory map of holding (read/write) registers from PLCs. We then use Pandas\textsuperscript{3}, a Python Data Analysis Library, to parse the log generated by Bro and to extract per PLC the time series corresponding to each of the registers. Each time series corresponds to a signal ($y_k$) in our experiments. We classify the signals as 91.5% constant, 5.3% discrete and 3.2% continuous based on the data characterization approach proposed to analyze Modbus traces [53] and uses AR models (as in Equation (2.1)). We follow that approach by modeling the continuous time-series in our dataset with AR models. The order of the AR model is selected using the Best Fit criteria from the Matlab System Identification toolbox\textsuperscript{4}, which uses unexplained output variance, i.e., the portion of the output not explained by the AR model for various orders [94].

Using the AR model, our first experiment centers on deciding which statistical detection test is better, a stateless test or the stateful CUSUM change detection test. Figure 4.11 shows the comparison of stateless vs. stateful tests with our proposed metrics (where the duration of an undetected attack is 10 minutes). As expected, once the CUSUM statistic reaches the threshold $S_k = \tau$, the attack no longer has enough room to continue deviating the signal without being detected, and larger thresholds $\tau$ do not make a difference once the attacker reaches the threshold, whereas for the stateless test, the attacker has the ability to change the measurement by $\tau$ units at every time step.

\textsuperscript{2}The Bro Network Security Monitor. \url{https://www.bro.org/}

\textsuperscript{3}Pandas: Python Data Analysis Library. \url{http://pandas.pydata.org}

\textsuperscript{4}System Identification. \url{http://www.mathworks.com/help/ident/ref/systemidentification-app}
Having shown that a CUSUM (stateful) test reduces the impact of a stealthy attack when compared to the stateless test we now show how to improve the AR physical model previously used by Hadziosmanovic et al. [53]. In particular, we notice that Hadziosmanovic et al. use an AR model per signal; this misses the opportunity of creating models of how multiple signals are correlated, creating correlated physical models will limit the impact of undetected attacks.

![Figure 4.11. Stateful performs better than stateless detection: The attacker can send larger undetected false measurements for the same expected time to false alarms.](image)

**Spatial and Temporal Correlation** In an ideal situation the water utility operators could help us identify all control loops and spatial correlations of all variables (the water pump that controls the level of water in a tank etc.); however, this process becomes difficult to perform in a large-scale system with thousands of control and sensor signals exchanged every second; therefore we now attempt to find correlations empirically from our data. We correlate signals by computing the correlation coefficients of different signals $s_1, s_2, \ldots, s_N$. The correlation coefficient is a normalized variant of the mathematical covariance function: 

$$
corr(s_i, s_j) = \frac{\text{cov}(s_i, s_j)}{\sqrt{\text{cov}(s_i, s_i) \text{cov}(s_j, s_j)}}
$$

where $\text{cov}(s_i, s_j)$ denotes the covariance between $s_i$ and $s_j$ and correlation ranges between $-1 \leq corr(s_i, s_j) \leq 1$. We then calculate the \textit{p-value} of the test to measure the significance of the correlation between signals. The \textit{p-value} is the probability of having a correlation as large (or as negative) as the observed value when the true correlation is zero (i.e., testing the null hypothesis of no correlation, so lower values of $p$
Figure 4.12. Three example signals with significant correlations. Signal $S_{16}$ is more correlated with $S_{19}$ than it is with $S_8$.

indicate higher evidence of correlation). We were able to find 8,620 correlations to be highly significant with $p = 0$. Because $\text{corr}(s_i, s_j) = \text{corr}(s_j, s_i)$ there are 4,310 unique significant correlated pairs. We narrow down our attention to $\text{corr}(s_i, s_j) > .96$. Figure 4.12 illustrates three of the correlated signals we found. Signals $s_{16}$ and $s_{19}$ are highly correlated with $\text{corr}(s_{16}, s_{19}) = .9924$ while $s_8$ and $s_{19}$ are correlated but with a lower correlation coefficient of $\text{corr}(s_8, s_{19}) = .9657$. For our study we selected to use signal $s_8$ and its most correlated signal $s_{17}$ which are among the top most correlated signal pairs we found with $\text{corr}(S_8, S_{17}) = .9996$.

Figure 4.13. Using the defined metrics, we show how our new correlated AR models perform better (with stateless or stateful tests) than the AR models of independent signals of previous work.

Our experiments show that an AR model trained with correlated signals (see Figure 4.13) is more effective in limiting the maximum deviation the attacker can achieve (assuming the
attacker only compromises one of the signals). For that reason, we encourage future work to use correlated AR models rather than the previously proposed AR models of single signals.

4.6.3 Simulations of the Physical World

Simulations allow us to launch actuator attacks without the safety risk of damaging physical equipment. In particular, in this section we launch actuation attacks and show how the control algorithm used can significantly limit the impact of stealthy attackers. In particular we show that the Integrative part of a Proportional Integral Derivative (PID) control algorithm (or a PI or I control algorithm) can correct the deviation injected by the malicious actuator, and force the system to return to the correct operating state.

We use simulations of primary frequency control in the power grid as this is the scenario used by the Aurora attack [152]. Our goal is to maintain the frequency of the power grid as close as possible to 60Hz, subject to perturbations—i.e., changes in the Mega Watt (MW) demand by consumers—and attacks.

We assume that the attacker takes control of the actuators. When we consider attacks on a control signal, we need to be careful to specify whether or not the anomaly detection system can observe the false control signal. In this section, we assume the worst case: our anomaly detection algorithm cannot see the manipulated signal and indirectly observes the attack effects from sensors (e.g., \(v_k\) is controlled by the attacker, while the detection algorithm observes the valid \(u_k\) control signal, see Figure 2.4).

Attacking a sensor is easier for our stealthy adversary because she knows the exact false sensor value \(\hat{y}\) that will allow her to remain undetected while causing maximum damage. On the other hand, by attacking the actuator the attacker needs to find the input \(u_k\) that deviates the frequency enough, but still remains undetected. This is harder because even if the attacker has a model of the system, the output signal is not under complete control of the attacker: consumers can also affect the frequency of the system (by increasing or decreasing electricity consumption), and therefore they can cause an alarm to be generated if the attacker is not conservative. We assume the worst possible case of an omniscient adversary that knows how much consumption will happen at the next time-step (this is a conservative approach to evaluate the security of our system, in practice we expect the anomaly detection system to perform better because no attacker can predict the future).

We now evaluate all possible combinations of the popular physical models and detection statistics illustrated in Table 2.1. In particular, we want to test AR models vs. LDS models estimated via system identification (SLS models do not make sense here because our system is dynamic) and stateless detection tests vs. stateful detection tests.
We launch an undetected actuator attack after 50 seconds using stateless and stateful detection tests for both: AR and LDS physical models. Our experiments show that LDS models outperform AR models, and that stateful models (again) outperform stateless models, as illustrated in Figure 4.14.

Figure 4.14. These figures show two things: (1) the stateful (CUSUM) test performs better than stateless tests when using AR (left) or LDS (right) models, and (2) LDS models perform an order of magnitude better than AR models (right vs left). Only for really small values of $\tau < \delta$ (0.04 minutes on average between false alarms), will the stateless test performs better than the stateful test.

Having settled for LDS physical models with CUSUM as the optimal combination of physical models with detection tests, we now evaluate the performance of different control algorithms, a property that has rarely been explored in our survey of related work. In particular, we show how Integrative control is able to correct undetected actuation attacks.

In particular we compare one of the most popular control algorithms: P control, and then we compare it to PI control. If the system operator has a P control of the form $u_k = Ky_k$, the attacker can affect the system significantly, as illustrated in Figure 4.15. However, if the system operator uses a PI control, the effects of the attacker are limited: The actuator attack will tend to deviate the frequency signal, but this deviation will cause the controller to generate a cumulative compensation (due to the integral term) and because the LDS model knows the effect of this cumulative compensation, it is going to expect the corresponding change in the sensor measurement. As a consequence, to maintain the distance between the estimated and the real frequency below the threshold, the attack would have to decrease its action. At the end, the only way to maintain the undetected attack is when the attack is non-existent $u^a_k = 0$, as shown in Figure 4.16.
Figure 4.15. Left: The real (and trusted) frequency signal is increased to a level higher than the one expected (red) by our model of physical system given the control commands. Right: If the defender uses a P control algorithm, the attacker is able to maintain a large deviation of the frequency from its desired 60Hz set point.

Figure 4.16. Same setup as in Figure 4.15, but this time the defender uses a PI control algorithm: this results in the controller being able to drive the system back to the desired 60Hz operation point.
In all our previous examples with attacked sensors (except for the pH case), the worst possible deviation was achieved at the end of the attack, but for actuation attacks (and PI control), we can see that the controller is compensating the attack in order to correct the observed frequency deviation, and thus the final deviation will be zero—technically speaking the asymptotic deviation is zero, while the transient impact of the attacker can be high. Figure 4.17 illustrates the difference between measuring the maximum final deviation of the state of the system achieved by the attacker, and the maximum temporary deviation of the state of the system achieved by the attacker.

As we can see, the control algorithm plays a fundamental role in how effective an actuation attack can be. An attacker that can manipulate the actuators at will can cause a larger frequency error but for a short time when we use PI control; however, if we use P control, the attacker can launch more powerful attacks causing long-term effects. On the other hand, attacks on sensors have the same long-term negative effects independent of the type of control we use (P or PI). Depending on the type of system, short-term effects may be more harmful than long-term errors. In our power plant example, a sudden frequency deviation larger than 0.5 Hz can cause irreparable damage on the generators and equipment in transmission lines (and will trigger protection mechanisms disconnecting parts of the grid). Small long-term deviations may cause cascading effects that can propagate and damage the whole grid.

While it seems that the best option to protect against actuator attacks is to deploy PI controls in all generators, several PI controllers operating in parallel in the grid can lead to other stability problems. Therefore often only the central Automatic Generation Control (AGC) implements a PI controller although distributed PI control schemes have been proposed recently [11].
Recall that we assumed the actuation attack was launched by an omniscient attacker that knows even the specific load the system is going to be subjected (i.e., it knows exactly how much will consumers demand electricity at every time-step, something not even the controller knows). For many practical applications, it will be impossible for the attacker to predict exactly the consequence of its actuation attack due to model uncertainties (consumer behavior) and random perturbations. As such, the attacker has a non-negligible risk of being detected when launching actuation attacks when compared to the 100% certainty the attacker has of not being detected when launching sensor attacks. In practice, we expect that an attacker that would like to remain undetected using actuation attacks will behave conservatively to accommodate for the uncertainties of the model, and thus we expect that the maximum transient deviation from actuation attacks will be lower.

4.7 Conclusions

We introduced theoretical and practical contributions to the growing literature of physics-based attack detection in control systems. Our literature review from different domains of expertise unifies disparate terminology, and notation. We hope our efforts can help other researchers refine and improve a common language to talk about physics-based attack detection across computer security, control theory, and power system venues.

In particular, in our survey we identified a lack of unified metrics and adversary models. We explained in this chapter the limitations of previous metrics and adversary models, and proposed a novel stealthy and adaptive adversary model, together with its derived intrusion detection metric, that can be used to study the effectiveness of physics-based attack-detection algorithms in a systematic way.

According to our survey, we are also the only research work conducting validation of our approaches in multiple setups, including: a room-size water treatment testbed, a real large-scale operational system managing more than 100 PLCs, and simulations of primary frequency control in the power grid. We showed in Table 4.1 how each of these validation setups has advantages and disadvantages when evaluating the x-axis and y-axis of our proposed metric.

One result we obtained across our testbed, real operational systems, and simulations, is the fact that stateful tests perform better than stateless tests. This is in stark contrast to the popularity of stateless detection statistics as summarized in Table 2.1. We hope our work motivates more implementations of stateful instead of stateless tests in future work.

We also show that for a stealthy actuator attack, PI controls play an important role in limiting the impact of this attack. In particular we show that the Integrative part of the
controller corrects the system deviation and forces the attacker to have an effective negligible impact asymptotically.

Finally, we also provided the following novel observations: (1) finding spatio-temporal correlations of Modbus signals has not been proposed before, and we showed that these models are better than models of single signals previously proposed in the literature, (2) while input/output models like LDS are popular in control theory, they are not frequently used in works published in security conferences, and we should start using them because they perform better than the alternatives, unless we deal with a highly-nonlinear model, in which case the only way to limit the impact of stealthy attacks is to estimate nonlinear physical models of the system, and (3) we show why launching undetected attacks in actuators is more difficult than in sensors.
CHAPTER 5

IMPROVING VISIBILITY OF NETWORK MONITORING AGAINST
LOW-LEVEL ATTACKS

5.1 Introduction

Industrial Control Systems (ICS) increasingly rely on embedded computers and networks that
monitor and control components which interact with the physical world. While some of these
infrastructures have used sensing and control for almost a century—the steam governor was
introduced in 1788, and the first programmable controller (the Modicon 0844) was released
in 1969—the last decade has introduced significant technological advances. In particular,
wireless sensors (wirelessHART, ISA-100), embedded computers, and communication networks
(combined with drastic reductions in deployment costs) have enabled the ubiquitous use of
networking and embedded devices in diverse ICS sectors. Moreover, their use is expected to
continue growing significantly in the coming years.

One of the recent research trends for ICS security 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 [53, 88, 146, 29, 128]. For example, if we have a sensor that monitors
the height of 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 works working on this topic do not explicitly describe the trust assumptions for all
parts of a control loop—controllers, actuators, and sensors.

The remainder of this chapter is organized as follows: In § 5.2, we briefly summarize the
scope of our study. We then in § 5.3, based on our Survey in § 2.4, present limitations of
previous work. Based on that concept, we design and implement a deep ICS monitor in § 5.5.
We present the results of our experimental validation of our deep monitor prototype in § 5.6.

---

1 David I. Urbina, Jairo Giraldo, Alvaro A. Cárdenas, Nils Ole Tippenhauer. Improving Visibility of ICS
Network Monitoring Against Low-Level Attacks.

80
5.2 Scope of our Study

In this chapter, 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 a field-layer physics-based anomaly detection system for the industrial control protocol EtherNet/IP. As far as we are aware, we are the first to justify and implement a monitoring device in the field communications layer of industrial control systems.

Developing this monitor is more challenging than deploying a security monitor in the supervisory network. Nevertheless, it allows us to solve limitations of implicit trust assumptions made in previous works, giving the anomaly detection algorithm the ability to detect attacks that could have been missed if we had deployed our monitor in a different place. The parser for EtherNet/IP, the software modules dealing with inserting a device between Programmable Logic Controllers (PLCs) and field devices, and the source code for the monitor and statistical tests will be released as open source to increase the number of tools researchers have available in order to analyze the security of industrial control systems.

5.3 Limitations of Previous Work

Most related work that we reviewed focused on describing attacks to specific devices (i.e., explicitly enumerating devices that are not trusted). Unfortunately, those works do not provide an fine-grained trust model that can be used to describe what can be detected and what cannot be detected when the adversary is in control of different devices.

Lack of Trust Models. Most works do not describe their trust models with enough precision. Information exchanged between field devices (sensor to controller and controller to actuator in Figure 2.2) is communicated through a different channel from information that is exchanged between controllers or between controller and the supervisory control center. Works that monitor network packets in the supervisory control network implicitly assume that the controller (PLC) they monitor is trusted, otherwise the PLC could fabricate false packets that the monitor expects to see, while at the same time sending malicious data to actuators (what Stuxnet did). Thus, we need to monitor the communication between field devices in order to identify compromised PLCs in addition to monitoring supervisory control channels to identify compromised sensors or actuators.
Trust in Actuators. Similarly, most works implicitly assume that actuators are trusted. One of the few works that attempts to give a definition of trust assumptions is $C^2$ [95]; however it mentions that “the approach can prevent any unsafe device behavior caused by a false data injection attack, but it cannot detect forged sensor data”, and later in the work we find “$C^2$ mitigates all control channel attacks against devices, and only requires trust in process engineers and physical sensors.” These two statements are in contradiction, and the correct statement to satisfy the security of their model is the latter. In addition, a missing trust assumption in their system (and in most of the literature we review) is the implicit assumption of trusted actuators: just because a PLC sent a control command to the physical system, we should not expect with 100% certainty that this command will be received and executed as intended.

Attack Types. In the following, we summarize the different attack types that we identified. Figure 2.4 shows an attack on the actuator that modifies the control command send to the plant. Note that the controller is not aware of the communication interruption. On the other hand, Figure 2.3 shows an attack in the sensor, which allow the attacker to deceive the controller about the real state of the plant. In the worst case, the control device can be compromised as well, giving the attacker potentially unlimited control on the plant to implement any outcome (see Figure 2.5) This last figure also captures the threat model from a malicious control command sent from the control center as seen in Figure 2.6: While the implementation might be different–one monitor is placed in the supervisory control network and the other monitor on the field communications network–the logical architecture–what the monitoring application sees–will be the same. In these attack schemes we assume that the control has a trusted detection mechanism, which can recognize unexpected behaviors and potentially take counter measures.

5.4 Deep ICS network monitoring

In the related work review (see § 2.5), we found that often trust models are not explicitly considering attacks on communication between PLCs and field devices. As a result, many of the proposed schemes cannot detect such attacks.

The general problem can be re-phrased as follows: “At which location should we deploy an intrusion detection system that monitors $u$ and $y$, and uses any anomaly in sensor or control signals to detect attacks”.

Understanding the general architecture between actuators, sensors, controllers, and control centers is of fundamental importance to analyze the implementation of a monitoring device and most importantly, the trust assumptions about each of these devices, as any of these devices (actuators, sensors, PLCs, or even the control center) can be compromised.

5.4.1 Limitations of Security Monitors on the Supervisory Control Network

If an anomaly detection system is only deployed in the supervisory control network (which typically has network switches with mirror ports), 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 (see Figure 5.1). 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).

![Figure 5.1. A compromised PLC can send false control commands to devices in the field while reporting a fake status of the system to the supervisory layer.](image)

If the network monitor is deployed at the supervisory control layer, it is 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 (i.e., the dotted $y$ line in Figure 2.1). If the control center had monitored directly the frequency converters through an independent communication channel it could have detected the attack.

A similar attack was performed against the Siemens 417 controller [82], where attackers captured 21 seconds 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 [82].

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.

5.4.2 Making Trust Assumptions Explicit

A number of works we reviewed did not mention where the monitoring devices will be placed, which makes it difficult to classify their respective trust model. For example, analyzing the security of sensor and actuation commands observed in the Distributed Network Protocol 3 (DNP3) communications standard [87, 86] implicitly assumes that the monitoring device is placed between the PLC (or RTU) and the control center, where DNP3 is most commonly used.

Therefore, such works implicitly assume that the PLC is reporting truthfully the measurements it receives, and the control commands it sends to actuators. This weak attacker model limits the applicability of the analysis, as only an attacker attacking a sensor or actuator directly can be detected. To mitigate such restrictions, we argue that anomaly detection monitors should (also) be used in field networks to detect compromised PLCs, actuators, and sensors. Assuming the monitor is placed in the field bus and can detect network-based manipulation of traffic, the selection of trusted components determines the kind of attacks that can be detected.

In general, we recommend that all works must discuss trust assumptions and the limitations of the anomaly detection system. For example if we are only monitoring one control loop with one sensor, one actuator and one controller, the following limitations need to be considered (summarized in Table 5.1):

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.
Table 5.1. Detectability of Attack Depending on Trust in Components.

<table>
<thead>
<tr>
<th>Component Trust</th>
<th>Detection</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLC Sensor Actuator</td>
<td>possible</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- = trusted/detection possible, - = untrusted/detection not possible,
\(\sim\) = cannot detect zero-dynamics attacks

3. If we trust the sensors but not the controller or the actuators, then for most practical cases we can detect an attack, because the goal of the attack is to affect the physical system, and we assume we can monitor these changes through \(y_k\). We show that zero-dynamic attacks [139] is an example of an instance where even when we trust sensor measurements, we cannot detect attacks caused by a compromised actuator.

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.

5. If we trust the controller and the sensors, then we get a similar case to the last point: possible implausible combinations of control actions and sensor measurements. With the possible exception of zero-dynamic attacks.

6. If we trust the actuator and the sensors, we can detect a compromised controller by identifying if a control action is the correct response to the current state of the system.

5.4.3 Observations

Our analysis in § 5.4.2 shows that as long as you trust two components in the loop, it is possible to detect an attack on the remaining component. If we trust the sensors but do
not trust either the actuators or the PLCs, we can still detect attacks, unless they are zero-dynamic attacks [138, 139, 114] (although not all physical systems are vulnerable to these attacks). Finally, if we only trust the actuator (or only the PLC), the attacks could be completely undetected. We note that while there are still some attacks that cannot be detected, we can still detect more attacks than at the SCN.

5.5 Developing A Field-Layer Security Monitor for ICS

5.5.1 Testbed Description

The SWaT testbed we use for our experiments is a water treatment plant consisting of 6 main stages to purify raw water. The testbed has a total of 12 PLCs (6 main PLCs and 6 in backup configuration to take over if the main PLC fails). The general description of each stage is as follows: 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. 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 (see Figure 5.2). Finally, in RO final product stage stores the RO product (clean water).

Each stage is composed by two PLCs (one primary and one redundant in hot-standby mode); the primary and redundant PLCs for the raw water stage can be seen in Figure 5.3. 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. The primary PLC receives the sensor information (water level and water flow for stage 1) and replies back with the corresponding control actions.
5.5.2 Differences Between SCN and FCN Layers in the Testbed

The network of the testbed (illustrated in Figure 5.4) uses the Common Industrial Protocol (CIP) stack [27] for device communications at the SCN and FCN layers. There are however a variety of differences between these two layers, as illustrated in Figure 5.5.

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
Figure 5.4. Communication between actuators or sensors to PLCs is achieved by field communication protocols. Control between PLCs or between PLC and a central control server is achieved with supervisory industrial protocols. This network is part of a testbed we use for our experiments.

Figure 5.5. Differences between the SCN and the FCN network stacks. At the SCN layer, PLCs communicate between each other and with the HMI using well-known standard formats, while at the FCN layer, UDP connections are used for multicast connections to update multiple devices through I/O messages.
specific data item request/response exchange occurs with a low frequency or under special circumstances, the data exchanged will be missed.

At the SCN layer, devices communicate through point-to-point connections over the Transmission Control Protocol (TCP) and exchange explicit CIP messages—see Figure 5.6; these explicit messages are standard and openly accessible formats defining a clear semantic of the messages exchanged between devices. As shown in Figure 5.7.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.

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

Figure 5.6. Explicit CIP Message encapsulation over EtherNet/IP.

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 Figure 5.7.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.

### 5.5.3 Parsing FCN Data

According to the electrical drawings, we found that each PLC had three modules: a digital input module (to receive on/off status reports from sensors 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 opening of a valve).

The number of signals available per module are summarized in Table 5.2. 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, Figure 5.8 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).
Table 5.2. Modules for each PLC.

<table>
<thead>
<tr>
<th>I/O Message</th>
<th>Signal size (bits)</th>
<th># signals</th>
<th>Avg. Freq. (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Digital Input</td>
<td>1</td>
<td>32</td>
<td>50</td>
</tr>
<tr>
<td>Digital Output</td>
<td>1</td>
<td>16</td>
<td>60</td>
</tr>
<tr>
<td>Analog Input</td>
<td>16</td>
<td>12</td>
<td>80</td>
</tr>
</tbody>
</table>

Figure 5.8. Digital Input Module with 32 input signals (1-bit 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 milliseconds. 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 Figure 5.9.

![Figure 5.9. Analog Input Module with 12 input signals (16-bits signals) for the Raw Water Storage stage PLC.](image)

In order to scale back and forth between the 4-20 mA analog signal and the real measurement with standard units, we use Equation (5.1). 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. Figure 5.10 shows an example for the scaling of the water flow.

\[
Out = (In - RawMin) \times \frac{EUMax - EUMin}{RawMax - RawMin} + EUMin
\]

(5.1)

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 also can 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\(^2\) code and its only external dependencies are Scapy and NetFilterQueue.

Specifically, making use of Scapy\(^3\), 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 in use.

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, for later

\(^2\)Python Language. Version 2.7. [https://docs.python.org/2/](https://docs.python.org/2/)

releasing them back 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\textsuperscript{4} 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, as such it looks like Figure 2.2.

5.6 Experiments

We now illustrate how our tool can be used to launch and detect attacks at the FCN in the testbed.

On 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 a 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 water level. 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

$$\text{Area} \frac{dh}{dt} = Q^{in} - Q^{out},$$

where $\text{Area}$ 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

$$\hat{h}_{k+1} = h_k + \frac{Q^{in}_k - Q^{out}_k}{\text{Area}}$$

\textsuperscript{4}Python bindings for libnetfilter_queue. https://github.com/fqroutert/python-netfilterqueue
where \( h_k \) represents the received sensor measurement for the water level at time \( k \), \( Q_k^\text{in} \) represents the on/off variable of the state of the inlet valve at time \( k \), and \( Q_k^\text{out} \) represents the on/off variable of the state of the pump that takes water off the tank. Given these three 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 \( \delta \)) is the computed as part of the CUSUM anomaly detection test: \( S_{k+1} = \max(0, S_k + r_k - \delta) \). If this statistic is greater than a user-specified threshold \( \tau \) (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.

### 5.6.1 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 Figure 2.3. 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 5.11 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 1m). 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 attack is launched.

In this case 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 false sensor data.

### 5.6.2 Actuator Attack (Inlet Water Valve)

Now we turn our attention to attacks against the actuators, as illustrated in Figure 2.4). 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 0.8m 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 5.12 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 (0.8m) 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 an anomaly that is detected.

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.
5.6.3 PLC Attack

In this attack, the logic of the PLC is modified so that it sends a false control command (see Figure 2.5), but reports that everything is fine to the SCN as in Figure 5.1. In particular, the PLC sees that the water is at the high level of 0.8m (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 0.8m.

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. In addition, unless it receives sensor data directly from the field (and in our testbed the supervisory control network does not handle any field data directly; all of this is mediated through PLCs), then it doesn’t have access to sensor data, and it has to trust the reported sensor observations sent by the PLC to the control center.

Complicating things, in this case our physics-based anomaly detection algorithm in the FCN cannot detect this attack either because there is no discrepancy between the observed control commands, and their effect on sensor measurements. Figure 5.13 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 compromise controllers, we can deploy safety checks in our field monitoring device in order to guarantee that no unsafe controls will be taken. In this scenario, our field monitoring device can act essentially as a reference monitor, mediating between a potentially compromised PLC, and the physical process.

\[ \text{PumpOff} \land \text{Waterlevel} < 0.5 \]
\[ \text{ValveClose} \land \text{Waterlevel} > 0.8 \]

\[ \text{ValveOpen} \land \text{Waterlevel} > 0.8 \]
\[ \text{PumpOn} \land \text{Waterlevel} < 0.5 \]

Figure 5.14. Safety check.
To showcase our safety check-reference monitor architecture, we build a state machine with the safety rules for the process controlled by the PLC. There are two safety rules that can be violated by incorrect control commands: (1) if the height of the water is above 0.8m then the intake valve cannot be open, otherwise we risk a tank overflow, and (2) if the water level is below 0.5m, then the pump extracting water from the tank cannot be turned on, otherwise we risk damage to the pump. The corresponding state machine implemented in our field monitoring device can be seen in Figure 5.14. If the PLC command does not correspond to the safety rules dictated by the state machine, the attack can be detected.

![State Machine Diagram](image)

Figure 5.15. Redundant control.

Another approach that we can deploy in our field monitoring device is to use a redundant control logic to detect if the control signal sent by the PLC matches the redundant control signal from our monitoring device. That is, the detection algorithm can look at the control signal sent by the PLC to the field (as in Figure 2.5) and then compute a redundant control \(K(y_k)\) and then check if it gives the same result as the one observed.

The redundant control architecture in this case is illustrated in Figure 5.16 and the logic is illustrated in Figure 5.15. The redundant control in this case is not enough to detect if the pump is turned on at an incorrect state, as this control logic operates in another PLC (i.e., as will be explained later in this chapter, the decision to turn on or off the pump does not depend on PLC 1, it depends on another PLC, as illustrated in Figure 5.17), therefore safety checks appear to be a better test.

5.6.4 SWaT: Visibility by Correlating Data from Multiple Control Loops

The worst type of attack corresponds to the case when the sensors and actuators are attacked simultaneously (Figures 2.3 and 2.4). In this scenario, the adversary is able to send arbitrary actuator control signals (e.g., open the intake valve when the water in the tank is 0.8m) 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 0.8m).

However, if a different control loop in another part of the plant is safe from intrusions, this type of attack might be detected. In particular, it is possible to exploit the correlation between different stages of the testbed in order to increase the visibility of the system against resourceful adversaries.

In particular, our testbed has two stages controlled by two PLCs, each of them receiving different field signals from the physical process, as illustrated in Figure 5.17.

We focus our attention on the water level of both tanks, 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.

In particular, in this case we assume that our attacker has compromised both the pump actuator and the water level sensor in the first stage of the testbed. As a result, if the attacker wants to damage the pump, it can turn on the pump directly from the actuator command, but our security monitor will not see that command, and will believe that 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 0.8m, while in reality the water level in tank one is decreasing because the pump is on.

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 Figure 5.18 (left)) 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 Figure 5.18 (right).

This approach can be extended to large scale multi-stage processes. To illustrate this principle, we now turn our attention to a larger process with multiple stages: the Tennessee-Eastman process.

5.6.5 Tennessee Eastman Process: Visibility by Correlating Data from Multiple Control Loops

We also have access to a Hardware-in-the-Loop (HIL) testbed that implements the well-known Tennessee Eastman Process (TEP) in a real-time simulator. The real-time simulator then interacts with Siemens S7 controllers.

The TEP was first proposed by Down and Vogel [41] and has been extensively used for the evaluation of novel control techniques, due to its complexity and large number of sensors. One key difference from the previously describe testbed, TEP is typically controlled by several distributed PID control loops [122] (i.e., the PLCs have analog outputs in this case).
Figure 5.18. Sensor and actuator attack in stage 1. The attack cannot be detected by the detection algorithm in stage 1, but it can be observed in the other stage.
Due to the correlation of the different stages of the process, including distributed FCN detection mechanisms can increase the visibility of the system, and facilitate the detection of anomalies or attacks that may remain stealthy for one anomaly detector located in one control loop, but where the attack will affect other loops. Under these conditions, adversaries would need to attack all distributed loops simultaneously to remain stealthy, or decrease the impact of such attacks in a way that it does not trigger alarms in other parts of the plant.

TEP sensor measurements include a large amount of redundancy. For instance, there are 11 control loops, but 42 measurements. We can define two types of measurements: i) controlled sensors i.e., measurements used to generate control signals, and ii) non-controlled measurements—that is, measurements that provide information but are not used by the controllers. The correlation among controlled and non-controlled sensors can be exploited to increase the difficulty on deploying stealthy attacks.

Figure 5.19. Tennessee Eastman Process with some Multi-Variable Detection (MVD). Similar colors indicate a high correlation coefficient (i.e., > 0.95). Dashed circles indicate low correlation coefficient (i.e., < 0.01). Each CD receives sensor information and the actuator signal corresponding to at least one of the sensors.

Figure 5.19 depicts the general scheme of the TEP. Using the correlation coefficient, we are able to identify the sensors with the highest correlation. Since the adversary’s objective is to maximize the deviation of the states of the system towards unsafe levels, her target will be the sensors that are used by the controllers, i.e., the measurements that directly affect the
systems behavior. As a consequence, we exploit the redundancy by adding anomaly detection mechanisms for the non-controlled measurements.

As an example, colored circles in Figure 5.19 indicate correlation and black dashed circles represent sensors without correlation. Multi-Variable Detection (MVD) can be located in such a way it receives information from different PLCs. The information is used to obtain detection statistics for the measurements that are not used, and from other control loops as illustrated in Figure 5.20. If an attacker wants to remain stealthy, she will have to attack several sensors simultaneously from different PLCs.

![Multi-Variable Detection (MVD) architecture.](image)

Figure 5.20. Multi-Variable Detection (MVD) architecture. The control input $u_k^1$ is used to generate predictions for different correlated sensors; $u_k^N$ and $y_k^N$ belong to a correlated control loop. Stealthy attacks on $y_k^1$ or $u_k^1$ can be detected.

Since the TEP is highly nonlinear, obtaining a Linear Dynamic Model estimation is not feasible (as we did with the water height in our first testbed). Using Matlab’s System Identification toolbox, we are able to obtain individual Hammerstein-Weiner models for several control loops that estimate the input/output relationship. These models combine linear and nonlinear blocks that approximate the sensor measurement behavior for a given control input. Therefore, we are able to generate detection statistics, such as the residuals, by comparing the estimated model with the real system behavior.

As an example, let us consider the product separator temperature (typically called $x_{meas 11}$), which is controlled by a PID control in PLC 1 that modifies the Condenser Cooling water flow (called $x_{mv 11}$). Similarly, the stripper temperature ($x_{meas 18}$) is managed by PLC 2; even though $x_{meas 18}$ is not a controlled variable, PLC 2 receives it and transmits it through the supervisory control network for redundancy purposes. The correlation coefficient

---

of both sensors is 0.968, which indicates a high level of correlation, i.e., changes in one sensor can be observed in the other. Using the Matlab’s toolbox, we obtained estimation models for each sensor with input xmv 11 (condenser cooling water flow). First, let us consider the case a simple attack at \( k = 30 \) of the form \( y_k^a = y_k + 0.5(k - 30) \), which suddenly increases \( y_k \) at each time instant. Figure 5.21 illustrates how the anomaly detection statistic in xmeas 11 rapidly detects this attack. At the same time, the detection metric for xmeas 18 is also affected.

![Figure 5.21](image)

Figure 5.21. Attack on product separator temperature sensor (xmeas 11) after 30 hours. The attack is detected by the anomaly detector of xmeas 11 and a correlated sensor xmeas 18.

We propose a more sophisticated attack whose aim is to remain completely undetected while maximizing its impact. Since the attacker wants the attacked signal \( y^a \) to not generate any alert, then \( r_k = |y_k^a - \hat{y}_k| \leq \tau \), and the attack is as follows: \( y_k^a = \hat{y}_k + \tau \). The attack is launched at \( k = 40 \) and it ensures that \( r_k = \tau \); therefore it is never detected. Figure 5.22 illustrates the impact of the undetected attack for the product separator temperature (xmeas 11); however, the attack is rapidly detected by the detection mechanism of the stripper temperature.

Now, let us assume that the adversary is able to attack simultaneously the product separator temperature sensor and the control input that the stripper temperature estimator
receives, such that the attack cannot be observed for neither detector. We can exploit the correlation of xmeas11 with controlled loops. In this case, the stripper level, which corresponds to a different control loop, possess a level of correlation of 0.7, and is able to observe the presence of the attack, as illustrated in Figure 5.23. The more controlled and non-controlled sensors are included in the MVD, the more difficult is for an attacker to remain stealthy.

5.7 Discussion

In this chapter, we have discussed in detail the challenges and opportunities for detecting attacks using the detection block in Figures 2.2 and 2.5.

In particular, we have expanded this generic detection block with a detailed list of the ways that attacks can be detected given the information field monitoring devices capture. In particular, our proposed detection architecture is illustrated in Figure 2.7. There are two blocks that are straightforward to implement: (1) The controller block in Figure 2.7 is a redundant control algorithm (i.e., in addition to the controller of Figure 2.2) that checks if the controller is sending the appropriate $u_k$ to the field, and (2) The safety check block
Figure 5.23. For the optimal stealthy attack on Product Separator Temperature sensor (xmeas11) and an attack on the transmitted control signal information xmv11, the detection with the correlated non-controlled sensors is unable to detect the attack; however, analyzing other control loops (in this case the stripper level with a correlation coefficient of 0.7) allows the detection of an attack.

is an algorithm that checks if the predicted future state of the system will violate a safety specification (e.g., the pressure in a tank will exceed its safety limit). Finally the prediction, residual generation and anomaly detection can be used to detect attacks on the sensors or actuators.

When we need to consider correlations between multiple loops (because both sensors and actuators have been compromised in a control loop) we then need the architecture presented in Figure 5.20, where multiple field device monitors exchange anomaly detection scores in the hopes of detecting these powerful attacks.

5.8 Conclusions

In this chapter, 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. As far as we are aware, we are the first to implement a monitoring device in this lower layer of industrial control systems. 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.

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 started the discussion of integrating the information from multiple field monitors at different stages in a large process. Our work in this distributed architecture is preliminary but shows promise in the quest to improve the visibility of our system, and make the work of the attacker harder if it wants to remain undetected.
CHAPTER 6
ANALYSIS OF ENCRYPTED TRAFFIC

6.1 Introduction

The use of Intrusion Detection Systems (IDSes) in Smart Grid deployments is becoming a fundamental approach in securing smart grid networks, especially as the scale and scope of such networks increases, encompassing ever more critical functionality, and as they complete their migration to standardized communication stacks. Similarly, increased use of encryption is expected to both prevent eavesdropping and better protect sensitive and private data, such as fine-grained meter readings. A side-effect of encrypting communication is the loss of visibility into network traffic, which prevents IDSes from performing packet-level inspection analysis.

While an IDS can be given encryption keys to decrypt and parse messages, in practice there are several reasons for having a hierarchical analysis where one IDS monitors encrypted traffic and another one has the ability to decrypt special messages. Having an IDS capable of analyzing encrypted communications is particularly important to prevent information leakages, as large companies keep increasing the number of analysts looking at intrusion alarms, and smaller companies outsource the analysis of logs to outside companies.

A variety of techniques have been discussed to enable IDSes to monitor encrypted traffic in AMI networks [23], including sharing keys with IDS sensors, leveraging partial encryption, or applying traffic analysis techniques. In this work, we investigate how some of those approaches could work in practice by applying them experimentally on real traffic captured at a large operational utility AMI network.

In particular, we study four different approaches:

- Monitoring the periodicity of meter communications,
- Detecting rogue devices through passive fingerprinting,

---

• Tracking unknown flows by baselining the network connectivity graph, and

• Identifying traffic patterns and outliers through unsupervised clustering.

We discuss the effectiveness of each of these approaches in detecting suspicious activity in the context of the AMI traces that we were able to collect. The main contributions of this work are to offer a detailed view of the internal network communications found on a large AMI and to provide insights on the challenges and possible solutions to identify intrusions despite the deployment of encryption.

6.2 Related Work

Analysis of encrypted traffic has been an active research area for the past two decades, and can be classified into two broad categories: 1) traffic classification techniques that use machine learning to classify flows per application [106, 66], and 2) packet analysis techniques that attempt to identify protocol operations through packet-level inspection [70].

Our work falls in the second category, since AMI communication traffic uses a single application-level protocol, namely ANSI C12.22. The work closest to ours is [56], where the authors proposed an IDS approach that does not need to inspect packet payloads. They leveraged the periodicity of process control system communication to identify outliers that could potentially represent malicious behavior. They use packet sizes, packet directions, and the relative time position of packets within a time series to label similar packets. The main limitations of this approach is the challenge of correctly tuning or training the parameters related to the polling cycles in order to achieve a good detection accuracy. A difference with our approach is our focus on AMI traffic and the need to identify different C12.22 sessions within a single TCP session.

Ali and Al-Shaer [7] demonstrate that AMI behavior can be modeled using ‘event logs’ collected at access points. Specification invariants, generated from AMI device configurations, are used to verify the AMI behavior models near the access points where the logs are collected. Our approach looks at the encrypted C12.22 traffic to detect anomalous behavior, which means there is more flexibility for placing IDS sensors in different physical locations, and enabling us to detect attacks that are located within the Neighborhood Area Network (i.e., not reaching the access points).
6.3 Data Collection

We partnered with a large U.S. electric utility to collect network traffic at the head-end of a 30,000-meter operational AMI network, over several time periods. An AMI deployment usually comprises a Wide Area Network (WAN) that connects collection engines at the head-end with access points in the field, and Neighborhood Area Networks (NANs), each of which is connected to one or more access points. A NAN is intended to carry smart meter communications. In our case, the WAN uses TCP/IP and consists of about 90 distinct access points. Each NAN uses a wireless mesh network that is not IP based but uses proprietary protocols. All datasets were recorded in PCAP format on the TCP/IP portion of the network.

The AMI uses ANSI C12.22 [132] (see § A.2) as the application-layer communication protocol. Only packets on the TCP port used by C12.22 (port 1153) were recorded. The PCAP capture files were dissected using both Wireshark and an IDS sensor called Amilyzer [25]. For each trace, TCP/IP and C12.22 [132] protocol information were extracted. The C12.22 payloads consist of an Association Control Service Element (ACSE) header and one or more Extended Protocol Specifications for Electric Metering (EPSEM) elements. ASCE headers contain control information about the association between communicating entities, such as caller and called identifiers (calling/called AP-titles). EPSEM elements have either a service request or response and hold C12.19 data values. In most of the packets recorded at the utility facility, the EPSEM elements were encrypted but the TCP/IP headers and the ACSE header were sent in the clear. Table 6.1 summarizes the main features of the two traces collected at the utility site.

<table>
<thead>
<tr>
<th>Date recorded</th>
<th>Duration</th>
<th>#Packets</th>
<th>#IPs</th>
<th>#ApTitles</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 23, 2014</td>
<td>24 hours</td>
<td>1,009,982</td>
<td>86</td>
<td>25,645</td>
</tr>
<tr>
<td>May 5, 2014</td>
<td>24 hours</td>
<td>1,011,988</td>
<td>87</td>
<td>28,471</td>
</tr>
</tbody>
</table>

6.4 Experimental Data Analysis

6.4.1 Taxonomy of AMI Traffic

Our goal in this section is to understand what types of network traffic are present in real-world AMI network traces. Figures 6.1 and 6.2 depict packets flowing between the meters and the
Figure 6.1. Number of packets per minute collected on April 23 over a 24-hour period. Overall traffic is shown in blue and TCP retransmissions are shown in red.

Figure 6.2. Number of packets per minute collected on May 5 over 24 hours on an AMI of 30,000 meters. Overall traffic is shown in blue and TCP retransmissions are shown in red.
collection engine in our two traces. The sampling resolution is one minute, i.e., the Y-axis counts the number of packets per minute (1,440 minutes per day).

The red (lower) curve in both figures represents TCP retransmissions (1.6% of the total number of packets). Those retransmissions indicate possible losses and delays on the communication links used by the WAN. They are useful information to be monitored by an IDS in order to detect irregularities. For instance, a sudden loss of quality of service could be due to a denial-of-service attack.

We then identified three categories of traffic in both traces: 1) a periodic and continuous background traffic of keep-alive requests, 2) a periodic set of AMI requests/responses, and 3) aperiodic traffic.

Packets in the first category are unencrypted and consist of identify requests sent by cell relays every 60 seconds. They have an empty called-Ap-title indicating that they are broadcast messages, but they receive an acknowledgment packet from the collection engine about 0.2 seconds after being sent.

Packets in the second category are encrypted and larger in size (254 bytes compared to 92 bytes long for the keep-alive packets). They have a period of 240 minutes and are initiated by the collection engine. The period of 240 minutes was identified by applying an auto-correlation technique, as shown on Figure 6.3. In the figure, the interval between spikes can be considered to be the period. Meters being contacted by those requests respond after a random delay over a period of 135 minutes that starts 10 minutes after the collection engine sent the requests. This traffic, which consists of periodic meter readings, shows strong regularity on the second trace (Figure 6.2) but only starts after 367 minutes in the first trace (Figure 6.1). After checking with the utility, this initial gap of 6 hours with no periodic traffic was due to a system outage. Knowledge of the expected periodicity of the traffic enables an IDS to report such irregularities.

The third category consists of three types of unencrypted requests for which we did not identify a period. They include registration requests, write requests, and response errors to some encrypted requests. They were initiated by 114 devices in the first trace, and 159 devices in the second trace, which represents less than 0.5% of the meter population. We compared the ApTitles and found that only 14 were common across the two traces, indicating that this traffic is specific in time and in scope.

We analyzed further the registration requests and responses and found that there was about 1 registration response every 10 minutes, and that most registrations happen within 20 minutes of each other. This indicates that the network as a whole is relatively stable, and that there are occasional disconnections, typically in small clusters where one poor quality or
disconnected link led to the disconnection of multiple meters and their change of affiliation to a neighboring access point’s network. We believe that tracking the patterns of registration responses in an AMI network, can be helpful in identifying anomalous behavior, including attacks against the AMI mesh routing protocol, which lead to an increase in registration traffic.

![Auto-correlation](image)

Figure 6.3. Auto-correlation applied to the second 24-hour trace to find the period of 240 minutes between encrypted requests initiated by the collection engine.

### 6.4.2 AMI Device Fingerprinting

While encryption hides payload content, it is still possible to extract useful information from the unencrypted packet headers. The idea in this section is to check if the network stack of AMI devices and OS/firmware running on them could be fingerprinted to be validated despite the encryption. This would enable us to add device fingerprint signatures to the IDS in order to flag unknown or unauthorized devices appearing on the network.

In order to explore this direction, we ran a passive fingerprinting tool called *P0f* (version 3.06b) [151] on both traces. Based on the contents of captured packets, P0f extracts fingerprints consisting of MTU signature and TCP signature [151], which vary across protocol stack implementations. We note that, in traces collected at the head-end, only device information about the collection engine and the access points is visible. Monitoring the characteristics of smart meter devices would require that packets be recorded within NANs or
that a fingerprinting mechanism based on C12.22 payloads, which are visible to the collection engine even when access points are mediating the communication, is implemented.

P0f identified three distinct fingerprints in both of our traces, besides ones attributed to the collection engine. Those three fingerprints were not found in P0f’s signature database and are associated with the IP addresses of access points. The exact fingerprints being sensitive information, we label them $FP_1$, $FP_2$, and $FP_3$. $FP_1$ indicates larger initial TTL and window size, as well as more TCP options set. The only difference between $FP_2$ and $FP_3$ is found in initial TTL. We observed that all access points share $FP_1$ when they are initiating TCP connections, while $FP_2$ and $FP_3$ divide access points into two groups when they accept incoming TCP connections. This result indicates that the pairs of fingerprints ($FP_1, FP_2$) and ($FP_1, FP_3$) can be used to identify two families of devices. We confirmed with the utility partner that those devices are different models. Another important result is the consistency of those pairs of fingerprints across the two traces, indicating that an attacker intruding on the network with a personal computer, impersonating meters [97], or access points running malicious/unauthorized firmware could be flagged.

6.4.3 Connectivity Graph

The concept of identifying intrusions using a network connectivity graph refers to learning the relationships among end points in order to flag when a new end point or a new link appears in the graph. Access to two 24-hour windows of traffic, 10 days apart from each other, can help us understand if such a technique could be promising.

The addresses of end points in an AMI can be extracted at the physical layer (MAC addresses), IP layer (IP addresses) and at the application layer (ApTitles). An IDS sensor can extract those addresses and keep a database updated to identify when a new end point initiates communication and when a new pair of end points is exchanging traffic.

Table 6.2 contains the unique number of IP addresses and ApTitles found in each dataset. Those results reveal important changes from one collection date to the next. In particular, we observe greater variations in the source and destination roles of end points compared to their unique count. This indicates that new nodes appeared but a larger number of nodes that were used solely as sources became destinations, and vice-versa.

Table 6.3 contains similar results but for relationships among end points, where each connection is identified by a source and destination IP address pair. We observe the consequence of role reversal with a large number of new relationships added and relationships removed between the initial and the most recent data set collected.
Table 6.2. Address information collected at layers 3 and 7 over a 24-hour period, and differences with a 24-hour trace collected 10 days later.

<table>
<thead>
<tr>
<th>End points</th>
<th>Initial set</th>
<th>Added</th>
<th>Removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP addresses</td>
<td>86</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Source IP</td>
<td>69</td>
<td>21</td>
<td>9</td>
</tr>
<tr>
<td>Dest. IP</td>
<td>57</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>ApTitles</td>
<td>25,645</td>
<td>9,096</td>
<td>6,270</td>
</tr>
<tr>
<td>Calling ApTitles</td>
<td>21,768</td>
<td>11,843</td>
<td>5,951</td>
</tr>
<tr>
<td>Called ApTitles</td>
<td>8,768</td>
<td>946</td>
<td>7,328</td>
</tr>
</tbody>
</table>

Table 6.3. Number of unique connections among end points collected at layers 3 and 7 over a 24-hour period, and differences with a 24-hour trace collected 10 days later.

<table>
<thead>
<tr>
<th>Unique connections</th>
<th>Initial set</th>
<th>Added</th>
<th>Removed</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Src IP, Dst IP)</td>
<td>124</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>ApTitles</td>
<td>27,777</td>
<td>15,152</td>
<td>13,449</td>
</tr>
</tbody>
</table>

The results from the connectivity graph analysis indicate that the AMI under consideration was too dynamic at the time of our data collection for this approach to be effective. We got confirmation from the utility partner that a large number of meters were added and reassigned to specific cell relays between the two days during which we collected data. Once the population of end points and the communication patterns are stable, the technique of monitoring the connectivity graph can be highly effective to detect rogue end points, spoofing attacks, and anomalous changes in communication patterns.

6.4.4 Unsupervised Learning

Traffic analysis of encrypted communications has traditionally leveraged two pieces of information that are not concealed by encryption: social behavior of a node (who is talking to whom), and network traffic statistics such as packet sizes, timings, and header information from the various traffic flows.

Our goal in this section is to apply unsupervised learning algorithms to encrypted AMI network traffic in order to understand whether or not traditional feature vectors used for encrypted traffic classification can reveal an underlying structure of AMI network flows. In particular, since we know that C12.22 packets contain either requests or responses from a limited number of commands, our initial goal is to see if unsupervised classification algorithms can identify clusters of encrypted network flows of packets with potentially similar commands.
being sent or received and also can detect an outlier, which may be an indication of attacks, and regular AMI communications, which would be represented as clusters.

One of the main differences between the challenges we encountered and prior work on encrypted traffic classification [107, 21, 22, 20] is that in that prior research, the authors extract feature vectors from TCP flows. Because TCP is a connection-based protocol, this allows researchers the ability to collect aggregate statistics of several packets (multiple packet sizes and timing information corresponding to one session) going back and forth between sender and receiver. While C12.22 supports connections, most of the communication (in fact, all of the flows in our trace) consist of only one or two packets: either a notification update such as a meter sending its table to the server at periodic intervals, or a request and a reply.

Because all communications in our C12.22 trace are either a single packet or a request-response pair, we can only create network flows of one or two packets. We extracted C12.22 flows by focusing on the following header parameters: called-AP-title, called-AP-invocation-id, calling-AP-title, and calling-AP-invocation-id.

Calling-AP-title and called-AP-title are the device identifiers of the source and destination of the packet, respectively. Calling-AP-invocation-id is a sequence number used to keep track of sessions and to eliminate duplicate packets. Called-AP-invocation-id is empty for the first packet of a session, and matches the previous calling-AP-invocation-id for response packets.

C12.22 headers also contain a Request Control Flag (RCF) that according to the standard specifies the following values: 0x0 to denote that the sender wants a reply to this packet, 0x1 to denote that the sender wants a reply to this packet only under error conditions, and 0x2 to instruct the receiver to not respond to this packet. We also observed a flag value of 0x3 that is manufacturer-specific. This flag helped us in identifying flows (when we see 0x2 we know we do not have to look for a matching calling-AP-invocation-id). We also confirmed that all replies (the second packets of our flows) have the flag RCF 0x2, thereby confirming that our matching flows identifies a maximum of only two packets per flow. In the traces analyzed, we also found an unknown flag (not defined by the standard).

Among all unsolicited messages (first packets in a flow), the distribution of the RCF flag is shown in Table 6.4.

Identifying single-packet flows (unsolicited-messages) and two-packet flows with request-reply pairs enabled us to start creating flow-based feature vectors. Those vectors consists of (1) the size (in bytes) of the payload in the first packet of the flow, (2) the size (in bytes) of the payload in the second packet of the flow (zero if there is no second packet), (3) the time (in ms) between packets (zero if there is no second packet), and (4) the direction of the
Table 6.4. Distribution of RCF flag among all unsolicited messages (first message of a session).

<table>
<thead>
<tr>
<th>Flag</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Always Respond, 0x0</td>
<td>44%</td>
</tr>
<tr>
<td>Respond on Exception, 0x1</td>
<td>18%</td>
</tr>
<tr>
<td>Never Respond, 0x2</td>
<td>19%</td>
</tr>
<tr>
<td>Manufacturer Flag, 0x3</td>
<td>19%</td>
</tr>
</tbody>
</table>

flow: 1 if the flow is originated from the collection engine, and 0 if the flow originated from a device in the field.

Figure 6.4 shows the multidimensional space of the feature vectors. A first observation is that there is a clear outlier: one of the response messages has a payload of 1,044 bytes, which makes it much larger than the rest. Looking at the packet that generated this outlier we found that it is a packet composed of three C12.22 layers. Similarly we found that the largest unsolicited packet (a request of 2,089 bytes) consisted of four C12.22 layers concatenated together. Upon further inspection, we found 82 packets in the first trace with multiple C12.22 payloads; furthermore those extra layers have the same source and destination ID (Called and Calling AP titles). They represent 0.0147% of all packets and are due to packet aggregation performed by the access points.

Figure 6.4. Feature vector space representing each feature against the payload size in bytes and the time difference in seconds.

Other observations from Figure 6.4 include the following: (1) unsolicited messages have a wide range of packet payload lengths, while replies are usually small and tend to be less than 200 bytes; (2) unsolicited messages that receive a reply have small payloads (less than 900
bytes) while unsolicited messages do not ask for replies can be up to 2000 bytes; (3) there is a wide range of packet sizes for unsolicited messages originating from the AMI, while packet sizes of unsolicited messages starting at the server side are small (up to 500 bytes). These traffic patterns can be used to develop anomaly detection rules such as: (1) a reply larger than 200 bytes is anomalous, (2) if an unsolicited message is larger than 800 bytes, then it will not receive a reply (if it does, then this is an anomaly), and (3) an unsolicited message from the server to the AMI larger than 500 bytes is an anomaly.

We then applied unsupervised learning algorithms on the feature vectors to identify patterns in the dataset. In order to make sure that one dimension is not dominating the others, we normalized each dimension using the following formula: \[ x' = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} \].

We used the K-means algorithm to identify clusters of packets. While there are several automatic suggestions recommending values for \( K \) there is no clear consensus on the selection of \( K \), and the best approaches rely on attempting multiple cluster numbers and evaluating them with exploratory data analysis. We experimented with small values of \( K \) from 2 clusters to 6 in order to identify a small number of clusters from which we could do a manual analysis of the clusters, and in particular of the packets belonging to each cluster. We focus on 6 clusters in this work, with cluster labels of 0 through 5.

From our clusters, we were able to identify that cluster 5 has 98% of the identify requests (from a total of 75,563), which are unencrypted and recognizable. Although, we could not extend our analysis to identify other specific commands sent and received due to a lack of corresponding plaintext packets, these results support our hypothesis that clusters of encrypted C12.22 communications can identify specific commands exchanged in the network. Our goal is to extend this work to identify all commands being exchanged in the network (e.g., with the help of a testbed) and then instruct an IDS to allow communications within traditional clusters (commands) and flag as suspicious communication flows that do not fall into any traditional commands. The other side of this interpretation is that encrypted communications can leak the type of commands being shared in a network, and this is something a designer should take into account when doing vulnerability assessments.

Clusters can also be used to identify faults or misconfigurations. Looking at clusters 2 and 4, we found several packets with missing elements such as calling invocation IDs. Overall, clusters 2 and 4 contain 100% (50.1%, and 49.9% respectively) of malformed packets (from a total of 174,607). This clustering of malformed packets likely occurs because of different types of errors. As part of the next steps to implement this approach in an IDS, we plan to add root cause information to those clusters in order to inform operators about their criticality. Finally, we also identified clusters 3 and 5 as being composed by packets with
responses, while the other clusters have an overwhelming percentage of unsolicited messages without replies.

As a final part of our analysis, we now visualize the clusters. While the feature vector lies in a four dimensional space, we can make a projection that maximizes the variance of the feature space. This projection of a 4-D vector to a 2-D vector can be achieved with Principal Component Analysis (PCA). Table 6.5 shows that keeping only the two most significant components retains 98% of the variance of the data.

<table>
<thead>
<tr>
<th># Comp.</th>
<th>Ret. Var.</th>
<th># Comp.</th>
<th>Ret. Var.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>78%</td>
<td>2</td>
<td>98%</td>
</tr>
<tr>
<td>3</td>
<td>99%</td>
<td>4</td>
<td>100%</td>
</tr>
</tbody>
</table>

Figure 6.5 depicts the 2-Dimensional space resulting after applying PCA with 2 components to the feature vectors; each cluster is assigned a different color. While in our initial analysis of the clusters we did not identify this clear separation of the network flows, upon looking at this PCA projection we noticed that the cluster on the right is cluster 0, and upon further inspection we noticed that this cluster is composed of over 90% of unsolicited messages (unidirectional flows) sent to the meters; therefore, the connections in cluster 0 stand out from the ones in other clusters by having a 1 in the flow dimension, and 0 for both, the timing and the size of the reply. This behavior is a clear discriminant feature of the network flows as only 7% of connections are initiated by the server (93% of connections are initiated by the smart meters); furthermore, only 9% of connections are followed by a response flow.

It is clear that the last feature, the direction of the flow, is the feature that dominates in this projection to two dimensions. To illustrate better the variety of clusters we decided to apply PCA to only the first three dimensions of the feature vector, the results can be seen in Figure 6.6. The remaining clusters are not as easily separable, but it is clear that they lie on different parts of the 2-D space.

6.5 Conclusions

This work offers an experimental study of different approaches that a network-based IDS for AMI could use to cope with encrypted traffic. An AMI has unique characteristics related to the portion of traffic and packets being encrypted, the number of service requests available, and the way session flows are handled that make this study an important step towards
Figure 6.5. PCA with 2 components retaining 98% of variance.

Figure 6.6. PCA of clustered packets with $k=6$
achieving the twin goals of confidentiality and security monitoring. In particular, we found that fingerprinting devices and watching for the periodicity of meter requests were good candidates to alert on suspicious activity. We also found that the 30,000-meter operational AMI used in our study was too dynamic to train a connectivity graph baseline over 2 days of traffic. By extracting feature vectors representing connection flows we were able to identify several anomalies (e.g., packets with multiple C12.22 payloads), and rules of “normal” behavior, such as the size of reply packets being lower than 200 bytes, or that bidirectional communications have request packets with sizes smaller than 800 bytes.

Table 6.6 provides a summarized mapping between the different approaches investigated in this work and the types of malicious behavior they can detect.

Table 6.6. Mapping between intrusion detection techniques and detectable attacks despite encryption.

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Period.</th>
<th>Fingerpr.</th>
<th>Graph</th>
<th>Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic tampering</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Traffic injection</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Replay attack</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Authentication abuse</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Spoofing</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rogue device</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compromised meter</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Resource exhaustion</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>

As the next step, we plan to increase the length of our data collection to confirm our exploratory data analysis, and to measure the security of IDS rules for encrypted packets against adversaries that will try to evade them.
CHAPTER 7
CONCLUSIONS AND FUTURE WORK

There have been few cases of cyber-attacks against Industrial Control Systems (ICS). However, the risk to computer attacks is increasing because their vulnerabilities are becoming more exposed and available. The biggest threat to control systems are targeted attacks. These attacks are the ones where highly sophisticated and resourceful attackers, who possess enough information about the control system, tailor their attack strategy with the aim of damaging the physical system under control. Finally, because ICS systems are integral part of national critical infrastructures, such as electric power distribution, oil and gas refining, and water treatment and distribution, their disruption could have a significant impact on public health, safety, and lead to large economic losses. Securing ICS in critical infrastructures is thus of national priority [143, 62].

It was not until 2007, with the work of Cheung et al. [33] that intrusion detection was considered as a plausible technique for ICS. 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 Information Technology (IT) networks. Following this claim, multiple anomaly detection approaches for ICS have been proposed, but it was Cárdenas et al. [30] who highlighted that the intrinsic characteristic of ICS systems is their unbreakable relationship with a physical process. In other words, traffic exchanged in a control network may, ultimately, effect an impact over the “real world”. This relationship can be leveraged not only to understand the deviations imposed by attacks over the normal behavior of a control system, but also for the creation of physical process models that help on their quantification. Finally, providing for natural mean on the attends for anomaly detection in ICS.

In this dissertation, we expanded on the recent and growing literature in computer security conferences, studying how Deep Packet Inspection of sensor values from physical observations, and control signals sent to actuators, can be used to build physical models that allow for the detection of attacks in industrial control networks. To the best of our knowledge, it provides the most extensive analysis of Physics-based Anomaly Detection in real-world ICS networks. We implemented multiple attacking and detection mechanisms, and analyzed
traffic of different ICS networks used in the utility domain: a room-sized water treatment testbed, a chemical plant simulation, a real operational water treatment utility, and a real operational electricity utility. We addressed two research questions:

**Main RQ:** How can Deep Packet Inspection be leveraged in order to achieve Physics-based Anomaly Detection in ICS? What are the implementation challenges? How can we evaluate physics-based anomaly detection in the presence of stealthy attacks?

In Chapter 3, we showed how an attacker can pragmatically implement attacks against the Field Communications Network of a control system. We depicted the challenges, information, and access necessary to introduce a Man-In-The-Middle (MitM) attack into the Device Level Ring topology of a room-sized water treatment testbed. An attacker who successfully deploys a MitM device into a ring established for field communications must be assumed to have access to all digital and analog signals. He could inject manipulated data in all sensors and actuators monitored and controlled by the ring’s PLC at any time. Therefore, effectively isolating and manipulating the physical process disregarding control actions sent by the control room or the PLCs. More robust communication infrastructures can be designed in order to decrease an attacker’s impact over the system.

In our Systematization of Knowledge in Chapter 2, we identified a general physics-based anomaly detection framework that captures most of the proposals available in the literature. We also provided a comprehensive taxonomy of related work, discussed general shortcomings we identified, and proposed new directions of research. We found in our survey in § 2.4 that the studied previous works, published in security conferences, cite at most three other physics-based attack detection works from the same table. This presentation of physics-based attack detection ideas without contextualizing them with the large body of previous work is a big limitation to progress in this field. To address this problem, and to help researchers identify related work, in our survey we included over 45 physics-based attack detection works presented under a unified taxonomy.

Moreover, in our survey we explained the limitations of previous metrics and adversary models, and in Chapter 4 proposed a novel stealthy and adaptive adversary model, together with its derived intrusion detection metric, that can be used to study the effectiveness of physics-based attack-detection algorithms in a systematic way. According to our survey, we are also the only research work conducting validation of our approaches in multiple setups, including: a room-size water treatment testbed, a real large-scale operational system managing more than 100 PLCs, and simulations of chemical plants and primary frequency
control in the power grid. We showed in Table 4.1 how each of these validation setups has advantages and disadvantages when evaluating the x-axis and y-axis of our proposed metric.

One result we obtained across our testbed, real operational systems, and simulations, is the fact that stateful tests perform better than stateless tests. This is in stark contrast to the popularity of stateless detection statistics as summarized in Table 2.1. We hope our work motivates more implementations of stateful instead of stateless tests in future work. We also showed that for a stealthy actuator attack, PI controls play an important role in limiting the impact of this attack. In particular, we showed that the Integrative part of the controller corrects the system deviation and forces the attacker to have an effective negligible impact asymptotically. Finally, we also provided the following novel observations: (1) finding spatio-temporal correlations of Modbus signals has not been proposed before, and we showed that these models are better than models of single signals previously proposed in the literature, (2) while input/output models like LDS are popular in control theory, they are not frequently used in works published in security conferences, and we should start using them because they perform better than the alternatives, unless we deal with a highly-nonlinear model, in which case the only way to limit the impact of stealthy attacks is to estimate nonlinear physical models of the system, and (3) we showed why launching undetected attacks in actuators is more difficult than in sensors.

In Chapter 5, we explained the difference and challenges between performing security monitoring at the Supervisory Control Network (SCN) versus at the Field Communications Network (FCN). We also 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. We then implemented a field security monitor. As far as we are aware, we are the first to implement a monitoring device in this lower layer of Industrial Control Systems.

Supplemental RQ: How to perform anomaly detection when in presence of encrypted ICS traffic?

Our work in Chapter 6 offers an experimental study of different approaches that a network-based IDS for AMI could use to cope with encrypted traffic. An AMI has unique characteristics related to the portion of traffic and packets being encrypted, the number of service requests available, and the way session flows are handled that make this study an important step towards achieving the twin goals of confidentiality and security monitoring. In particular, we found that fingerprinting devices and watching for the periodicity of meter
requests were good candidates to alert on suspicious activity. We also found that the 30,000-meter operational AMI used in our study was too dynamic to train a connectivity graph baseline over 2 days of traffic. By extracting feature vectors representing connection flows we were able to identify several anomalies (e.g., packets with multiple C12.22 payloads), and rules of “normal” behavior, such as the size of reply packets being lower than 200 bytes, or that bidirectional communications have request packets with sizes smaller than 800 bytes.

7.1 Future Directions

While physics-based attack detection can improve the security of control systems, there are some limitations. For example, in all our experiments the attacks affected the residuals and anomaly detection statistics while keeping them below the thresholds; however, there are special cases where depending on the power of the attacker or the characteristics of the plant, the residuals can remain zero (ignoring the noise) while the attacker can drive the system to an arbitrary state. For example, if the attacker has control of all sensors and actuators, then it can falsify the sensor readings so that our detector believes the sensors are reporting the expected state given the control signal, while in the meantime, the actuators can control the system to an arbitrary unsafe condition.

Similarly, some properties of the physical systems can also limit us from detecting attacks. For example, systems vulnerable to zero-dynamics attacks [139], allow an attacker with control of actuators to change the state of a system while our state estimator is not able to observe any changes in the variables. To reveal these zero-dynamics attacks we need to change the configuration of the physical system or add more sensors to estimated variables vulnerable to this attack. Other physical systems allow attackers to drive the system to an unsafe state fairly rapidly (in one time step) while delaying the visibility of these attacks in the residuals unbounded systems [140] (i.e., the attacker can always drive the system to an unsafe state before we can see any changes in the residuals). Finally, non-linear systems also pose a variety of problems; for example, chaotic systems can be driven from one state (attractor) to another one with very small control actions and highly non-linear or chaotic systems [112] (the butterfly effect), making detection of these small “attacks” very difficult.

One of the biggest challenges for future work is the problem of how to respond to alerts. While in some control systems simply reporting the alert to operators can be considered enough, we need to consider automated response mechanisms in order to guarantee the safety of the system. Some works in our survey cover attack-response (i.e., reconfiguration of the system), nevertheless, this is an area that has received relatively less focus. How to respond
to alerts will become a vital area of research once we agree on some of the attack detection fundamentals.

Likewise, we could also extend our metric to instead of measuring the false alarms, it would measure the impact of a false response. For example, the work of Cárdenas et al. [29] considered switching a system to open-loop control whenever an attack in the sensors was detected (meaning that the control algorithm will ignore sensor measurements and will attempt to estimate the state of the system based only on the expected consequences of its control commands). As a result, instead of measuring the false alarm rate, they focused on making sure that a reconfiguration triggered by a false alarm would never drive the system to an unsafe state. Therefore, maintaining safety under both, attacks and false alarms, will need to take priority in the study of any automatic response to alerts.

Finally, 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 started the discussion of integrating the information from multiple field monitors at different stages in a large process. More work it is necessary to uncover other sources of information that may be leveraged to improve the visibility of our system, and make the work of the attacker harder if he wants to remain undetected.
APPENDIX

INDUSTRIAL CONTROL PROTOCOLS

A.1 Modbus

The Modbus protocol [105] was originally designed by Modicon (now Schneider Electric) on 1979, and currently administered by the Modbus Organization Inc. responsible for its continuous adoption and evolution around the control system’s landscape. On its origins, it was designed as a straightforward way to transfer data between controls and sensors via RS-232 interfaces. Nowadays, Modbus supports other communication media, such as Ethernet networks.

![Modbus stack and its different type of networks.](image)

Figure A.1. Modbus stack and its different type of networks.

Modbus is an application protocol which defines a client and server communication pattern among devices connected on different types of networks (see Figure A.1). It defines rules for organizing and interpreting data, while remaining simply a messaging structure. Modbus introduces network-dependent Application Data Units (ADU) which encapsulate network-independent simple Protocol Data Units (PDU). Each PDU transports the request/response data being communicated between client and server devices.

The Modbus data model defines four primary tables which correspond to the input and output of single-bit and 16-bits data items, also called coils and registers, respectively. The read and write operations over data items are defined by a set of 255 functions, and each data item is addressed by a 16-bits unsigned integer. Depending on the function, ranges of data items may be addressed for reading or writing.

The Modbus TCP protocol is the same original Modbus protocol with a TCP interface that runs over Ethernet. In other words, a Modbus TCP/IP message is simply a Modbus
communication encapsulated in an Ethernet TCP/IP wrapper. In practice, Modbus TCP embeds a standard Modbus data frame into a TCP frame, without the Modbus checksum, as shown in Figure A.2.

Figure A.2. Modbus vs. Modbus TCP packet structures (Figure from [4]).

A.2 ANSI C12.22

The ANSI C12.22 protocol [13] was originally designed by the American National Standards Institute to transport electric utility metering data over “reliable” networking systems. Hence, allowing for interoperability and support of multiple manufacturers in the Smart Grid. The metering data structure follows the ANSI C12.19 specification [12] which defines a set of end device data tables.

The specification of ANSI C12.22 proposes the use of the Advanced Encryption Standard (AES) family of block ciphers to enable strong, secure communications, while guaranteeing confidentiality and data integrity.

A.3 Common Industrial Protocol

The Common Industrial Protocol (CIP) network specification library [110] was originally developed by Rockwell Automation and finally standardized and maintained by Open Device Vendors Association (ODVA) and ControlNet International. It aims to fulfill the main three needs of ICS systems: control, configuration, and collection of data [27]. It defines the CIP application layer protocol as a encapsulated object-oriented protocol for transmission of
connected (I/O implicit) messages between a data producer and one or more data consumer devices, and unconnected (explicit) messages between two devices in the control network. Transmissions associated with a particular connection are assigned an unique connection ID. While being an application layer protocol, CIP is independent of the underlying layers, and requires an encapsulation protocol which allows abstraction from different data link and physical layers. It also includes a Common Object library defining commonly used objects, some of which are specific for a particular encapsulation protocol, and allows for extension and definition of vendor specific objects. The CIP specification library includes the definition of 4 different CIP stacks depending of the physical layer in use (see Figure A.3): EtherNet/IP (over IEEE 802.3 Ethernet), CompoNet, DeviceNet, and ControlNet.

A.3.1 EtherNet/IP

The CIP stack introduces the EtherNet/IP protocol [111] for both, SCADA network and fieldbus communications alike. Its specification defines the Common Packet Format (CPF) for the encapsulation of message oriented protocols, such as CIP, Modbus, and vendor proprietary messages. EtherNet/IP CPF can be stacked over UDP or TCP, in both multipoint and point-to-point connection modes. When stacking over UDP, it requires devices to select a maximum of 32 consecutive addresses from the range 239.192.1.0 to 239.192.128.255 (which belongs to the Organizational Local Scope [51]), and specifies an algorithm for selection.

A.3.2 Device Level Ring

Because the EtherNet/IP protocol stacks over an Ethernet physical and data link layer, which by design does not support rings topologies, its specification includes the definition of
a Device Level Ring (DLR) protocol, to allow devices with double Ethernet ports to form rings. A DLR network is tolerant to all single-point failures, providing reliable and robust communications in a single-ring topology. In the ring, one of the connected devices is selected as ring supervisor, in charge of avoiding infinite cycling of packets and network reconfiguration in case of ring breaks or restorations. The ring supervisor and other ring devices follow a message protocol to identify which segment of the ring suffered the failure and reconfigure their MAC Address Tables to correctly redirect packets away from it, towards their expected destination.
REFERENCES


VITA

David Ignacio Urbina Fuentes was born in Caracas, Venezuela on December 29th, 1982. In 2006, he received his Bachelor of Science (B.Sc.) degree in Computer Science at the Simón Bolivar University (USB). During his bachelor studies, he was invited as guest researcher to the Texas Advanced Computing Center (TACC) at the University of Texas at Austin (UT Austin) to design and develop secure grid portals for the Teragrid and SUMA/G computing grids. In 2009, he moved to United States to pursue a Master of Science (M.Sc.) degree in Computer Science with a Major in Software Engineering at the University of Texas at Dallas (UT Dallas), for which he had been awarded with the Organization of American States Graduate Studies Scholarship. After achieving his M.Sc. degree in 2010, he re-enrolled at UT Dallas as a Ph.D. candidate for the Cyber Security Research and Education Institute (CSI). During his research time at UT Dallas, he was invited as guest researcher at the iTrust – Centre for Research in Cyber Security in the Singapore University of Technology and Design (SUTD), and at the National Institute of Standards and Technology (NIST) – U.S. Department of Commerce, where he performed research on protocol analysis, and design and development of physics-based anomaly detection systems for Industrial Control Systems.

A list of his publication in reverse chronological order:

- David I. Urbina, Jairo Giraldo, Alvaro A. Cárdenas, and Nils Ole Tippenhauer. Improving Visibility of ICS Network Monitoring Against Low-Level Attacks.


