A Hands-on Modular Laboratory Environment to Foster Learning in Control System Security

Pallavi Prafulla Deshmukh

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Cameron D. Patterson, Chair
William T. Baumann
Muhammad R. Hajj

February 16, 2016
Blacksburg, Virginia

Keywords: Embedded Systems, Cyber-Physical Systems, Control Systems, Cyber Security, Trust, Embedded Security, Modular Education, Security Lab, Hands-on Learning

A Hands-on Modular Laboratory Environment to Foster Learning in Control System Security

Pallavi Prafulla Deshmukh

ABSTRACT

Cyber-Physical Systems (CPSes) form the core of Industrial Control Systems (ICSes) and critical infrastructure. These systems use computers to control and monitor physical processes in many critical industries including aviation, industrial automation, transportation, communications, waste treatment, and power systems. Increasingly, these systems are connected with corporate networks and the Internet, making them susceptible to risks similar to traditional computing systems experiencing cyber-attacks on a conventional Information Technology (IT) network. Furthermore, recent attacks like the Stuxnet worm have demonstrated the weaknesses of CPS security, which has prompted increased effort to develop more effective security mechanisms. While this remains an important topic of research, often CPS security is not given much attention in undergraduate programs. There can be a significant disconnect between control system engineers with CPS engineering skills and network engineers with an IT background.

This thesis describes hands-on courseware to help students bridge this gap. This courseware incorporates cyber-physical security concepts into effective learning modules that highlight real-world technical issues. A modular learning approach helps students understand CPS architectures and their vulnerabilities to cyber-attacks via experiential learning, and acquire practical skills through actively participating in the hands-on exercises. The ultimate goal of these lab modules is to show how an adversary would break into a conventional CPS system by exploiting various network protocols and security measures implemented in the system. A mock testbed environment is created using commercial-off-the-shelf hardware to address the unique aspects of a CPS, and serve as a cybersecurity trainer for students from control system or IT backgrounds. The modular nature of this courseware, which uses an economical and easily replicable hardware testbed,

make this experience uniquely available as an adjunct to conventional embedded system, control system design, or cybersecurity courses. To assess the impact of this courseware, an evaluation survey is developed to measure the understanding of the unique aspects of CPS security addressed. These modules leverage the existing academic subjects, help students understand the sequence of steps taken by adversaries, and serve to bridge theory and practice.

A Hands-on Modular Laboratory Environment to Foster Learning in Control System Security

Pallavi Prafulla Deshmukh

GENERAL AUDIENCE ABSTRACT

Cyber-Physical Systems (CPSes) use computers to control and monitor physical processes in domains such as aviation, industrial automation, transportation, communication, water distribution, waste treatment, and power systems. These systems are increasingly connected with both private and public networks, making them susceptible to disruption or destruction by adversaries anywhere in the world. Stuxnet worm was a prime example of a sophisticated attack that managed to infiltrate and damage centrifuges in Natanz uranium enrichment facility in Iran. Such attacks have highlighted CPS vulnerabilities and motivated the development of CPS-specific defenses. CPS security is not given much attention in undergraduate programs because networks and control systems are traditionally distinct areas of study. We have developed hands-on courseware to show how an adversary can compromise a CPS in a Stuxnet-like manner by tunneling through a layered sequence of network protocol and interface weaknesses. The courseware incorporates cyber-physical security concepts into effective learning modules that highlight real-world technical issues. This modular learning approach helps students understand CPS architectures and their vulnerabilities to cyber-attacks via experiential learning, and acquire practical skills through actively participating in the hands-on exercises. This courseware uses an economical and easily replicable hardware testbed, making this experience available as an adjunct to conventional embedded system, control system design, or cybersecurity courses. To assess the impact of this courseware, an evaluation survey is developed to measure the understanding of the unique aspects of CPS security. These modules leverage existing academic subjects, help students understand the sequence of steps taken by adversaries, and serve to bridge theory and practice.

To my parents..

For their endless love, support and encouragement

Acknowledgments

I would like to express my deepest gratitude to Dr. Cameron Patterson for giving me the opportunity to work on this project. I have been very fortunate to have Dr. Patterson as my adviser. He gave me the freedom to work on my own, at the same time helped me overcome all the difficulties with his support and guidance. I am also very thankful to him for encouraging the use of correct grammar and coherent writing and for carefully reading and commenting on countless revisions of this thesis. I would like to thank Dr. William Baumann for his insightful comments and constructive criticisms that helped me improve my work. I am grateful to Dr. Muhammad Hajj for being a part of my advisory committee. I would like to thank my colleagues Teja, Omkar, Abhinav, Thomas, and Yihan for their support to make this project successful.

I am extremely thankful to my parents and family for their endless love and patience. None of this would have been possible without their constant encouragement and immense faith in me that helped me gain the drive and ability to face challenges head-on. Lastly, I would like to thank all my friends and roommates who made my journey filled with unforgettable memories, I will forever remember all the stories and moments we shared during our time together.

This material is based upon work supported by the National Science Foundation (NSF) under Grant Number CNS-1222656. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. Zedboards and design tools were donated by Xilinx, Inc.

Contents

List of Figures

List of Tables

Acronyms

AMP	Asymmetric Multiprocessing
ARP	Address Resolution Protocol
CLI	Command-line Interface
CPU	Central Processing Unit
CS	Computer Science
CPS	Cyber-Physical System
DOE	Department of Energy
DoS	Denial of Service
DCS	Distributed Control Systems
DDR	Double Data Rate
ECE	Electrical and Computer Engineering
FPGA	Field Programmable Gate Array
FSBL	First Stage Boot Loader
GUI	Graphical User Interface
HMI	Human-Machine Interface
HTTP	Hypertext Transfer Protocol
ICS	Industrial Control System
I/O	Input/Output
IOI	I/O Intermediary
IP	Internet Protocol
IT	Information Technology
JPG	Joint Photographic Experts Group
JSON	JavaScript Object Notation
MITM	Man-in-the-Middle
NBISE	National Board of Information Security Examiners
OCD	On-Chip Debugging
OCM	On-Chip Memory
PCS	Process Control System

PL Programmable Logic

PLC Programmable Logic Controller

PS Processing System

RIP Rotary Inverted Pendulum

RTU Remote Terminal Unit

SD Secure Digital

SoC System on Chip

SCADA Supervisory Control and Data Acquisition

SSH Secure Shell

TAIGA Trustworthy Autonomic Interface Guardian Architecture

TCP Transmission Control Protocol

TISN Trusted Information Sharing Network

TPM Trusted Platform Model

TR Trust Requirements

UART Universal Asynchronous Receive Transmit

UDP User Datagram Protocol

USB Universal Serial Bus

Chapter 1

Introduction

Cyber-Physical Systems (CPSes) have become an ubiquitous component of our everyday lives, with an immense range of applications including, but not limited to: medical devices, aviation, telecommunications, ground transportation, industrial automation, and power systems. These systems are often networked through IT infrastructure to allow for a reliable exchange of information among humans and machines. However, as these applications continue to increase, so does the potential for cyber-attacks that can cause system failures and result in potentially catastrophic events.

For example, in June 2000, a disgruntled former contractor for Maroochy Water Services in Australia used radio transmitters in his car to release one million liters of untreated sewage into local waterways [1]. In June 2010, Stuxnet, a 500-kilobyte computer worm, damaged centrifuges in Natanz uranium enrichment facility in Iran [2]. In December 2014, a malicious actor infiltrated a German steel mill facility using a phishing email to gain access to the corporate network, moved on to the plant network and caused unexpected conditions and physical damage to the plant [3]. All these attacks point out the risks and vulnerabilities of an Industrial Control System (ICS) [4, 5] and hence, it is crucial to understand the various challenges in securing a CPS.

1.1 Need for Modular Education

It is widely accepted that there is a critical need today to educate students in the field of cybersecurity with respect to such critical systems. A Presidential Executive Order [6], released in 2013, discusses the need to prioritize improved security of the critical infrastructures defined in Presidential Directive 7 [7].

More specifically, there is a high demand for cybersecurity training in the context of CPSes. However, this is limited by the extremely high costs and, consequently, the lack of testbeds capable of representing actual instantiations of a CPS. In 2003, multiple national laboratories collaborated to create the National Supervisory Control and Data Acquisition (SCADA) Testbed Program established by the Department of Energy (DOE), to provide their integrated expertise and resources in CPS security assessment, standards, and training [8]. This program offers cybersecurity training mostly for professionals in network/control systems and operators responsible for critical infrastructure [9]. Although this program is beneficial in many ways, it is limited by various factors such as time availability, geographical locations, and high training costs. These factors create a need for local programs which aim to build awareness and relevant skills across the critical infrastructure industries.

According to the Chief Executive Officer of the National Board of Information Security Examiners (NBISE), Michael Assante, the current security strategies are too disjointed, and there is a need to close the gaps in the software and system engineering foundations [10]. In his testimony on security issues in control systems, he criticizes the security research programs that are working on implementing yesterday's general IT security measures into today's ICSes and SCADA systems. He also argues that "these efforts were proven ineffective in general IT systems against more advanced threats." He also talks about the need for training individuals, and integrating security tools in a controls environment. The right exposure to system with large complexity and interconnectedness will help bridge this gap.

Thus, it is quite apparent that there is a need for remotely accessible, open-source programs that build awareness and relevant skills in the field of CPS security. Also, there is a strong demand for a hands-on modular educational approach that uses an economical mock testbed to delve more deeply into a wide range of topics in this area.

Modern control systems leverage many commercial-off-the-shelf IT components as shown in Figure 1.1. The Trusted Information Sharing Network (TISN) report on Achieving IT Resilience states that an IT infrastructure outage can impact critical infrastructure [11]. This is because the CPS engineers are often skeptical of the IT counterparts as the CPS systems have to be operational 24/7, whereas IT maintenance may require system outages; differentiating the CPS network services from the traditional IT network services. Hitherto, authors [12, 13] have highlighted this gap between the experts in network security with an IT background and the CPS specialists.

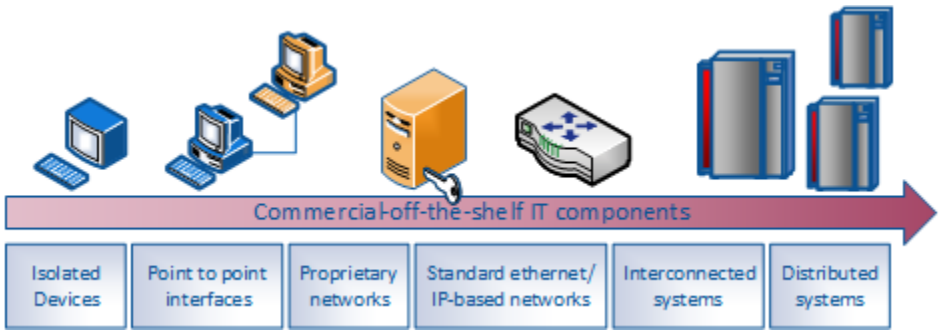


Figure 1.1: Modern control systems with specialized IT infrastructure

Crowley's [14] research offers a new perspective on IT security in education. According to Crowley's theory, security issues are dependent on various dynamic factors such as discovery of new vulnerabilities, publishing of new exploits, and the need for new countermeasures, making security a multi-disciplinary field. This theory can very well be extended to CPS security, as these systems offer unique challenges of a complex, interconnected system of the traditional communications systems as well as the physical plant. In order to understand the security issues in a CPS, it is crucial to perform a vulnerability assessment that aids in developing secure mechanisms to protect the CPS against various attacks.

Our aim is to build a foundational framework that is well-suited to advance cybersecurity emphasis on a CPS and encourage students to learn about the unique aspects of CPS security. Thus, the challenge is to develop hands-on learning modules from the perspective of maintaining reliable process control, including known exploits as well as attacks, to reinforce security concepts. Cyber-physical security education and training in control systems and other related programs is an excellent method of building awareness in students.

1.2 Previous Educational Work

As security has become a critical issue that affects our everyday life, there has been a consistent effort to develop courses that involve security training. Rowe et al. discuss the role of cybersecurity in IT education [15]. Raj et al. emphasize on increasing cybersecurity awareness by pushing security concepts in education [16]. Howles et al. discuss a holistic, modular approach to infusing cybersecurity in undergraduate degree programs [17]. Although modules have been a common pedagogical platform for cybersecurity [15–19], these do not highlight the real-world security issues in CPSes.

Navarro et al. discuss the use of cybersecurity in smart grid systems as an education tool in which they use a simulated environment to understand the penetration process that can compromise a SCADA system [20]. Yardley et al. propose a pedagogical framework to promote a modular learning platform in smart grid security [21]. It is clear, however, that these approaches do not focus on hands-on learning experiences.

In the past few years, there have been efforts to implement a SCADA security course in the academic curriculum [12]. This course focuses on understanding SCADA security concepts, and the need for SCADA system security in academia and their role in Australia's critical infrastructure. Although a good example of curriculum for CPS education, this is a specialist engineering course targeting a graduate program.

Other work has also been done to address the gap between theory and practice, as well

as the domain gap between IT and CPS experts [12, 13]. In one curriculum model, two educational tracks are included to cover engineering gaps for network specialists and IT gaps for engineering students.

There are a limited number of institutions that offer limited training programs for CPS security. InfoSec Institute offers a 5-day course in SCADA security architecture [22]. SANS Institute also has a five-day training course called ICS410 ICS/SCADA Security Essentials [23]. This course is designed for a broad audience to provide the foundational landscape of the security essentials for control systems.

1.3 Integration into Existing Programs

Cybersecurity concepts are generally addressed in Computer Science (CS) and related disciplines mainly focusing on the security aspects of traditional computing systems. The control systems that are deployed in industry and their associated functionalities are addressed in Electrical and Computer Engineering (ECE) and similar programs. Consequently, this has resulted in the separate treatment of cyber and control aspects in the CPS industry; with CS engineers focused on process security and control engineers responsible for process reliability. However, there is an important distinction between protecting traditional computing systems and safeguarding physical infrastructure, and lumping them together as cybersecurity is not a viable solution [24].

To address this issue, it is imperative to look at a CPS as a whole, integrating its cyber and physical aspects, and design an open interdisplinary approach for an audience from either end. This project aims to attract undergraduate students to the area of cybersecurity by incorporating an extensive research experience with the use of hands-on remote labs that target control systems security and can supplement an embedded systems or control systems class. The idea of hands-on modules to learn about CPS Security is a key step to broaden awareness and learn good practices in CPS security.

1.4 Contributions

In this thesis, we propose a holistic, modular approach to emphasizing cybersecurity in control and computing courses at the undergraduate level. We exploit the layered structure of a networked CPS to develop methods to potentially compromise the process stability. While the previous research work, Trustworthy Autonomic Interface Guardian Architecture (TAIGA) [25, 26], describes a secure mechanism to maintain process stability and safety, this work focuses on emphasizing the presence of vulnerabilities that exist in each layer of this system. This series of modules will contribute to enhancing the understanding of fundamental components of security and raise students awareness and skills in a area of CPS security.

The key contributions in this thesis are as follows:

1. Formulation of incremental modules: We develop hands-on modular courseware to understand the sequence of steps taken by an adversary to break into a conventional CPS system.
2. Understanding CPS architectures: We analyze, identify and exploit existing vulnerabilities in various network protocols and security measures implemented in a CPS and identify ways to mitigate these vulnerabilities.
3. Creation of a mock testbed: We create a mock testbed to replicate key components of a CPS to enable hands-on exercising of these modules. We demonstrate these individual modules in each layer of a CPS, using a Xilinx Zynq-7000 All Programmable SoC as a PLC, and a Raspberry Pi as a Remote Terminal Unit (RTU). A Quanser Rotary Inverted Pendulum (RIP) constitutes our physical process.

1.5 Thesis Organization

This thesis is organized in 9 chapters.

Chapter 1 describes the existing educational research and the need for hands-on modular education with a focus on security in CPSes and also highlights the contribution of this work.

Chapter 2 provides the relevant background to establish the research goals of this work. It describes the previous work, TAIGA, to enhance process safety and stability.

Chapter 3 gives an overview of our mock CPS built with commercial off-the-shelf components and how it maps to the different layers of a CPS.

Chapter 4 discusses the vulnerabilities in each layer and a possible roadmap for incremental attacks to penetrate into the lowest layer (controller) that interacts with the physical process.

Chapters 5 to 7 discuss the different modules in the network, regulatory and reconfiguration layers respectively.

Chapter 8 evaluates the effectiveness of these modules by recording students' feedback. It also discusses the experience gained from these modules.

Chapter 9 concludes with what was achieved through this research and discusses the future scope of integrating these labs with a course.

Chapter 2

Background

A CPS can be described as an integration of computation, networking and physical processes [27]. Embedded controllers govern the interaction between these three elements. They can be characterized as closely integrated, resource-constrained, networked, dynamically reconfigurable systems [28]. CPS leaf nodes enable computation and control element interaction. Figure 2.1 shows an overview of a conventional CPS.

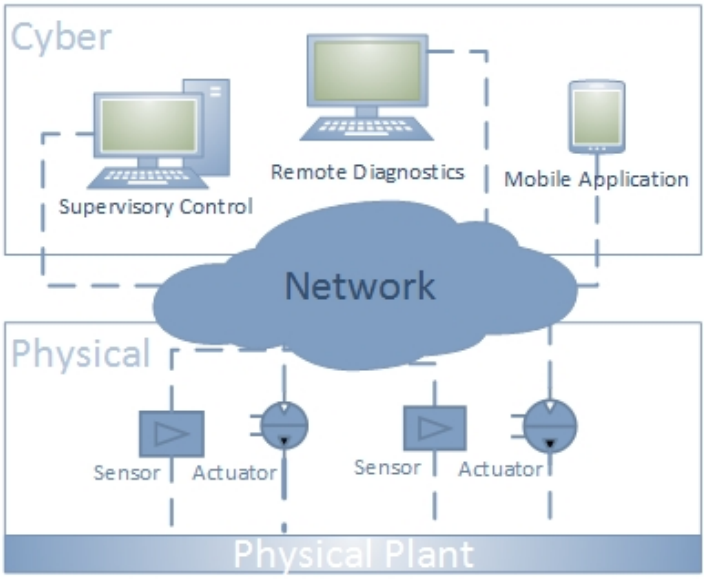


Figure 2.1: A Cyber-Physical System

The complexity of the dynamics of closed-loop controlled systems (the physical process) governed by the actions of the networked components (arbitrary or malicious actions) makes security an important aspect addressable not only on the cyber relationships but also the physical relationships of CPSes.

2.1 CPS Topology

CPSes are organized into multi-layer hierarchies that contain numerous embedded control nodes with different functionalities. Distributed Control Systems (DCS) or SCADA systems contain a defined set of subsystems and protocols for communication between systems. A CPS has three layers of control and computation (see Figure 2.2):

- The lowermost level in the hierarchy is the *infrastructure layer* that contains sensors, actuators, and the physical process.
- The *regulatory layer* includes a PLC and a RTU that govern the physical process through interaction with the supervisory and plant layers.
- The *supervisory layer* consists of a SCADA or a Human-Machine Interface (HMI) that monitors as well as controls the system via the regulatory controllers.

The components in the regulatory layer, RTUs, and PLCs are custom embedded controllers that interact with the supervisor through the IT network telemetry. A similar network channel exists for reconfiguration and remote diagnostics used for firmware updates and troubleshooting issues. The IT network layer connects to the regulatory layer making the embedded controllers susceptible to malicious reconfiguration that not only violates the network integrity of the channel, but also threatens the integrity of the physical process since the regulatory layer interacts directly with the components of the plant infrastructure.

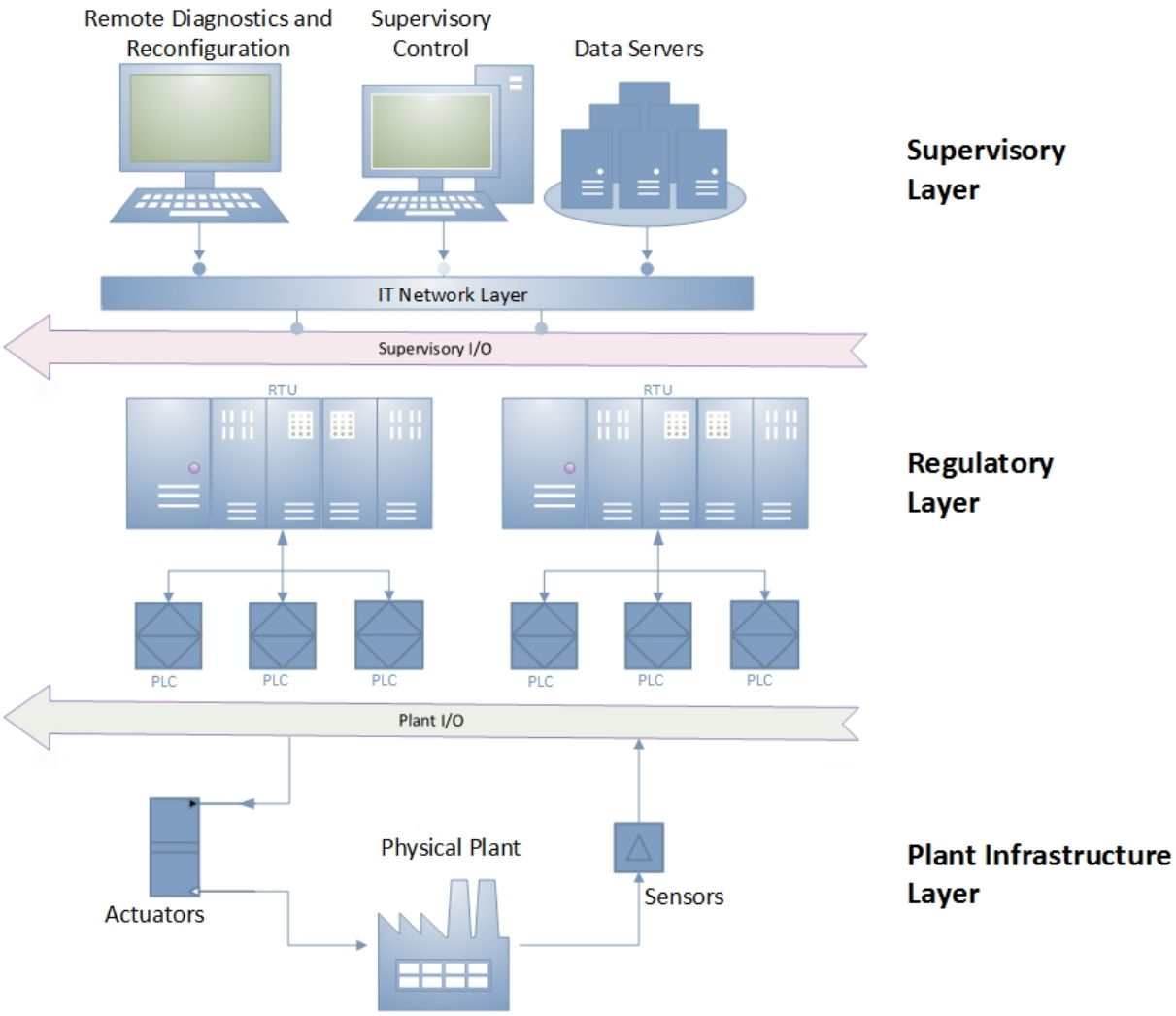


Figure 2.2: Hierarchical topology of a DCS or SCADA system

2.2 Cyber Threats to CPSes

CPS security and IT security have been treated as independent fields, with less consideration given to characterizing their combined vulnerabilities [29]. CPS security is typically five to ten years behind traditional IT security [30]. Although it may seem that the approaches to security in CPS or IT are very similar, they are developed to serve substantially different purposes. The proliferation of CPSes to offer remote access and monitoring has resulted in a convoluted mix of these two fields. Traditional IT security technologies, previously designed

to protect the exchange of information between cyber components, are being deployed to protect physical processes on CPS networks. Firewalls and Intrusion Detection Schemes can work well for safeguarding information in an IT environment; however they are not sufficient for protecting the processes controlling the critical infrastructure. Cardenas et al. discuss how a combination of control and security practitioners can work together to detect and survive even stealthy attacks [4].

Traditional cyber defense techniques are often reactive to attacks, and cannot promise protection against unknown exploits [31]. These security measures often assume that the embedded controllers are protected by the network, thus unaffected by Trojans or code modifications [32]. However, these controllers that manage the physical processes often include inadequate security measures. Often built using custom or proprietary protocols and third party software, these devices can easily hide Trojan behavior and vulnerabilities [33]. Adversaries attempt to exploit this lack of sufficient security measures or vulnerabilities in the existing infrastructure to cause damage. CPS attacks can be classified into reconfiguration attacks that target malicious reconfiguration of embedded platforms, and network integrity attacks that exploit the IT network vulnerabilities.

2.2.1 Case Study: Stuxnet Worm Attack

Usually, reconfiguration attacks are a result of a combination of several network integrity attacks that attempt to advance an adversary's intrusion into the system to access reconfiguration resources. The Stuxnet worm was a prime example of a sophisticated attack that managed to infiltrate a complex and interconnected control system. It was designed to sabotage industrial processes controlled by Siemens SIMATIC WinCC and PCS 7 control systems. It exploited at least four zero-day vulnerabilities to install, infect and propagate itself and was covert enough to evade state-of-the-art security technologies and procedures [2].

The Stuxnet worm initially targeted Microsoft Windows machines and proliferated itself over networks, searching for systems that were used to program the ICSes. It ultimately infected

the Siemens software that reconfigures the PLCs, which controlled the centrifuges, which resulted in the fast-spinning centrifuges tearing themselves apart [2]. The destruction of the centrifuges along with the forced shut down allegedly delayed Iran's ability to produce highly enriched uranium.

Rise of DuQu

Duqu is essentially the precursor to a future Stuxnet-like attack [34]. Unlike Stuxnet, that was created to attack ICSes, Duqu is an information stealer rootkit. One hypothesis suggests that Duqu was created by the same authors or those who had access to Stuxnet source code [35]. According to the literature, the primary purpose of Duqu is to gather information from industrial infrastructure and system manufacturers to direct a future attack against a third party. It is known to have 20 confirmed victims. In two instances, the attack was executed by exploiting a zero-day kernel vulnerability using a Microsoft Word document targeted through an email. The zero-day exploit was able to install Duqu; it took advantage of TrueType Font (TTF) parsing via Microsoft Word. Further, the attackers used Duqu to install another infostealer that can record keystrokes and collect other system information [34]. Duqu is configured to run for a certain period (30 days) by default, and then automatically remove itself from the system, thus preventing possible discovery.

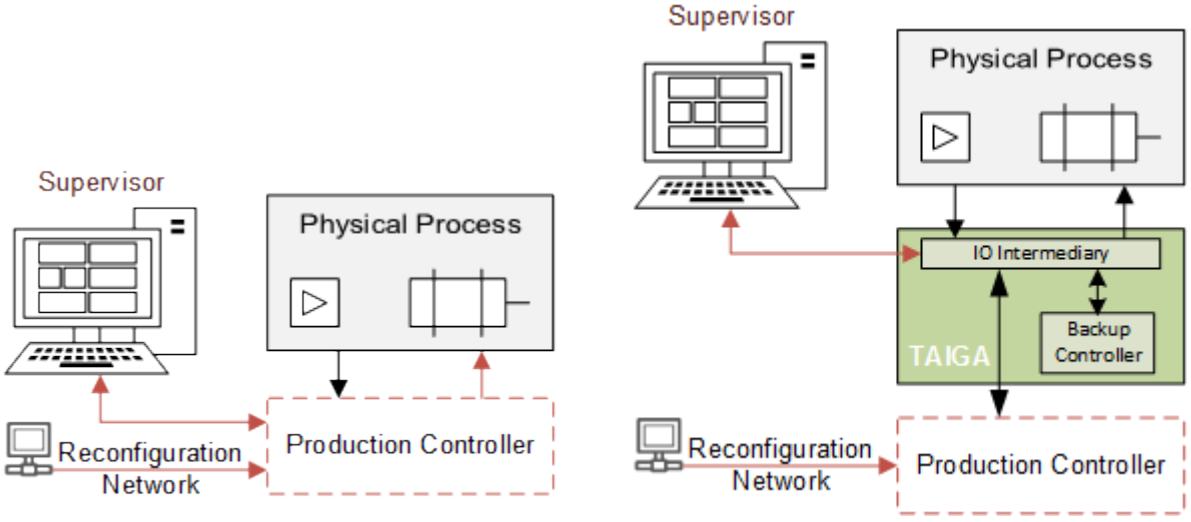
2.3 Overview of Prior Work

CPS security is a complex issue, and solving this requires examination of a broad field of challenges. As a result research in the field involves different focuses, extending from security protocols, authentication schemes and algorithms, to firewalls, intrusion detection systems, SCADA security and so on [36]. As the awareness about CPS security continues to increase, so does the research in this area. Previous work at Virginia Tech, Trustworthy Autonomic Interface Guardian Architecture (TAIGA), is used to preemptively detect malicious controller

behavior to protect a physical process [37]. The TAIGA architecture uses an additional trusted hardware component to serve as the last line of defense to prevent physical harm. In this section, we briefly discuss the development of TAIGA.

TAIGA introduces as an intermediary module for controller input/output acting as the sole means of communication from a PLC or RTU to any sensors or actuators. It acts as a physical layer of trust between the controller and the physical process. TAIGA not only has isolation from untrusted components, but also provides an architecture that monitors plant behavior and supervisory commands to preemptively respond to attacks [25].

TAIGA is implemented in the Programmable Logic (PL) of the Xilinx Zynq-7000 SoC. Figure 2.3 shows the TAIGA as the mediator interface and the last line of defense to the protect physical process by monitoring the commands issued by the physical controller.



(a) Conventional CPS leaf nodes (b) CPS leaf nodes with TAIGA

Figure 2.3: CPS leaf nodes with TAIGA

If the production controller issues a command that will lead to unacceptable plant behavior, TAIGA switches over the control to the backup controller, which is instantiated in the PL, thus maintaining plant safety. In most applications, it may be reasonable to have the backup controller drive the system to a predetermined safe state to prevent any damage to

the physical system.

2.3.1 Control Strategy

TAIGA incorporates both an untrusted production controller and a trusted backup controller that are isolated and run concurrently. Guards continuously monitor the plant state, and may trigger a switchover from the production to the backup controller.

Production Controller

The production controller maintains the physical process close to the desired set points. Contained within a PLC leaf node, the production controller runs the control algorithm to meet the performance and throughput specifications of the plant. This controller is prone to stealthy reconfiguration attacks as the parameters and even code may be periodically updated.

Backup Controller

A backup controller is a high-assurance controller that provides reliable and stable plant operation. [25]. This controller trades off performance for reliability. In contrast to the production controller, this controller does not undergo updates and is verified to not have any hidden malware. Thus, the primary purpose of this controller is to simply maintain plant stability and safety in the presence of an attack on the production controller.

Guards

Guards define the operational and safety bounds of the physical system, and can be categorized as follows:

- *Safety critical guards* define the operational limits of the system for safe operation. These could include the absolute limits for stable operation; violation of these guards can often involve difficult procedures to bring the system back to normal safe operation.
- *Operational guards* define the normal limits of the physical system. These guards are intended to prevent a safety critical guard violation.

2.3.2 Detection and Mitigation of Attacks

The trigger mechanism is responsible for the switch-over from the production controller to the backup controller in the presence of any malicious activity [38]. The plant's sensor values are stored, a process state model is maintained and used to predict the plant's future behavior. The prediction allows preemptive switchover to the backup controller before process stability and safety are risked.

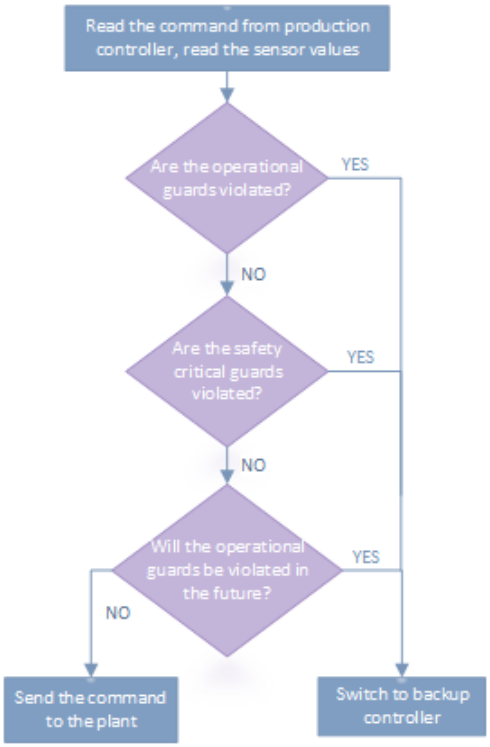


Figure 2.4: Trigger mechanism to guard against plant safety violations

As shown in Figure 2.4, TAIGA uses the following control logic to detect possible attacks. First, the control action taken by the production controller is checked. Set-points that would violate safety critical or operational guards cause a switch to the backup controller. These checks consider the future effects of the production controller's action. Thus, if the production controller is inactive or ignoring the set-points issued by the supervisor, the backup controller will be invoked.

The trigger mechanism's ability of this mechanism to accurately and reliably forecast the plant's future state depends on the state vector and the effectiveness of the plant model to describe the plant dynamics.

2.3.3 Isolation of Trust

As shown in Figure 2.3(b), TAIGA is inserted between the regulatory and plant layers. TAIGA is invisible to the production controller. In order to be considered trustworthy, TAIGA components are required to have formal trust requirements (TRs) [31]:

TR1 The source code and implementation for the entire component are analyzed.

TR2 The component uses private hardware resources for computation, internal communication, and memory, and does not invoke external components as sub-functions.

TR3 All external communication with untrusted components is through hardware-implemented, bounded, and isolated queues.

TR4 The component cannot be bypassed or disabled, and has a fixed repertoire of essential services, such as I/O or cryptography.

TR5 Critical functionalities of the component, such as rule checking logic, cannot be updated without provably secure or physical access.

A device that satisfies these TRs can be considered trustworthy throughout its lifetime. A Trusted Platform Model (TPM), primarily used as a secure cryptoprocessor, is the only other commercial security framework that fulfills all five TRs [39]. The separation of trust between the production controller and TAIGA components ensures that TAIGA will function properly even if the production controller is corrupted.

Chapter 3

A Mock SCADA System

A CPS is an integration of different computational, networking and physical elements. These systems are organized into multi-layered hierarchies to achieve timely information exchange, computation, and process control. CPS leaf nodes are interfaced through RTUs for interaction with the IT infrastructure. Conforming to the conventional cyber-physical control, our testbed implements regulatory controllers interfaced with the cyber network and a Rotary Inverted Pendulum (RIP) application as the physical process integrated into a CPS topology as shown in Figure 3.1.

The ZedBoard resembles a PLC leaf node with supervisory and reconfiguration channels interfaced through an RTU. This node contains the control loops that govern the physical process, the RIP system in our testbed. As shown in Figure 2.1, an RTU mediates the communication between the ZedBoard and IT network. This chapter elaborates on the implementation of each of these components and their functions in different layers of the system.

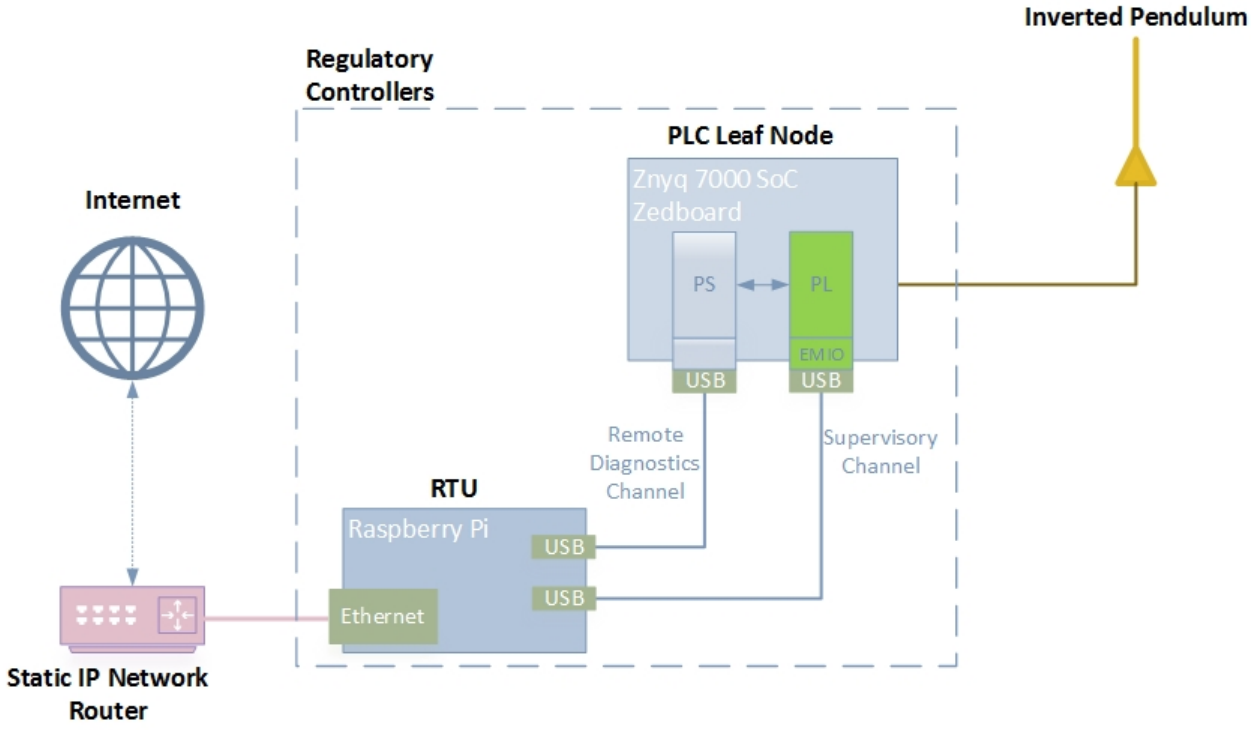


Figure 3.1: Testbed with a CPS topology

3.1 Rotary Inverted Pendulum

The RIP example is a classical control experiment representative of a CPS. It is an electromechanical system incorporating various characteristics of a CPS like non-linear dynamics, open-loop instability, actuator saturation,and noise. The Quanser RIP system is used in our testbed as the physical process of the system [40].

The Quanser RIP contains a flat arm with a pivot at one end and a metal shaft at the other. The pivot-end is mounted on top of the rotary servo base unit load gear shaft. The pendulum link is fastened onto the metal shaft, which is instrumented with a high-resolution encoder to measure its angle. The result is a horizontally rotating arm with a pendulum at the end. Figure 3.2 shows the pendulum in a balanced inverted position. The rotary servo base unit contains a DC motor that actuates the pendulum arm radially, and high-resolution optical encoder for sensing servo arm position. A linear voltage amplifier drives

the electromechanical pendulum system.

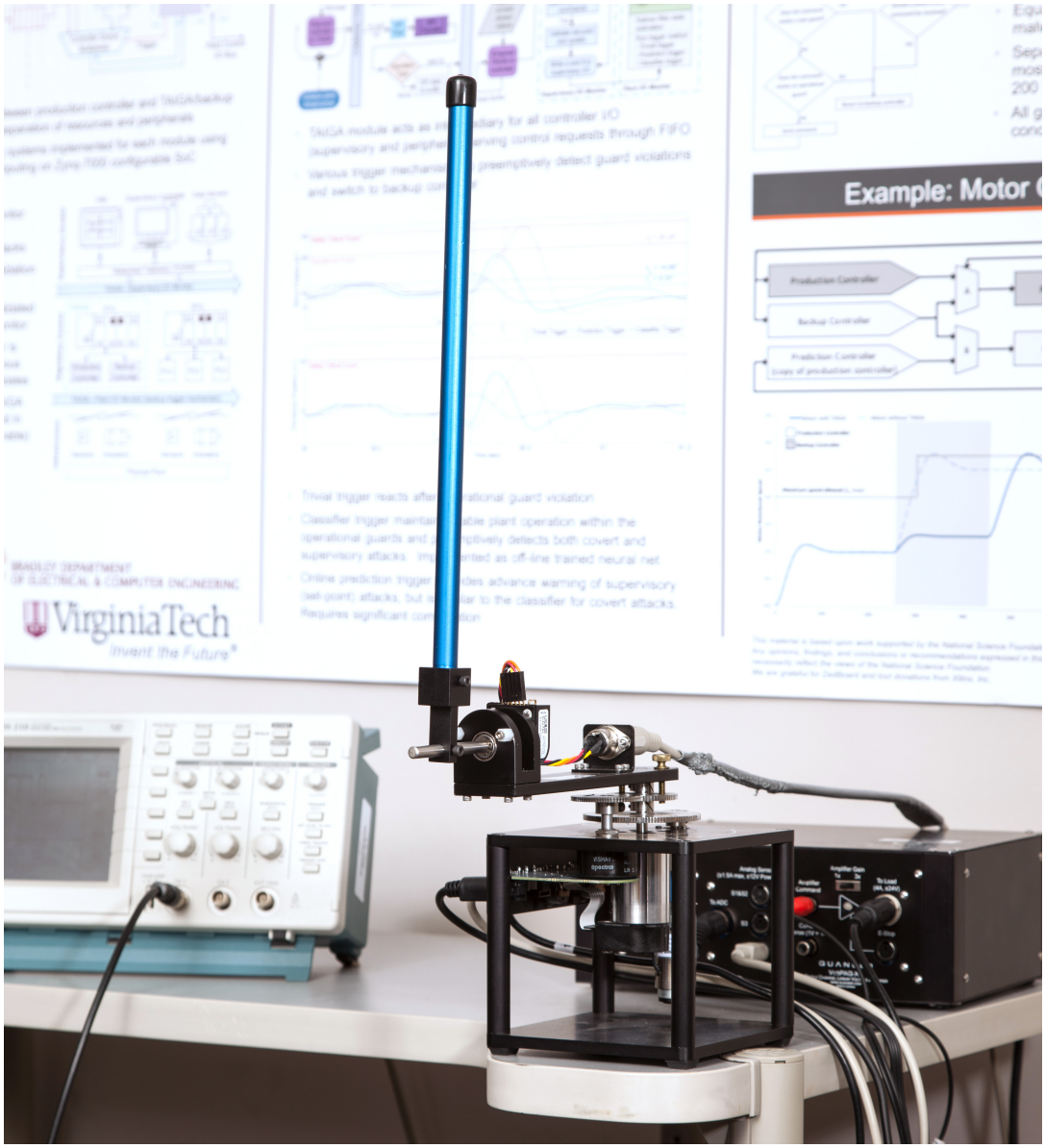


Figure 3.2: Quanser rotary inverted pendulum in a balanced condition

The complete RIP system represents the plant infrastructure layer of the mock testbed.

3.2 Programmable Logic Controller

The Zynq-7000 System on Chip (SoC) in the ZedBoard serves as the leaf PLC node. This component is the computational element that interacts with the plant to execute the control algorithm, receive sensor data and provide actuator signals. Typically, a PLC governs an individual process control loop contained within the entire plant, which is realized in the regulatory layer of the CPS.

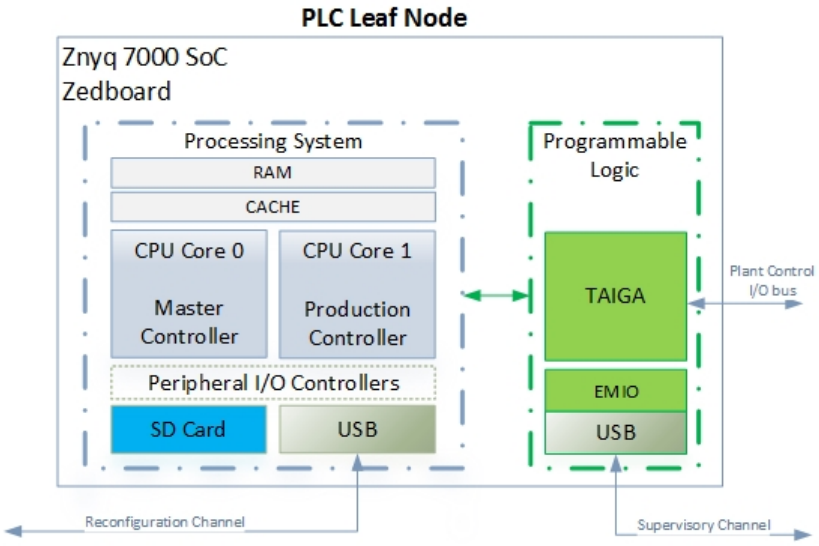


Figure 3.3: PLC realization on a configurable SoC

Figure 3.3 illustrates the PLC implemented on a configurable SoC platform. The Zynq SoC contains a dual-core ARM Cortex-A9 processor, and is configured to run independent software stacks on each of its processor cores using Asymmetric Multiprocessing (AMP). The Processing System (PS) implements a master controller and a production controller. Processor core 1 is the production controller that hosts instances of the control algorithm for actuating the process. Processor core 0 is the master controller responsible for system initialization including booting of communication with core 1. Core 0 also hosts the communication interface with the RTU. This communication interface serves as the remote diagnostics/ reconfiguration channel and is implemented on a dedicated Universal Asyn-

chronous Receive Transmit (UART). UART is connected to UART/USB converter chip, so although the interconnection of boards is shown as USB, PLC node communicates to the RTU over UART.

The system boots from an the SD card. The master controller implements the file transfer methods to allow communication to the PS that serves as a reconfiguration network for modifying certain parameters and executing firmware updates of the entire PS image. In our implementation, these updates require modification to the boot images stored within the SD card using file transfers with the RTU over a UART. The peripheral controllers for interfacing with the SD card and UART are enabled within the PS as shown in Figure 2.3.

Remote Diagnostics Channel

The master controller in the PLC has a remote diagnostics channel that acts as a troubleshooting unit. It is important to monitor the operation of an PLC to find problems that can be either internal to the PLC processor or in the I/O systems. Industrial PLCs provide functions for debugging and troubleshooting the PLC software. In our testbed, this is realized by providing a memory debug interface providing a read/ write channel.

Reconfiguration Channel

A reconfiguration channel is required to provide firmware updates and performance optimization for the embedded controller. In our testbed, this channel is realized using the UART channel between the Raspberry Pi and the production controller to store a new system image on the Secure Digital (SD) card. A newly transferred image takes effect after the system is rebooted.

This channel, responsible for both remote diagnostics and reconfiguration, is susceptible to malicious activity if the RTU is compromised. Thus the network integrity attack space is very critical since these regulatory layer components directly interact with the components

of the physical plant infrastructure. An adversary can use the reconfiguration channel to update the system files and exploit the memory debug channel to gain system knowledge, controller dynamics, and process behavior. This can allow the adversary to better conceal the attack.

3.3 Remote Terminal Unit

In a DCS or SCADA system, an RTU interacts with the PLC over a simple telemetry channel to relay messages with the IT network. The PLCs are the lowest-level embedded platforms that do not directly interact with the IT supervisory layer. A Raspberry Pi implemented the RTU as illustrated in Figure. 2.1, and runs an optimized flavor of Linux called Raspbian that combines network services and lower-level telemetry channels. As shown in the figure, the reconfiguration network and supervisory I/O network are both implemented on dedicated UARTs.

3.3.1 Remote Monitoring

The Raspberry Pi hosts a web server that transmits the process data to remote clients. A User Datagram Protocol (UDP) is implemented on the Raspberry Pi using Node.js, an event-driven JavaScript environment. The webserver receives the information packets on the supervisory UART channel, assembles the process states, and then transmits it to clients requesting the information. A client sends a intial setup message to the web server that adds the client's IP and port to its internally maintained client list. Once added, new data is streamed to all the clients at each control cycle.

The Raspberry Pi is a private node connected to a router with a static IP address, 128.173.52.36. The UDP web server is opened on port 32392. The server manages the transmit/ receive data requests on this port. This port is forwarded via the router, making this web server

accessible to any clients on the Internet.

Processing GUI

The web server is standalone and isolated from any applications of the client. For client software, a GUI is designed to represent the pendulum state in real-time and control the position of the pendulum for the RIP application. The GUI, as illustrated in the Figure 3.4, is implemented in the Processing development environment [41]. Processing is an open-source project animating a sketch that will display all of the information sent by the Node.js script.

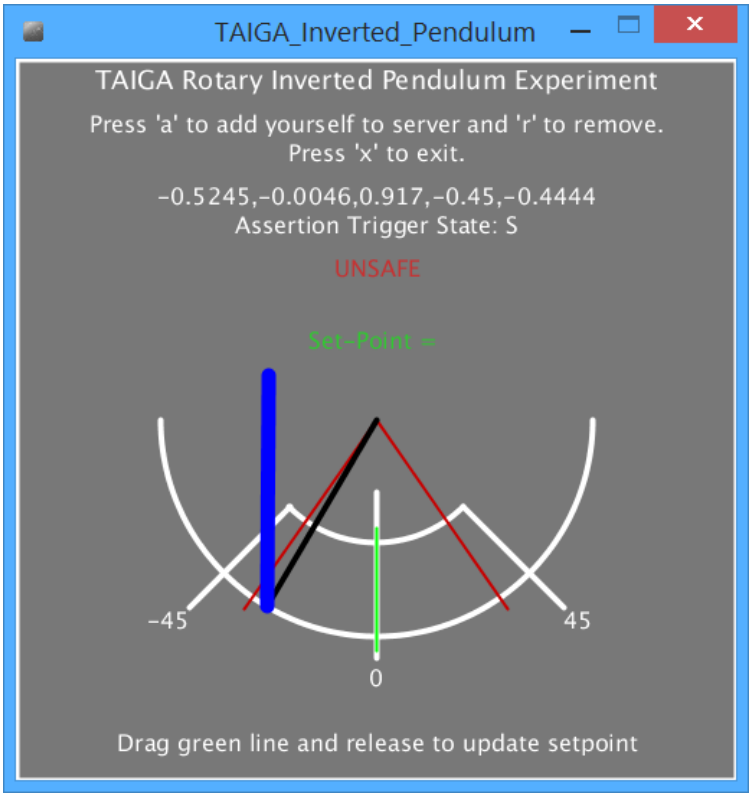


Figure 3.4: Processing GUI for remote control and monitoring

A client requests process data by pressing ‘a’ on the GUI application. The client’s IP address and port are then added to the server’s client list, and data is transmitted to the client at every control cycle. The current state vector and voltage is displayed on the GUI. The GUI

also displays whether or not the trigger is asserted. As shown in Figure 3.4, the pendulum is projected into 2-D to visually represent the RIP operation. The black line represents the servo arm that rotates along the center axis based on the θ position, whereas the thick blue line represents the pendulum that protrudes from the end of the servo arm vertically. The pendulum also oscillates along the vertical axis based on the current α position.

The GUI also provides an option to select the desired set-point for the pendulum. The green line, as shown in Figure 3.4, can be moved radially along the θ axis to set the new desired operational set-point. Once the mouse is released on the desired set-point, the new set-point is displayed on the GUI and it is transmitted to the UDP server. If the trigger is not asserted, these actions are governed by the production controller and can be manipulated from the supervisory channel. But in the case of any malicious activity, the trigger is asserted, the pendulum oscillates at the pre-programmed set-point, and all the set-point commands from the supervisor are ignored.

Remote Surveillance

Most of the time, safety-critical systems are under remote surveillance using networked cameras for enforcing perimeter security. In the RIP application, a Raspberry camera module is used to stream live images when the setup is in operation. The camera is connected to the USB port on the Raspberry Pi where a video streaming service is configured in Linux and configured to broadcast on port 8081. The webcam server's port is forwarded to the public IP via the network router similar to the UDP web server. A live feed can then be accessed remotely via the Internet by directing a browser to the webcam server's IP address and port: 128.173.52.36:8081.

A client-side live feed is shown in Figure 3.5. The web surveillance provided by this web camera is a secondary means of monitoring the RIP application. Data from the supervisory channel is not relayed. In the case of a security breach, adversarial access to the RTU is possible, which can compromise the supervisory channel communication. In these situations,

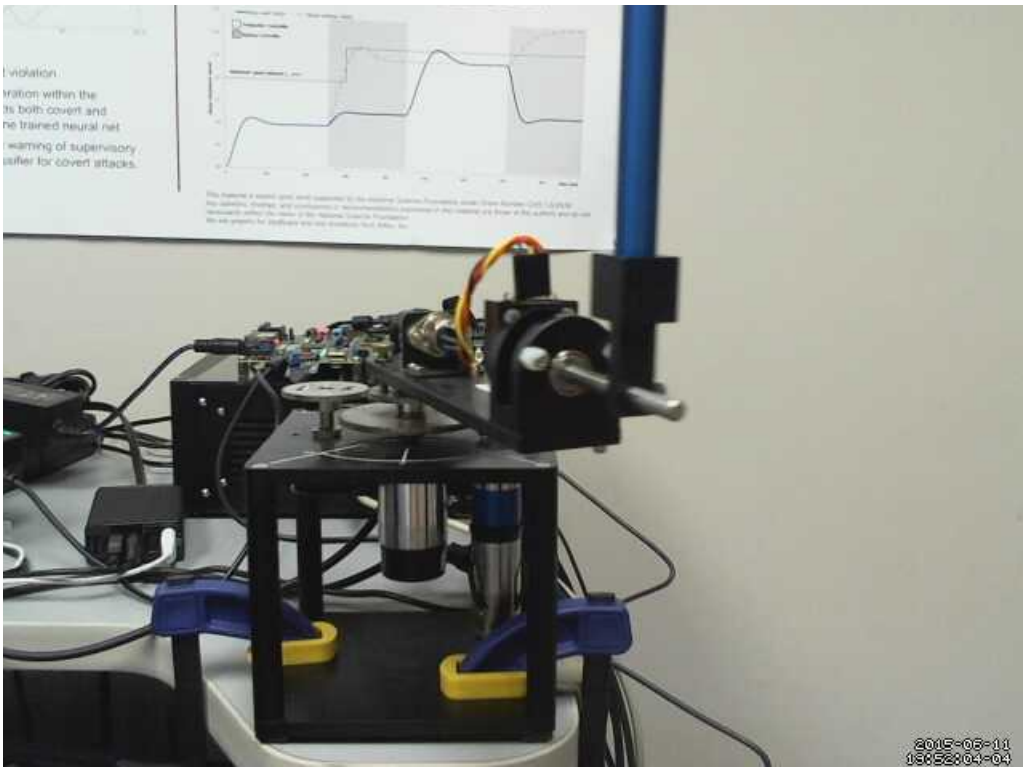


Figure 3.5: Remotely accessible live camera feed

replay attacks on the web camera can transmit old camera data to the supervisor while mounting an attack on the PLC. Such replay attacks can leave the supervisor unaware of an attempted attack on the process. Similar attacks can also occur on the Processing GUI feed to replay process data. A combination of replay attacks along with the exploitation of the reconfiguration channel can lead to a more stealthy attack causing physical damage to the system.

Chapter 4

Overview of the Modules

4.1 Learning Aspects

All computing and technology programs should ensure a thorough and pervasive security curriculum within their courses. Not only would the lack of a strong security curriculum disadvantage students, but it would also be detrimental to have students trained with advanced technical skills but lacking in security awareness. Such an exclusion would almost certainly result in the systems designed by these graduates containing potential security vulnerabilities.

It is widely accepted that learning-by-doing is one of the most effective ways to learn [42,43]. Authentic learning can be implemented by incorporating hands-on and relevant real-world problems in the curriculum. Hands-on experience is a good learning tool [43]; Ma et al. studied different methods of laboratory instruction and concluded that hands-on and remote labs help to build design skills and conceptual understanding in students [44]. The labs we describe here allow students to select tools and methodologies to come up with open-ended solutions to better understand the importance of cybersecurity in real CPSes. As emphasized by Rowe [15], these labs should not be exclusively connected to a particular technology, but

rather should focus on concepts, methodologies, and skills. Extending this premise, the CPS security labs focus on using many different modules to understand a CPS environment and its vulnerabilities.

Reflective Practice

The main idea is to implement a reflective practice approach that can refocus students' skills and knowledge to these hands-on security modules. Reflective learning involves using practical experiences and developing skills that are essential for problem solving and decision making [45]. Using a reflective approach, students can establish a theoretical knowledge base reinforced with practical hands-on modules. The modules can be aligned with Hinett's four-step [46] reflective approach that enables students to:

1) *understand what they already know*
   The modules introduce the CPSes and leverage the students' understanding of different network protocols and tools.
2) *identify what they need to know in order to advance understanding of the subject*
   The modules establish a roadmap to introduce the CPS system architecture and present the various components.
3) *make sense of new information and feedback in the context of their own experience*
   The modules actively engage the students in a series of incremental tasks and collect reflective feedback on this newly gained knowledge.
4) *guide choices for further learning*
   These modules can help encourage students to relate this experience to industry practice and open new cybersecurity outlooks.

4.2 Organization of the Modules

The modules are organized in an incremental fashion to understand the vulnerabilities in each control system layer penetrated to gain access to the leaf-node controllers. Each module includes the following components: Learning Objectives, Tools, Assumptions, and Hands-on Exercises.

One of the key takeaways from this example is how complex and interconnected a typical control system is. We know that potential pathways exist from the outside world, through the enterprise IT network and down to the process controllers. As illustrated in Figure 4.1, Eric Byres et al. summarize various pathways in an attack graph that Stuxnet used to reach its target processes [47]. We must also note that Stuxnet could have completely bypassed a few of the possible attack pathways. For example, a PLC programming laptop could have been compromised at another site and might have been directly used in the control network to program PLCs. Alternatively, the infected USB storage drive might have first compromised one of the support stations and gained direct entrance to perimeter or process control networks. To secure critical infrastructure, it is important to understand the complexity of these systems that are now the target of Stuxnet-like attacks. In particular, security programs need to:

1. Consider all possible infection pathways and develop strategies for mitigating those pathways.
2. Recognize no protective security measure is perfect, and take steps to understand the vulnerabilities in each of the CPS layers to limit the consequences of a compromise.
3. Look beyond the traditional network layer security and towards securing the local communication between a PLC and RTU.
4. Include security assessments and testing as part of system development. Perimeter security should be augmented with an independent last line of defense.
5. Work to improve the awareness in security of critical infrastructure and enable students to build relevant skills across industries.

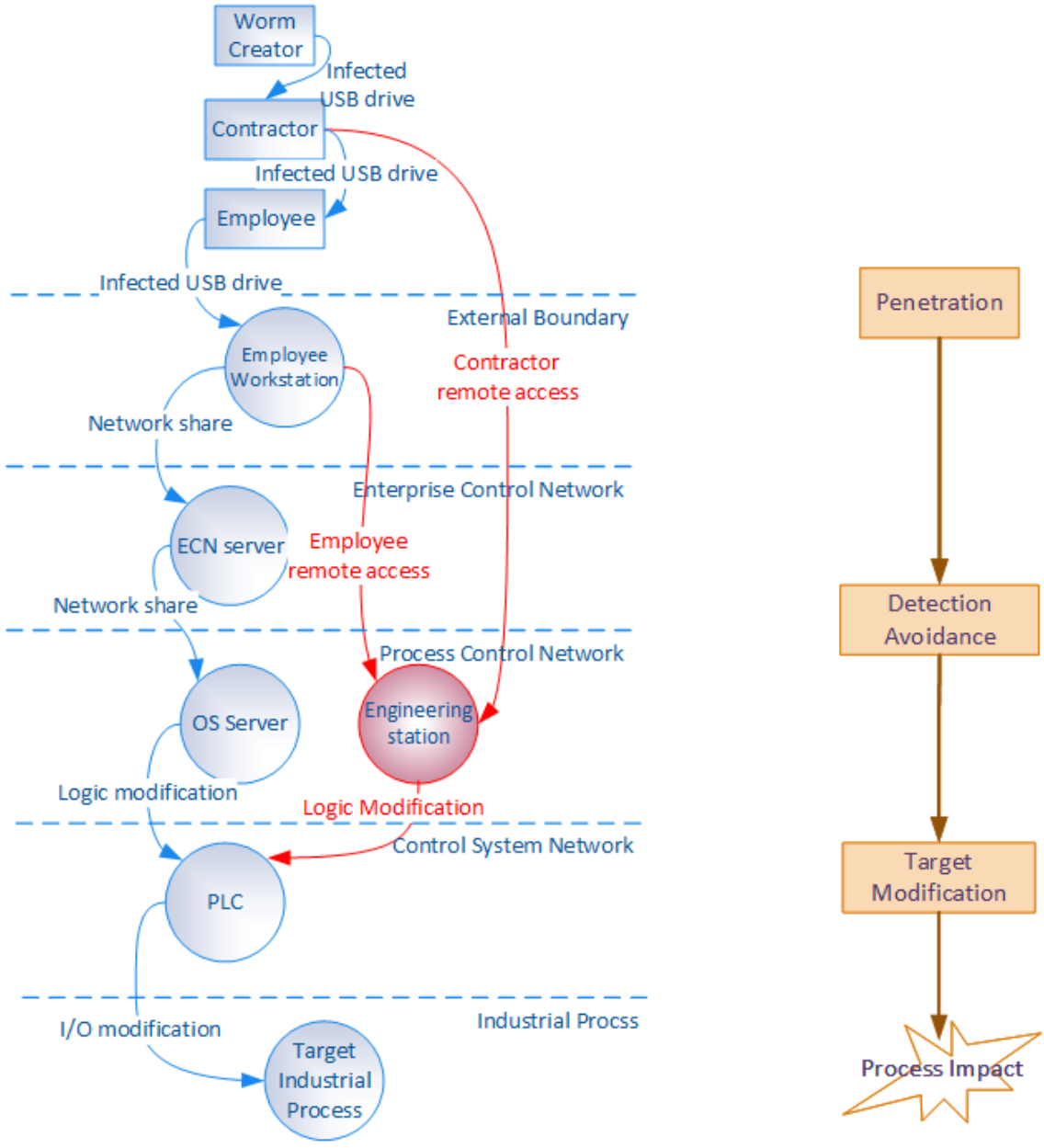


Figure 4.1: Stuxnet attack graph

This work mainly focuses on demonstrating the vulnerabilities of CPSes to cyber attacks with the goal of making CPS designers aware of security issues. All the modules are developed to have a generic nature that can be applied to different CPS settings. These modules focus on understanding and exploiting the vulnerabilities that affect these systems. Students are

exposed to exploitation techniques and tools that may be used against these systems. Similar to the Stuxnet attack flow, the modules incrementally gain access to the controller regulating a physical process. The flow of the modules is illustrated in Figure 4.2.

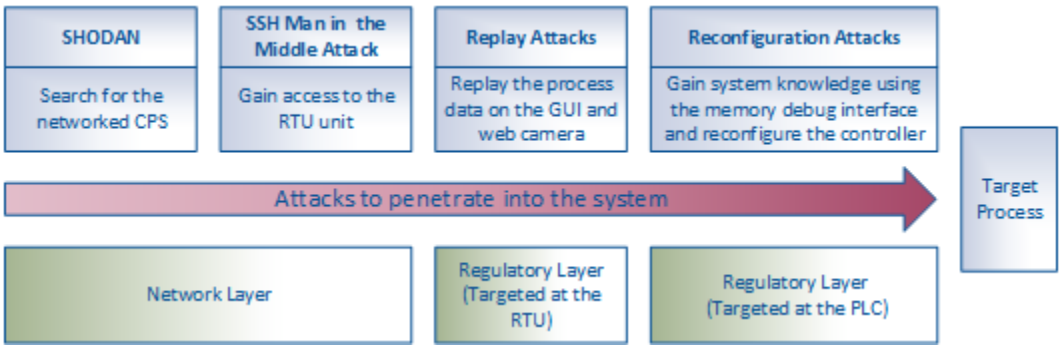


Figure 4.2: Overview of the modules

Our intent is to augment an existing course with hands-on experience on critical infrastructure and control systems security. Perhaps the greatest value of this modular approach lies in the potential to adopt modules as supplements to other courses on cyber security, embedded systems, or control systems. The key to advancement in this field lies in the recognition of the multi-disciplinary nature of CPS security. Table 4.1 provides a brief overview of the modules.

Table 4.1: Overview of the hands-on modules

Module Sections	SHODAN	SSH MITM attack	Replay attacks	Reconfiguration attacks
Learning Objectives	Search engine exploitation, vulnerability assessment	SSH1 vs SSH2, ARP poisoning	Packet capturing, video streaming	Memory corruption, exploiting the boot image
Tools Used	SHODAN, google hacking	PuTTY, Python, Cain and Abel, processing	MJPG streamer, wireshark, ettercap	Linux terminal
Tasks	• Using SHODAN search engine to find systems connected to the Internet. • Accessing the security vulnerabilities in the spotted unsecured devices. • Learning the importance of vulnerability assessment and creating awareness about CPS security.	• To use Address Resolution Protocol (ARP) poisoning to intercept packets transferred and execute MITM attacks. • To use network security testing software such as Cain and Abel to intercept Secure Shell (SSH) login credentials. • Understand the different versions of SSH and how SSH-2 is much more secure than SSH-1.	• To understand the basics of packet capture and finding the data inside them. • Using Python libraries (such as Socket) to write scripts to help execute the replay attack. • To perform a replay attack on a Raspberry Pi Camera Module by exploiting the video streaming service in Linux.	• To modify the reconfiguration channel to modify the boot image and gain system information using the memory debug channel.
Assessment	Demonstration of the solution and feedback from the hands-on exercises			

Chapter 5

Network Layer Modules

In a CPS, embedded controllers govern a physical process based on operator commands. Increasingly, the need for remote monitoring and control entails a networked CPS environment. A networked interface for the supervisory layer in a CPS is a means for effective and timely flow of information between plant supervisors and processes. Supervisory interaction with the RTU accessible via the Internet introduces a set of diverse security issues. Network integrity attacks exploit the IT network vulnerabilities that can lead to an adversary communicating with the Process Control System (PCS). We summarize the network integrity attacks as:

1. Man-in-the-Middle (MITM) attacks: In a MITM attack, the adversary can use known methods like ARP poisoning and intercepting network packets to view or modify sensitive data sent by the operator to the CPS component. Sending false messages may cause the operator to believe that the plant is functioning correctly in spite of an active attack. For example, an adversary can set up an access point with the same network name and MAC address as the original server. The victim mistakenly uses this network instead of the genuine one, which gives the adversary complete control over communication with the victim on this network.

2. Eavesdropping attacks: Eavesdropping refers to an attack where an adversary can intercept one or more communication channels in a CPS. Although a passive attack, CPS can be particularly susceptible to eavesdropping through network traffic analysis such as intercepting the data transferred by HMIs and commands sent by operators.
3. Denial of Service (DoS) attacks: In a DoS attack, the adversary can prevent legitimate requests for network resources from being processed by the system. DoS is a type of network attack that prevents the system from functioning normally [48]. These attacks initially target the communication interfaces to deactivate supervisory monitoring and interaction with the PCSes. Once the adversary penetrates the CPS, the adversary can tamper with various aspects of the system, such as:
    - Overloading the controller by sending a large number of commands.
    - Sending false sensor data, which may cause abnormal process behavior.
    - Block access to networked resources that can result in loss of access by the supervisor.

The network layer consists of the operator units that connect to the regulatory layer containing the PLCs. In this chapter, we discuss two attack modules that describe methods to penetrate the network layer to gain access to RTU. The modules are described as follows:

1. SHODAN Search Engine: This module describes SHODAN search engine that can be used to discover devices connected to the Internet. SHODAN can be used to find PCSes and SCADA systems and their open services. This information can be used to access the underlying systems by exploiting protocol vulnerabilities and inadequate security policies.
2. SSH Man-in-the-Middle (MITM) attack: This module uses a SSH downgrade connection to perform a MITM attack to sniff SSH credentials. This attack can be accomplished by downgrading the connection from a more secure protocol (SSH-2) to a less secure protocol (SSH-1) by deceiving the parties into thinking that one or more clients can only support the less secure protocol.

5.1 SHODAN Search Engine Lab

Module Objectives:

The students will learn:

1. How we can find many traffic lights, security cameras, home automation devices, and heating systems connected to the Internet with very few or no security safeguards.
2. The importance of having proper security in place to design a system with the least number of vulnerabilities.
3. How these tools can be used to avoid cyberattacks by finding these unsecured, connected devices and services using SHODAN, and alerting owners and operators about the vulnerabilities.

Module Overview:

This lab module consists of the following key components:

1. SHODAN search engine: Understanding basic operations of the SHODAN search engine and using the SHODAN website to find PLCs, routers, web cameras, and open services like telnet.
2. SHODAN Command-line Interface (CLI): Using the SHODAN CLI enabled in Python to get an overview of SHODAN commands to perform advanced search queries.

Module Introduction:

SHODAN is a search engine that lets you find specific computers (routers, servers, SCADA systems, etc.) using a variety of filters. Some have also described it as a public port scan directory or a search engine for banners. It is mainly a search engine for service banners in which metadata is sent from the server to client. SHODAN currently probes for 50+ ports.

Figure 5.1: SHODAN Search Engine

Typically, search engines crawl for data on web pages and then index the data for searching. In contrast, SHODAN interrogates ports and stores the resulting banners, then indexes the banners instead of the web content for searching.

Basic operations in SHODAN include:

Search: Search terms are entered into a text box. Quotation marks can narrow a search. Boolean operators + and − can be used to include and exclude query terms (+ is the implicit default).

Login: Create and login using a SHODAN account.

Login is not required, but country and netfilters are not available unless you login.

Export requires you to be logged in.

Filters: `country`: filters results by two letter country code

`hostname`: filters results by specified text in the hostname or domain

`net`: filter results by a specific IP range or subnet

`os`: search for specific operating systems

`port`: narrow the search for specific services

SHODAN can be used for penetration testing but requires some basic knowledge of banners including HTTP status codes. Banners advertise service and version.

Table 5.1: HTTP status codes

Status Code	**Description**
200 OK	Request succeeded
401 Unauthorized	Request requires authentication
403 Forbidden	Request is denied regardless of authentication

Finding Control Systems on the Internet

Many PLC vendors use a network topology similar to the testbed in the mock CPS. For instance, these PLCs use fixed UDP and TCP ports for Ethernet communications. Port forwarding is used in the router that connects the plant to the Internet to allow Internet access to these ports. A major PLC vendor uses port 9600 for their PLCs. The vendor claim that when a router forwards a TCP or UDP port to their PLC, the traffic is delivered to a non-Windows based operating systems making it impenetrable to standard hacking methods. Interestingly, a SHODAN search for these PLCs discovered at least 700 devices with port 9600 connected to a PLC. Figure 5.2 illustrates the search banner and network topology of the PLC.

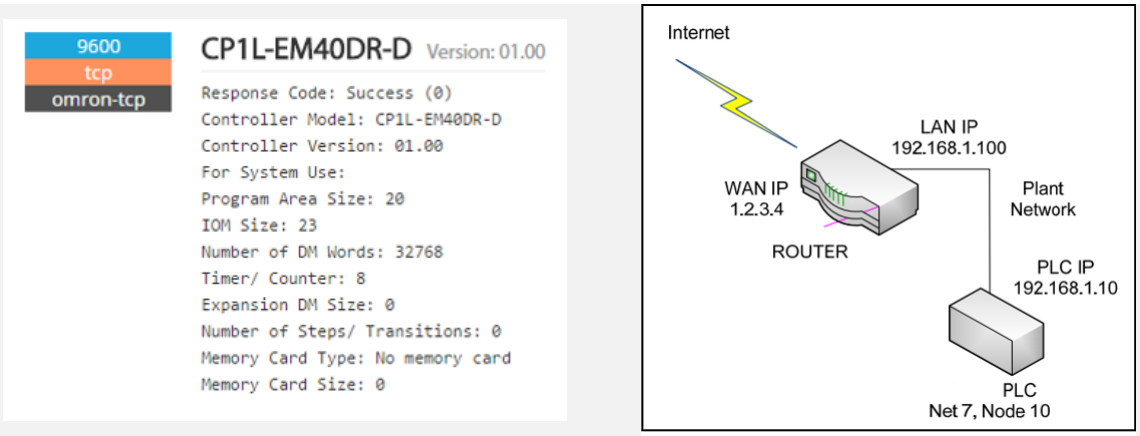


Figure 5.2: PLC connected to the Internet

A search for telnet connections resulted in the discovery of a open telnet port in an RS232 network adapter. Figure 5.3 shows the screenshot of the services and an active telnet session.

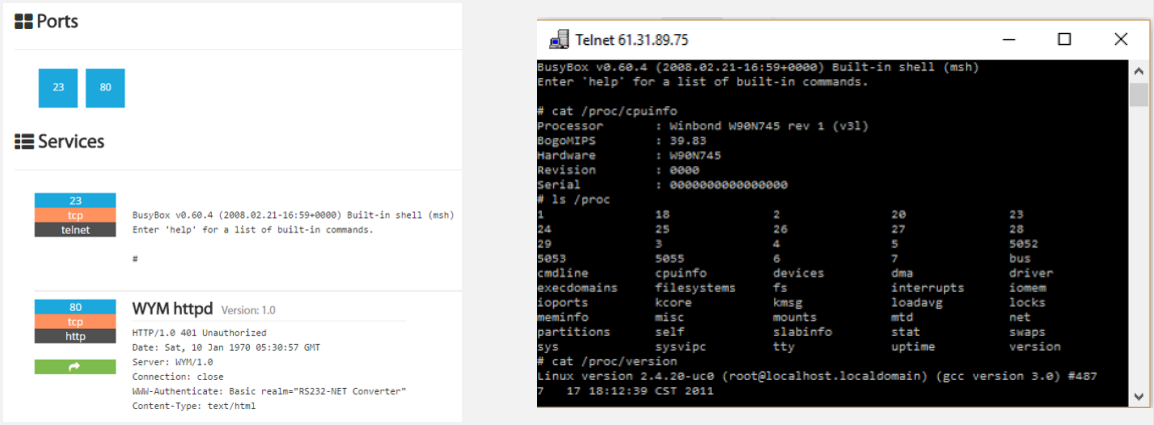


Figure 5.3: Open Telnet session

Module Tools:

1. SHODAN search engine for Internet-connected devices [49]. Accessible using the website (https://www.shodan.io/) as well as the command-line interface for SHODAN using Python.

Hands-on Exercise:

Using the website https://www.shodan.io/ for search:

Step 1: Use the SHODAN tool to find active Internet-connected devices for keywords like SCADA, RTU, web camera and other similar CPS keywords.

Step 2: Understand the banner codes to find the level of authentication and use of default passwords.

Step 3: Create a list of results from a targeted search to list the devices that do not require authentication, and have listed default passwords in their banner.

Figure 5.4 shows one of the open web camera discovered from a SHODAN search:

Figure 5.4: Open web cameras discovered using SHODAN

Using the SHODAN Python library for a command-line interface:

Step 1: The SHODAN CLI is packaged with the official Python library for SHODAN. To install the new tool simply execute:

```
easy_install shodan
```

Step 2: Once the tool is installed, initialize the environment with the API key using:

`shodan init YOUR_API_KEY`

You can get your API key once you create a Shodan account.

Step 3: SHODAN CLI can be used to execute several commands. To see the list of commands, run the following: `$ shodan`

Step 4: Try out different commands to explore the interface.

host: See information about the host such as where it's located, what ports are open and which organization owns the IP:

```
$ shodan host xx.xx.xx.xx
```

myip: Check your Internet-facing IP address:

```
$ shodan myip
```

download: Search SHODAN and download results in a file, where each line is a banner serialized in JavaScript Object Notation (JSON) by using:

```
$ shodan download example-file-name search-query
```

parse: Use parse to analyze a file that was generated using the download command. You can filter out the fields that are required and convert the file to a CSV format:

```
$ shodan parse --fields ip_str,port,org --separator,
example-file-name.json.gz
```

5.2 Man-in-the-Middle Attack Lab

Module Objectives:

The students will learn:

1. How ARP poisoning can be used to intercept packets transferred and execute a MITM attack.
2. How to use network security testing software such as Cain and Abel to intercept SSH login credentials.

3. The different versions of SSH and how SSH-2 is much more secure than SSH-1.

Module Overview:

This lab module consists of the following key components:

1. SSH protocol: Understanding the difference between SSH-1 and SSH-2.
2. ARP poisoning: Using ARP poisoning to create a MITM scenario and exploit the weak password authentication method in SSH-1.

Module Introduction:

While the SHODAN search engine shows us numerous devices connected on the Internet, it also highlights vulnerabilities in some of these systems due to the lack of security measures. In this module, a security vulnerability in the SSH protocol is exploited to sniff the SSH login credentials. Secure Shell (SSH) is a cryptographic network protocol that allows remote login and other network services to operate securely over an unsecured network [50]. In this module, ARP poisoning is used to sniff SSH passwords.

ARP poisoning is an attack that can be used as the precursor to many other network attacks. ARP poisoning works by sending false MAC address information to two hosts on a network so that both hosts believe that the attacking user is the other host. The reason this works so simply is because network switches are designed to send traffic based on a host's hardware address, the MAC address, and not the IP address. Without having to spoof an IP address or a MAC address, all we have to do is replace the victim's idea of the server's MAC address with our MAC address and replace the server's idea of the victim's MAC address with our MAC address as well. This is called a man-in-the-middle attack. If we become the man-in-the-middle to all network traffic, we can see every packet sent between both hosts. While this attack does not require being near

the plant, this attack can only be used if the attacker and victim are on the same local network. Figure 5.5 shows how the victim host and the webserver can both be ARP poisoned to send all network traffic through the attacking computer.

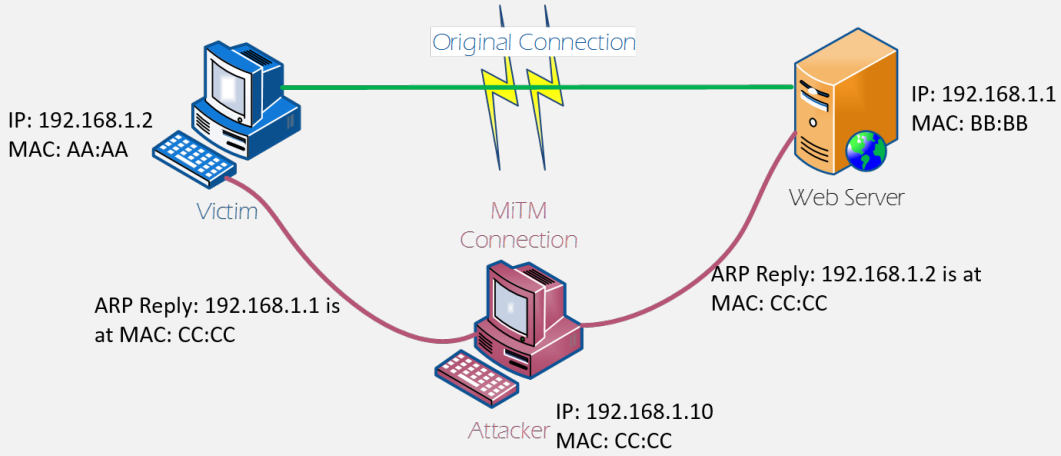


Figure 5.5: Man in the Middle Attack

Cain and Abel [51] is a network security software suite that can ARP poison the victim and the gateway to intercept SSH login credentials of Raspberry the Pi's SSH server.

Figure 5.6 shows a normal SSH connection, whereas Figure 5.7 illustrates the man-in-the-middle connection.

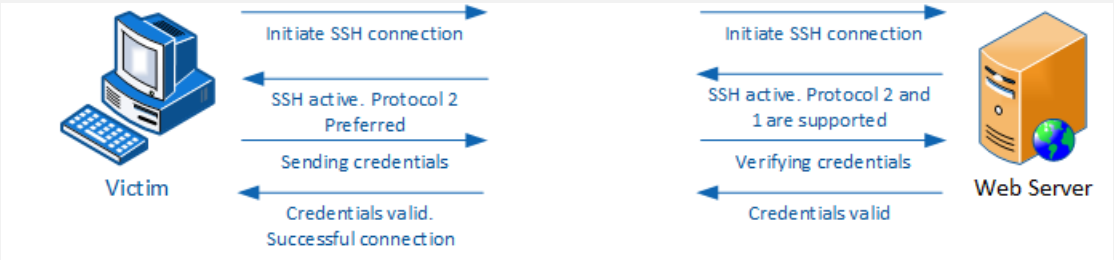


Figure 5.6: Normal SSH connection

An adversary modifies the messages between the client and server in order to force the victim to connect using a less secure protocol (SSH-1) that uses a weak password

authentication method.

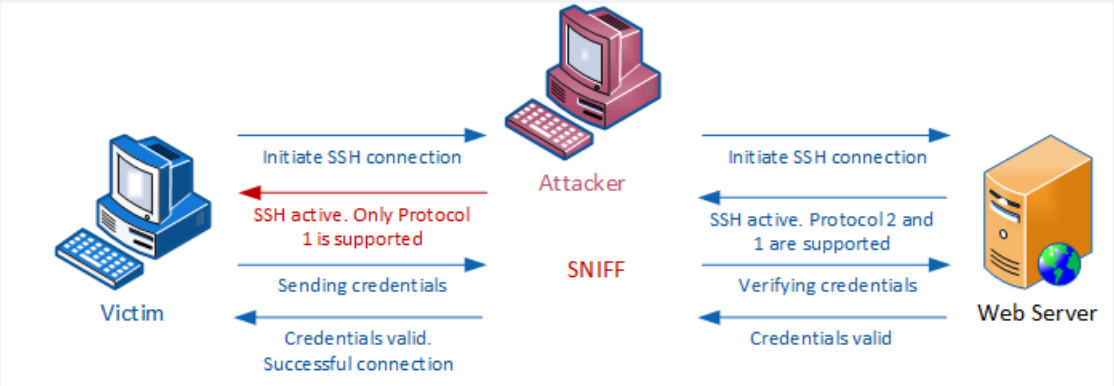


Figure 5.7: SSH connection under MITM attack

Module Tools:

1. Cain and Abel: This software will sniff and filter the packets send from the victim to the server [51].
2. PuTTY: This software creates a terminal on the Raspberry Pi.

Module Assumptions:

This module assumes that the attacker is on the same local network as the victim who is logging into the SSH server. The server can be on any other network on the Internet, and this attack will still work properly. It is also assumed that both the victim and server have SSH-1 enabled.

Hands-on Exercise:

Step 1: **Install Cain and Abel and download PuTTY.**

PuTTY is a terminal application used to log into the SSH server on the Raspberry Pi. Cain and Abel might be marked as malware while downloading; add

it to any exception lists, as it will not cause any harm to your computer.

Step 2: **Ensure that the Raspberry Pi has both SSH-1 and SSH-2 enabled.** Cain and Abel works by using an SSH downgrade attack. This is because SSH-2 is much more secure, and not easily crackable by simply being the host in between a transmission. A SHODAN search reveals that many of the CPS systems tend to have SSH-1 enabled, which means that communication can be established to these systems from hosts that use SSH-1. In order to exploit this weakness, the adversary can pose as a host trying to connect to the server, but can only support SSH-1. When the adversary allows the SSH traffic from the victim to the server, the SSH-1 protocol is used and the login credentials can be decrypted. Thus, by enabling both SSH-1 and SSH-2 on the server, the vulnerability of SSH-1 can be demonstrated and SSH-2 can still be used when needed.

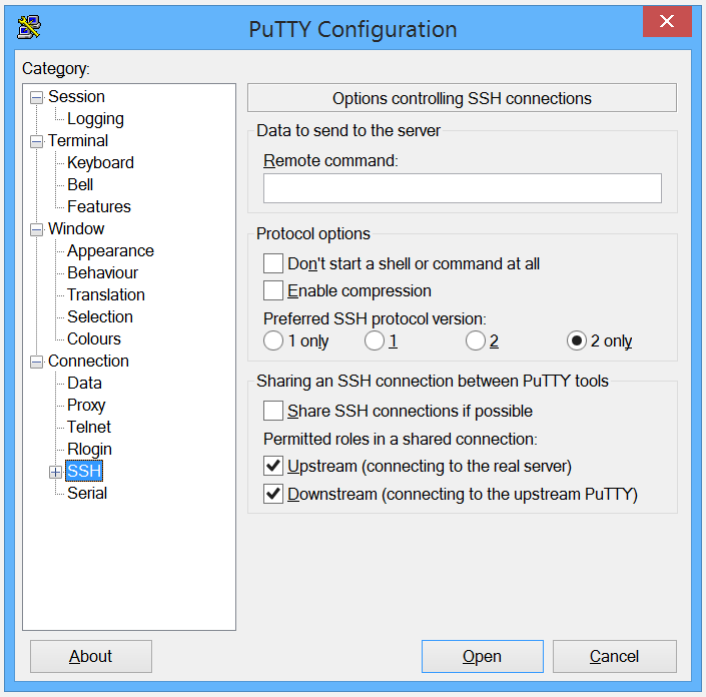


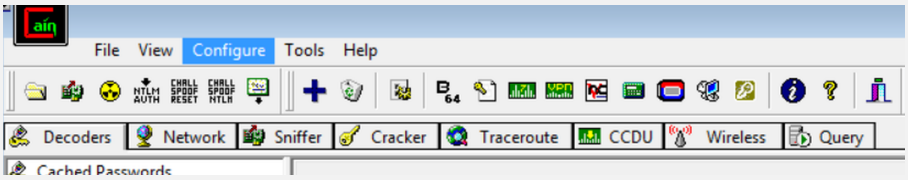
Figure 5.8: Putty Configuration

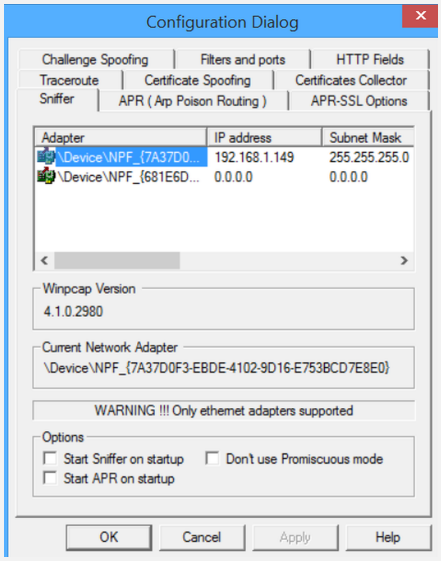
To check if SSH-1 is enabled, open the file `/etc/ssh/sshd_config` and find the line containing `"Protocol"`. If it says `"Protocol 2, 1"` or `"Protocol 1, 2"`, then both versions are enabled. For a new installation of OpenSSH, as a security measure SSH-1 is disabled by default. If SSH-1 is not enabled, simply add `", 1"` to the end of that particular line in the file, and add `HostKey /etc/ssh/ssh_host_key` among the other `HostKey` lines in the file. Save and close the file. Then generate the host key with the following command: `ssh-keygen -f /etc/ssh/ssh_host_key -N '' -t rsa1`.

Finally, restart the SSH server to ensure SSH-1 and SSH-2 are properly enabled. To verify, connect to the Raspberry Pi using PuTTY by selecting `"Only 2"` and `"Only 1"` before logging in as shown in the screenshot above.

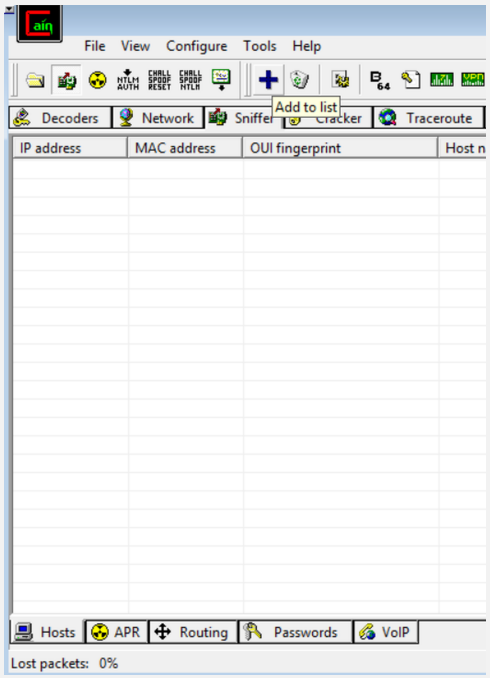
Step 3: **Start Cain and Abel and discover the hosts on your network.**

Run Cain and Abel and configure it to use the primary network adapter.

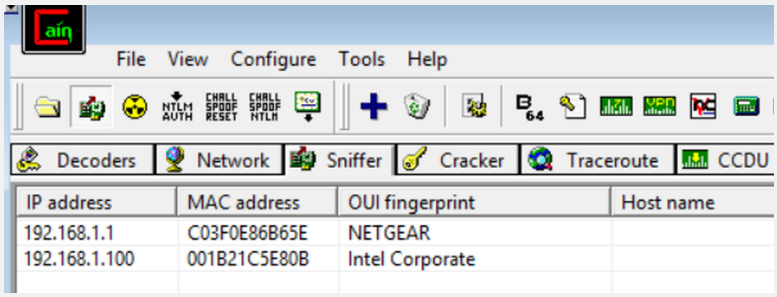




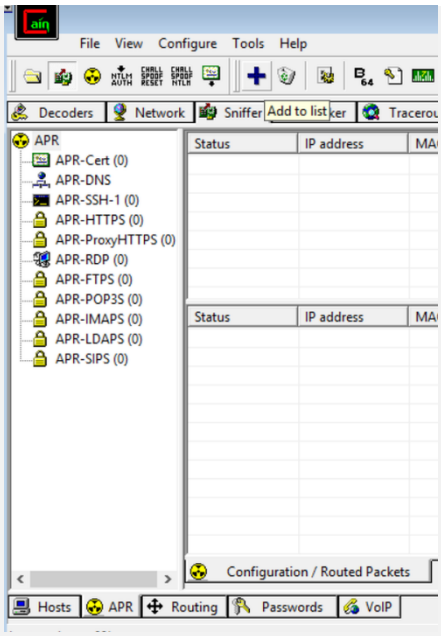
Start the sniffer and scan for hosts on the network. Make sure the hosts tab at the bottom left is open, click `add to list` and then OK as shown in the following screenshot.



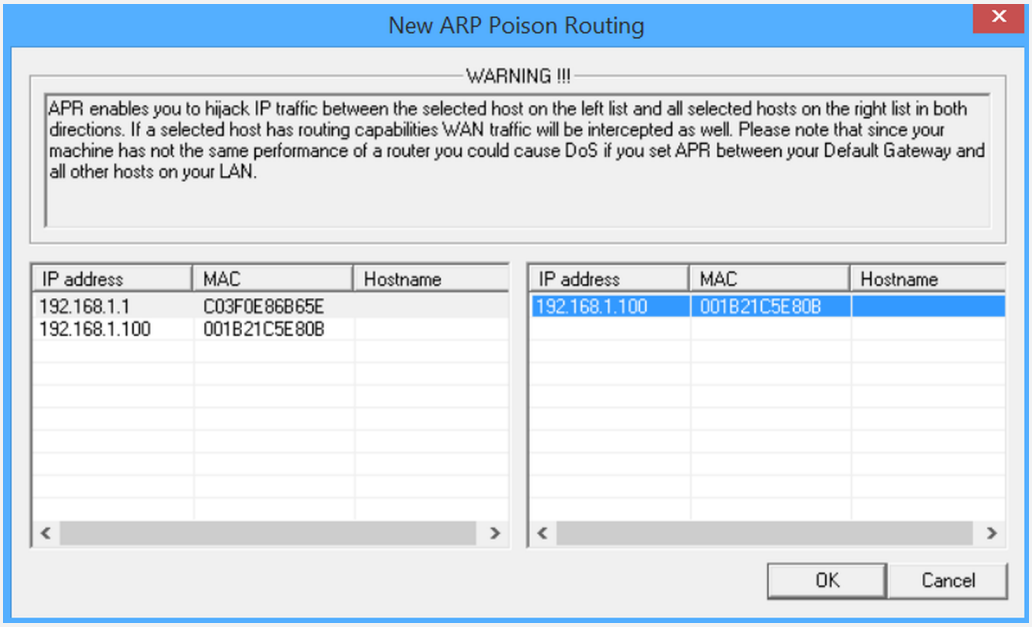
This shows the other devices on the network.



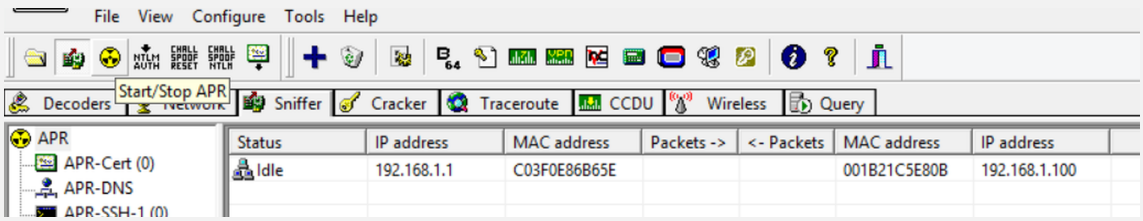
Step 4: **ARP poison your victim.** Change to the APR tab at the bottom left and click add to list.



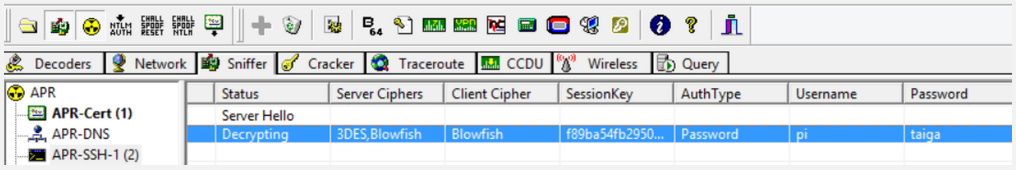
Choose the gateway and the victim. On this network it's quite simple. The gateway is the `.1` address and the other host is the `.100` address.



Once the victim and the gateway have been selected, start ARP poisoning and wait for the victim to enter their login credentials to the SSH server.



After the victim enters their credentials on the terminal, the credentials will be available in Cain and Abel. While still in the sniffer tab at the top and the APR tab at the bottom, you can select to view only SSH traffic that Cain and Abel has sniffed. Within this list, the login credentials are located in the rightmost columns.



There may be more than one entry in the SSH list in Cain and Abel, but if a user logged in successfully, their credentials will be available in at least one of the entries as shown in the screenshot above.

5.3 Summary

In this chaper, different network integrity attacks such as DoS, MITM, and Eavesdropping are discussed. This chapter explores the SHODAN search engine to exploit the open vulnerabilities in the devices discoverable on the Internet. Also, it explores the different versions of SSH protocol and illustrates how an adversary can use ARP poisoning combined with a protocol downgrade attack to sniff SSH credentials.

Chapter 6

Regulatory Layer Modules

In this chapter, we discuss the supervisory components of a CPS and their security vulnerabilities. A supervisory system consists of a HMI system with read-only or supervisory access control. These workstations can reside in a variety of locations throughout the industrial networks, as well as the business networks, up to and including Internet-facing web portals.

A supervisory workstation collects information from sensors used within a control system connected via a supervisory channel and presents that information for supervisory purposes. HMIs allow operators to start and stop process cycles, adjust set-points, and perform other functions required to adjust and interact with a control process.

When a supervisory system monitors a control system remotely, the connection between the workstation and the underlying CPS supervisory components must be carefully established, controlled, and monitored. Otherwise, the overall security of a control system network could be weakened because the supervisory system becomes an open attack vector to the CPS. For example, by placing a supervisory console in the business network, the console can be more easily accessed by an attacker and then utilized to communicate back to the CPS. If remote supervision can be provided via read-only data, a one-way communication path or some form of secure data replication should be used to prevent such an inbound attack.

Although securing the channel is one of the important aspects of ensuring resilience, attackers can exploit the physical process by exploiting this channel using valid set-points but issued in a manner that can damage the physical process. An active adversary may attempt to manipulate message routing and network coordination by injecting messages, recording and replaying old ones, or exploiting protocol interactions. A combination of these attacks can be used as a platform for a more stealthy attack if the adversary gains access to reconfiguration resources.

In our system, the supervisory HMI component is made available to the client using the RTU in the regulatory layer. The regulatory layer consists of two active components, namely the RTU and the PLC. An RTU interacts with the PLC on a simple telemetry channel to relay messages on the IT network. After penetrating into the network layer as illustrated in the previous chapter, we have access to the RTU. The RTU hosts a web server to transmit real-time process data to the supervisory HMI, and to stream live camera feed to anyone who requests it on the Internet. The web server is also used to send set-point commands to the PLC through the supervisory channel. Processing GUI is the client software used as a supervisory HMI to display real-time pendulum state and also to control the position of the pendulum in the RIP application. In this chapter, we discuss various attacks that can disrupt the functioning of the system by manipulating the processes managed by the RTU:

1. Replay attack on Processing GUI: This attack targets the client HMI interface, namely the Processing GUI, that receives real-time process information and displays it on all the clients.
2. Replay attack on the web camera: This attack targets the remote surveillance component of the system provided by the web camera. In this attack, the adversary attempts to transmit old camera feed to the client.
3. Supervisory set-point violation attack: This attack exploits the physical process by executing a combination of legitimate set-points through the supervisory channel that aims to cause a dangerous transient. By commanding the pendulum to slew rapidly back and forth it may be possible to cause the pendulum angle to become so large that

control is lost and the pendulum falls.

6.1 Replay Attack on Processing GUI Lab

Module Objectives:

The students will learn:

1. The basics of packet capture and extracting the data inside them.
2. Using Python libraries (such as Socket) to write scripts to help execute a replay attack.

Module Overview:

This lab module consists of the following key components:

1. RTU webserver manipulation: Using the RTU access (achieved as a part of the MITM lab) to capture and modify data sent to the Processing GUI.
2. Scripting the replay attack: Use tools such as Wireshark [52] for packet capturing and Python Socket library to replay the modified packets.

Module Introduction

A replay attack can be thought of as an elaborate postcard in front of a camera in a spy movie. This attack is a form of network attack where an adversary intercepts valid data and maliciously retransmits it. As the attack is launched on the system, the replay data gives the impression to the supervisors that the system is actually operating normally.

In the mock CPS system, Processing GUI is the client HMI used to read the pendulum

data broadcasted by the RTU. Processing GUI uses a Node.js script to send the data from the pendulum process to the client running the Processing application. To perform a replay attack on the Processing GUI, packets received by this application should be recorded under normal operation of the pendulum and replayed at a later time. This can be acheived by using Wireshark [52] to record the packets received by the GUI and replayed to the client. The Node.js script running on the RTU should be aborted to ensure the client receives only the replayed data. Figure 6.1 shows the Processing GUI. The black line represents the angle of the servo and the blue line represents the angle of the inverted pendulum.

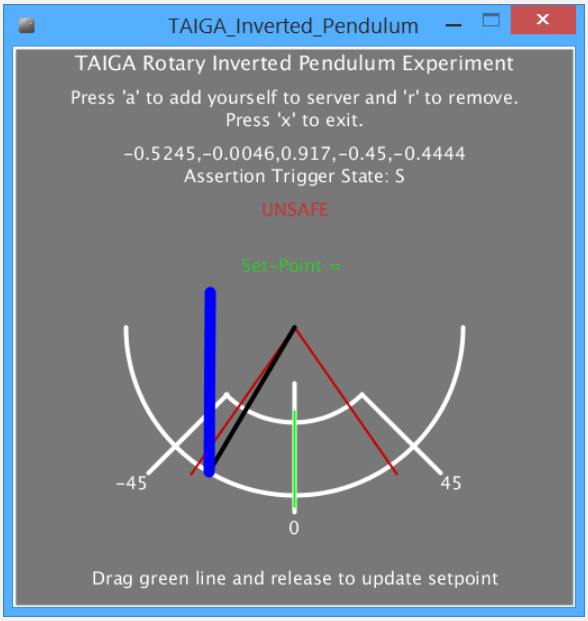


Figure 6.1: Processing GUI

Module Tools:

1. Wireshark: Software used to sniff and save packets sent to our computer [52].
2. Online text filterer: This separates the data from the packets exported by Wireshark.

3. Processing: Processing runs the GUI that monitors the pendulum movement [41].
4. Python: Python is used to send the packets back to execute the replay attack [53].
5. PuTTY: This is the software emulating a terminal on the Raspberry Pi.

Module Assumptions:

This module assumes that the student has the means to send the replay data packets to the victim. This can happen in a number of ways. The first is more elaborate and will not be demonstrated in this module since it involves the victim being on a different local network than the adversary and behind a router. In this case, the attack would work if the victim has network traffic running on the designated port forwarded to their computer. The second possibility is the attacker and victim on the same local network, which is demonstrated in this module. This module also assumes access to the RTU (Raspberry Pi).

Hands-on Exercise:

Step 1: **Start the Node.js script and the Processing GUI.**

Enter the Raspberry Pi terminal using PuTTY and enter the following command in the home directory:

```
node RPI-Remote-Terminal-Unit/TAIGA-RTU_udp-server.js &
```

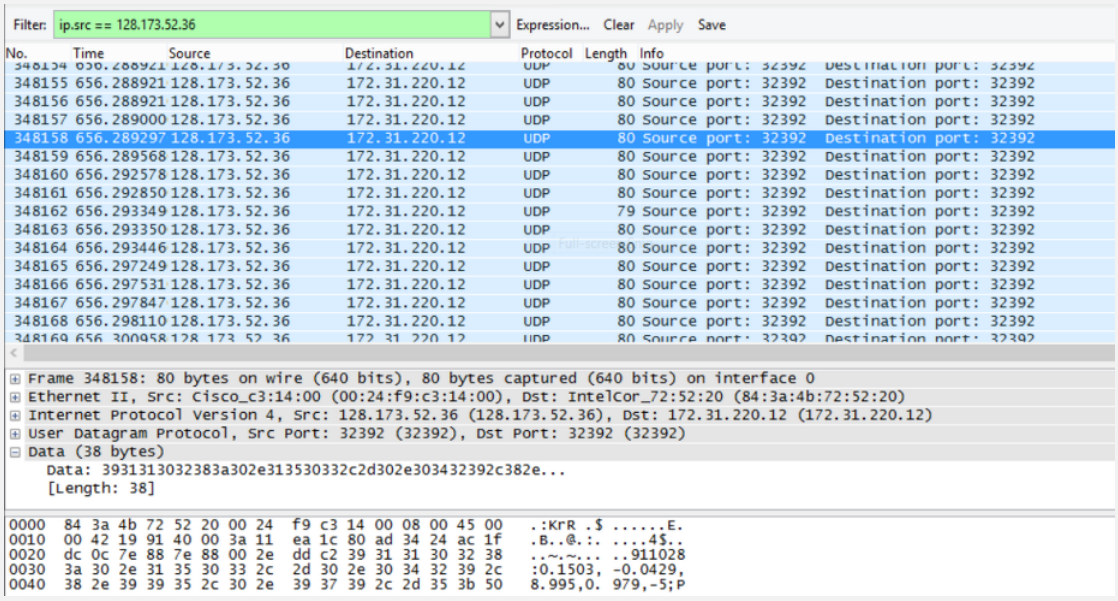
This starts the Node.js script that sends the data from the pendulum to the Processing GUI.

By running the Processing GUI, the real-time pendulum motion can be seen on the screen as the packets sent from the Raspberry Pi are being received. These packets contain the data played back during the attack.

Step 2: **Capture data with Wireshark.**

Open Wireshark and start sniffing on the active network adapter. Since infor-

mation is polled every millisecond for the GUI, the list is populated immediately.

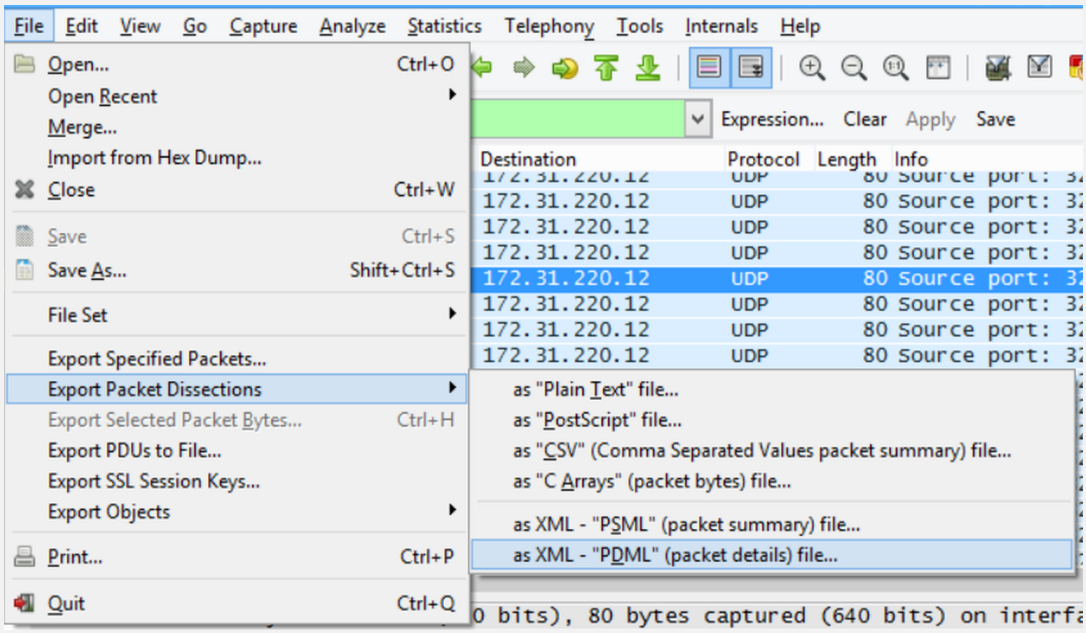


As shown in the above screenshot, multiple packets are sniffed in order to perform the replay attack. By sorting according to source Internet Protocol (IP) address (in this case, the IP address of the Raspberry Pi is 128.173.52.36), a good 5000 or so packets are sectioned off to give 5 seconds of data for the replay attack. Observe the data in the packet. The packet data (sent by the Raspberry Pi) contains 5 fields. The first field is the angle of the servo in radians, and the second field is the angle of the pendulum in radians. Using this information from the packet data, the state of pendulum can be determined at the time of recording the packet. Always make sure that the packets exported for the replay attack contain the data of the desired pendulum motion, i.e. pendulum oscillating at the desired set-point. It is important to verify the data in the captured packets before writing the replay scripts to ensure the replayed data is not representing an idle or a failing system.

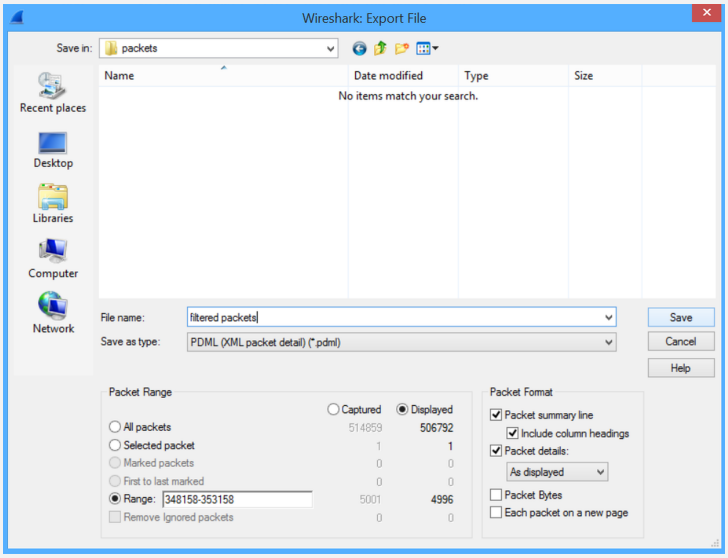
Step 3: **Export the data from Wireshark and separate the data.**

Once the packets for the replay attack are selected, they can exported to separate

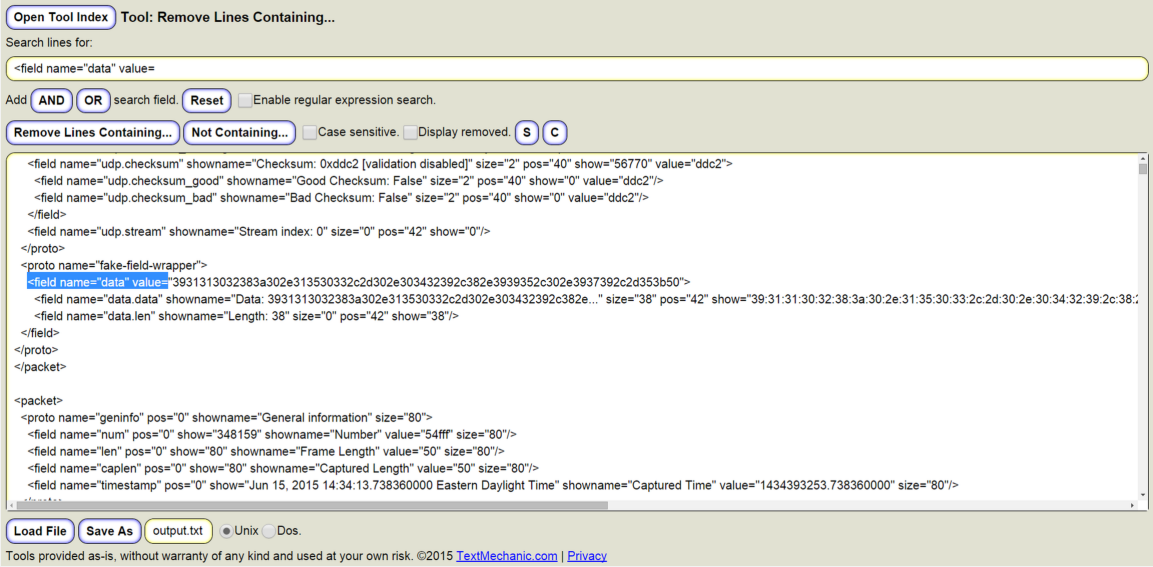
the data from them. Within Wireshark, there is an option to export packet details as shown in the screenshot below.



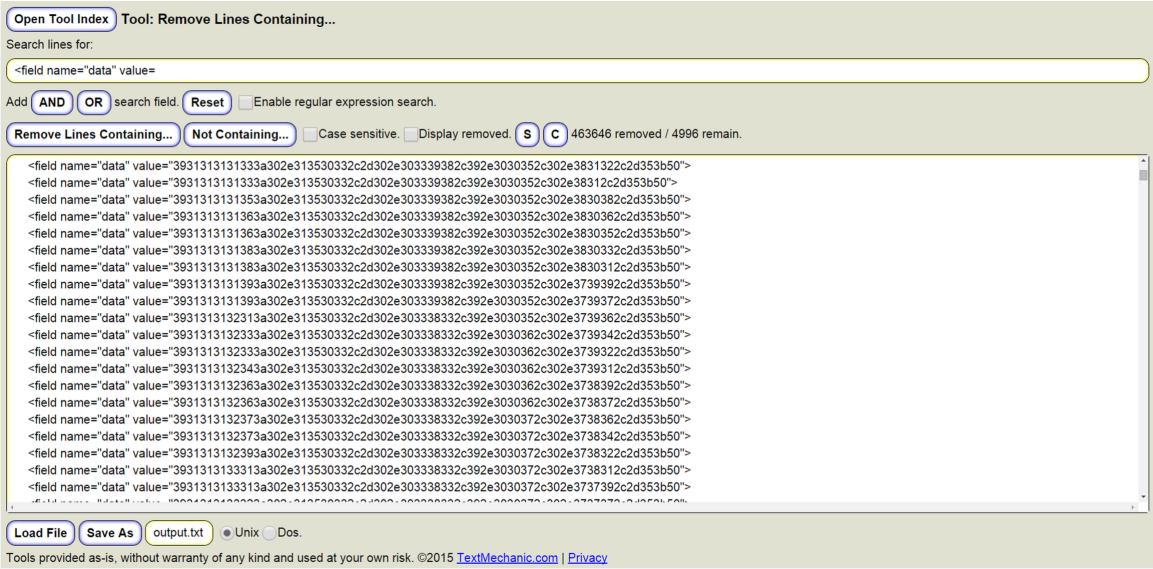
Save the output file for future reference. Also make sure to only export about 5000 packets. The oscillation of normal operation has a very small window of only a few seconds.



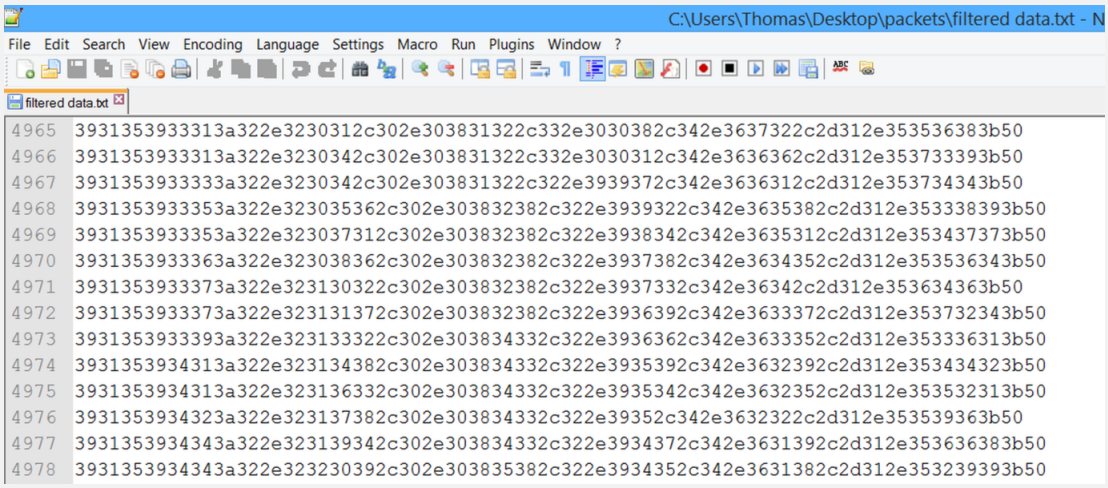
If you open the file as plain text, you will notice it contains XML formatting. To remove all lines within the file except those containing the data frame, go to the online text filterer linked in the tools section of this module. Load the file that was exported from Wireshark, and locate a line containing the data in hex. Copy the first part of the line, `<field name=''data'' value=`, and paste it into the search bar as shown in the screenshot below. Remove all lines not containing this phrase so that the file contains one packet's data per line.



Remove all lines except the lines containing the data from the packets as shown below.



After removing extra lines, copy this text and place it into a text editor, and remove the text before and after the hex values of each line's data. The data is as shown in the screenshot below. Also, check that the data is at least a few thousand lines.



Step 4: **Create a Python script.**

The text file obtained after filtering contains the data of the 5000 or so packets, which are replayed to the victim. To have a successful replay attack, the actual

data sent from the Raspberry Pi should be stopped. This can be accomplished by stopping the Node. js script that was started. Once the victim stops receiving the legitimate data, the recorded data can be sent. The Node.js script sends data in the form of UDP packets. These data packets can be easily formulated using the Socket library in Python. Import the text file containing the replay data into Python. Once the list is imported, use Socket methods to read through this list, create packets, and send them to the victim. Python's official Socket documentation identifies the best methods to be used.

6.2 Replay Attack on the Web Camera Lab

Module Objectives:

The students will learn:

1. Using Wireshark to sniff packets of web camera live data.
2. Using MJPG-Streamer to modify video streaming software to execute the replay attack on the web camera.

Module Overview:

This lab consists of the following key components:

1. Capturing images of the working system using access to the RTU.
2. RTU video-streamer modification: Modifying the streamer application on the RTU to stream new images to perform a replay attack.

Module Introduction:

A replay attack is defined as an attack that occurs when an adversary records a stream of messages between two nodes in a network and replays the stream to one or more victims [54].

In our mock testbed, a replay attack on the web camera implies that the image stream obtained from the RTU is manipulated by the adversary and played to the client requesting it. In our system, the remote surveillance system is implemented using a Raspberry Pi camera module and a MJPG-streamer implemented in the Raspberry Pi (RTU). The camera captures images of the pendulum system in motion at the rate of 15 frames per second, and the MJPG-streamer is configured to broadcast this stream on port 8081. The port is then forwarded to the public IP using the network router accessible remotely via the Internet on 128.173.52.36:8081 using any web browser. Figure 6.2 illustrates the setup of the web camera used in this module.

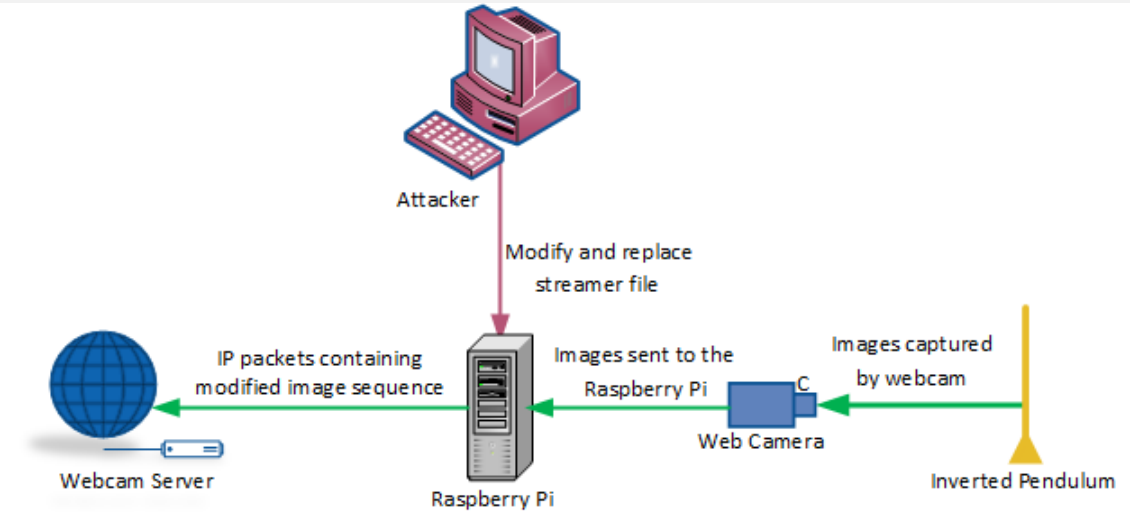


Figure 6.2: Overview of replay attack on the web camera

For this module, the MJPG-streamer code needs to be modified to allow streaming of pre-selected image sequence as opposed to real-time captured images.

Module Tools:

1. MJPG-Streamer: MJPG-streamer takes JPGs from Linux-UVC compatible webcams, or other input plug-ins to streams them as M-JPEG via HTTP to web browsers [55].
2. Wireshark: Software used to sniff and save packets sent to our computer [52].
3. Ettercap: Ettercap is a comprehensive suite used for man-in-the-middle attacks. It features sniffing of live connections, content filtering on the fly, and many other useful capabilities. Ettercap supports active and passive dissection of many protocols and includes many features for network and host analysis [56].

Module Assumptions:

For this module, it is assumed that the student already has access to the RTU unit. In case the SSH MITM module was not completed, the Raspberry Pi login credentials are: username: `pi` and password: `taiga`. It is also assumed that students who perform this reply attack are on the same subnetwork as the RTU.

Hands-on Exercise:

Step 1: **Capture images using Wireshark.**

Open Wireshark and start sniffing on the active network adapter. This step is similar to the data capture done in the previous lab. Alternatively, Ettercap can also be used to capture the images.

By sorting according to source IP address (in this case, the IP address of the Raspberry Pi is 128.173.52.36), and the port number 8081, collect consecutive images to form a 3-4 second stream. 10-15 images give enough data to form a stream of the pendulum system in its desired motion.

Step 2: **Understand how MJPG-Streamer works.**

In order to perform this attack, it is imperative to understand the functioning of the MJPG-streamer used for streaming live images. The script `start.sh` runs the streamer located under the following directory:

`/home/pi/experimentstreamer/mjpg-streamer-master/mjpg-streamer-experimental`.

The `README` file offers useful functions of the MJPG-streamer that will be beneficial to perform the replay attack.

The MJPG-streamer supports different input and output plug-ins to cater for different input and output options. Discover the three input and output plug-ins supported by this streamer.

Step 3: **Explore the right plug-in suitable for a replay attack in the MJPG-streamer.**

MJPG-Streamer uses a variety of plug-ins to stream videos. For this part of the exercise, the students' job is to look at the three different plug-ins: `input_raspicam, input_uvc` and `input_testpicture`.

The `input_testpicture` supports non-continuous streaming and can be used for a replay attack by programming this plug-in to stream the images captured for the replay attack.

Step 4: **Modify the plug-in to perform replay attack.**

For this step, students should look at this plug-in's `Makefile` and `input_testpicture.c` to understand the structure of the code. Students should use previously captured images while modifying the code to complete the replay attack. The following command is used to run the new plug-in:

```
export LD_LIBRARY_PATH=.
./mjpg_streamer -o ''output_http.so -w ./www''
-i ''input_testpicture.so''
```

6.3 Supervisory Set-point Violation Attack Lab

Module Objectives:

The students will learn:

1. How to exploit the supervisory channel to disrupt the pendulum process.
2. Using Python libraries (such as Socket) to write scripts to help execute the set-point violation attack.

Module Overview:

This lab consists of the following key components:

1. Pendulum process: Understanding the pendulum process and supervisory commands used to control the process.
2. Python script: Using Python to develop a script to issue set-point commands to sabotage the pendulum process.

Module Introduction:

In this module, we discuss the supervisory components of a CPS and their security vulnerabilities. Although securing the channel is important, the attacker can exploit the physical process by using a combination of legitimate set-points, but executed in a custom manner that can damage the physical process.

Set-point commands are used to establish the angle of the rotary arm. Our system uses a Node.js script that broadcasts the data intended to be read by a Processing sketch (supervisory system). Set-point commands can be sent using the Processing GUI to change the rotary arm angle. Node.js is written in JavaScript and sends the data from the pendulum to our Processing GUI. Processing is an open-source project to animate a

sketch that will display all of the information sent by the Node.js script. By launching the sketch in Processing and watching the system operate normally, Wireshark can be used to record the packets received.

In order to disrupt the pendulum process using legitimate set-points, the pendulum process should be observed. Swinging the pendulum near the operational guards, or trying to continuously swerve from one end to the other can cause the pendulum to fall over. In this lab, we develop a Python script to send different combinations of desired set-points to attack the system.

Module Tools:

1. Wireshark: Software used to sniff and save packets sent to our computer [52].
2. Online text filterer: This separates the data from the packets exported by Wireshark.
3. Processing: Processing runs the GUI that monitors the pendulum movement [41].
4. Python: Python is used to send the packets back to execute the replay attack [53].
5. PuTTY: This is the software emulating a terminal on the Raspberry Pi.

Module Assumptions:

This module assumes a basic understanding of the physical process.

Hands-on Exercise:

Steps 1 : **Repeat steps 1 to 3 from Replay Attack on Processing GUI module.** This involves starting the Processing GUI and capturing data using Wireshark. Explore and separate the data.

Step 2 : **Create a Python script.**

From the collected data, infer the command for setting up the set-point. Once that is known, develop a Python script to send different combinations of desired set-points to attack the system.

Hint: The format of the data in the captured packet is "5350xx", where xx denotes the angle of the pendulum in degrees.

For example, after initializing the socket for UDP communication, the following code can be used to send a particular set-point command to the controller:

```
PACKETDATA = "535010".decode('hex')
s.send(PACKETDATA)
```

6.4 Summary

This chapter discussed the supervisory components of a CPS. A supervisory system monitors a control system remotely. All the supervisory components must be carefully controlled and monitored in order to maintain a secure network. The overall security of a CPS is at risk if the supervisory system is open to attack. This chapter describes how an adversary can manipulate the message routing and network coordination by recording and replaying old data in the replay attack modules, and how legitimate set-points can be used against the system to exploit the physical process. Thus, a combination of these methods allows a stealthy attack on the physical process that is not detected by the supervisor.

Chapter 7

Reconfiguration Layer Modules

In the previous chapter, different attack modules were discussed that explored the weaknesses in a RTU and attacks that exploit the supervisory system. These attacks can be combined with reconfiguration attacks on the controller that are more stealthy in nature. At times, exploitation of CPSes requires adaptive behaviour in the presence of faults [57]. Reconfiguration attacks can be executed by reading values of variables, executing functions and procedures, and performing input/output. All of these operations are performed in modern microcontrollers by reading/writing specific memory addresses. For example, remote diagnostic ports can be exploited allowing adversaries to access the same debugging tools as the system developers. An adversary can then read symbol tables, read/write memory, and cause the processor to start execution from a certain address. Most modern CPSes provide a remote diagnostics channel that is responsible for controller debug and update. On getting access to this channel, an adversary can execute many attacks while maintaining a certain level of stealthiness, for example by executing a remote memory dump or reading/writing critical variables.

This chapter presents two modules that exploit the reconfiguration/remote diagnostics channel:

1. Memory debug channel: An adversary can exploit the memory debug channel realized through the remote diagnostics channel to understand the controller algorithm and system organization.
2. Boot image update channel: An adversary can modify the boot image used by the system.

The memory debug interface as well as the boot image update channel are implemented as simple tasks running on Core 0 that communicate over the UART channel with the RTU. The details of our implementation are available in the modules. Also, these two modules rely on students' understanding of the detailed nature of the PLC platform (Zynq SoC). Additional information about the Zynq SoC can be found in the Zynq reference manual [58].

7.1 Memory Debug Channel Lab

Module Objectives:

The students will learn about:

1. Exploiting the remote diagnostics channel to access the memory debug channel.
2. Reading and writing controller memory to gain system knowledge.

Module Overview:

This lab consists of the following key components:

1. Memory read/write: Understanding read/write access to On-Chip Memory (OCM) of the processor system.
2. Channel access: Sending the memory read/write commands through the memory debug channel.

Module Introduction:

Memory is not only used to store software in an embedded system; memory may also serve as the interface between the embedded processor system and some custom hardware in a Field Programmable Gate Array (FPGA). Memory debuggers may offer mechanisms to observe memory contents and execution patterns. Zynq's OCM is a dual-ported BlockRAM memory. External (e.g. DDR) memory works in much the same way, the only difference being that the memory controller interprets the request from the processor and generates an external memory transaction.

On-Chip Debugging (OCD) is a way to run program on the target chip. In our application, a debug interface is implemented using the remote diagnostics channel for remote debugging of the controller. Debug access provides:

1. Read/write access to registers.
2. Read/write access to memories while the program is executing.
3. Start/stop program execution.

This debug access can be used to monitor the program under execution. For an adversary, access to the memory debug channel can give insights into the controller algorithm and system organization. An adversary can extract memory patterns and match them against a database of known patterns from open source software. This module uses the read/ write access to OCM to understand the channel usage.

Memory access functions are provided by the Library Generator in Zynq. There are various functions designed specifically for reading from and writing to memory, in various byte widths. These functions include:
`Xil_In8, Xil_In16, Xil_In32, Xil_Out8, Xil_Out16, Xil_Out32.`

Contemporary microprocessors access memory in units of bytes (8 bits). Therefore, if we wish to write byte-wide data values into the first four consecutive locations in a region

of memory starting at `DDR_BASEADDR`, we must write the first to `DDR_BASEADDR+0`, the second to `DDR_BASEADDR+1`, the third to `DDR_BASEADDR+2`, and the last to `DDR_BASEADDR+3`. However, if we wish to write four half-word wide (16-bit) data values to four memory addresses starting at the same location, we must write the first to `DDR_BASEADDR+0`, the second to `DDR_BASEADDR+2`, the third to `DDR_BASEADDR+4`, and the last to `DDR_BASEADDR+6`. It therefore follows that when writing word wide (32-bit) data values, we must do so on 4 byte boundaries; 0x0, 0x4, 0x8, and 0xC. Hence, there are functions for reading and writing memory in 8-bit, 16-bit, and 32-bit transactions. The exact syntax is very simple to use; Table 7.1 shows the function templates for memory reads and writes:

Table 7.1: Syntax for reading and writing from the OCM

Write Syntax	Read Syntax
`Xil_Out8(memory_address, 8_bit_value);`	`8_bit_value = Xil_In8(memory_address);`
`Xil_Out16(memory_address, 16_bit_value);`	`16_bit_value = Xil_In16(memory_address);`
`Xil_Out32(memory_address, 32_bit_value);`	`32_bit_value = Xil_In32(memory_address);`

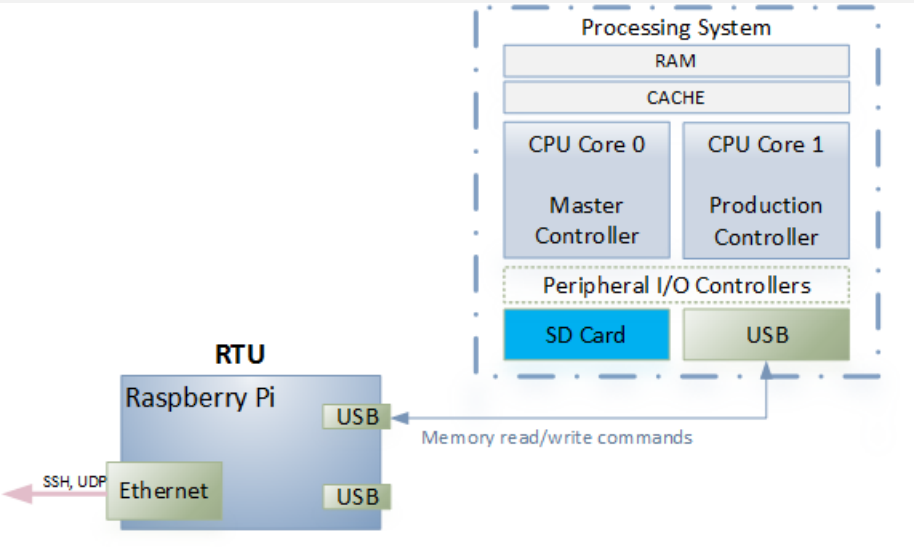


Figure 7.1: Overview of memory debug channel

Figure 7.1 shows the setup of the memory debug channel. The above read/ write functions are executed on Core 0 of the Zynq PS (PLC). Core 0 runs a baremetal application configured to run the memory access application. The RTU sends the read commands over UART. In this module, students develop a Linux application that runs on the RTU to send the commands to the PLC. The receiving application on the PLC is configured as follows:

1. The memory access is configured to read 8-bits of data from the memory location.
2. Send "1" to activate the memory debug channel.
3. Send "1" to get read access to the PLC.
4. Send the address to be read and receive the 8-bit data from the address.

Although this is a simple exercise, students should note that any application that is written could be embedded in a modified boot file as outlined in the next module.

Module Assumptions:

This module assumes the student has access to RTU.

Hands-on Exercise:

On the RTU side:

Step 1: **Write simple code in C or Node.js to send the read commands to the PLC.**

The following code can be used as a guide to setup serial communication and send data over the UART in Node.js:

```
1 var DEV_TAIGA = "/dev/ttyACM0";
2
3 // Setup Serial Port
```

```
var SerialPort = require("serialport")
var serialTAIGA = new SerialPort.SerialPort(DEV_TAIGA, {
  baudrate: 115200, parser: SerialPort.parsers.readline("--\n",
  "binary")
}, false);

serialTAIGA.open(function (error) {
  if ( error ) {
    console.log(error);
  }
  else {
    console.log('Sucessfully opened Serial Port');
        console.log(data);
        //Write your code for sending and receiving data here:
        //e.g (serialTAIGA.write(data));
    }
});
```

Step 2: **Check the received data.**

On the PLC side: (Optional)

Explore OCM memory access using the different functions described in the introduction:

Step 1: Write software on the Zynq PS that will write eight memory bytes with the following values: 0xAB, 0xFF, 0x34, 0x8C, 0xEF, 0xBE, 0xAD, 0xDE. The values should be written to consecutive addresses, starting at the Zynq OCM base address.

Step 2: Read three consecutive memory locations and display their contents to the RTU using the UART.

7.2 Boot Image Update Lab

Module Objectives:

The students will learn about:

1. Exploiting the reconfiguration channel to corrupt the memory boot image.
2. Modifying and sending the boot image through a UART channel and replacing the file on the SD card.

Module Overview:

This lab consists of the following key components:

1. Boot process: Understanding the boot process in the PLC.
2. Boot image: Understanding the contents of the boot image BOOT.BIN.
3. Image transfer: Using the remote update channel to modify the boot image in the SD card.

Module Introduction:

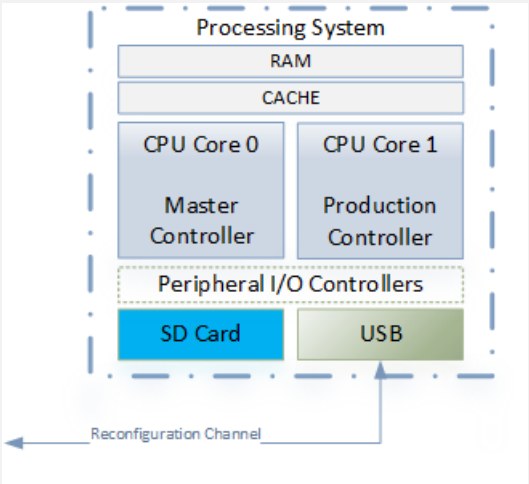


Figure 7.2: Reconfiguration channel in the PLC node

Most embedded systems have a channel in order to communicate with the outside for a variety of purposes. For example, communication channels can be used for updating the

system configuration as well as transferring files. Configuration channels can vary from using serial communications to any other communication protocol. Since a configuration channel commonly connects the embedded system to the outside, it will certainly draw interest from attackers and will potentially present major threats to system integrity. This section discusses potential vulnerabilities that might exist in configuration channels by exploring a known example.

Figure 7.2 shows the reconfiguration channel in the PLC. CPU Core 1 is the production controller. The entire system is booted through the boot file located in the SD Card. Both CPU cores have access to the peripheral I/O controller and has serial communication serving as a configuration channel. However, only Core 0 interacts with both the SD card and peripheral I/O controller once the system has booted up. Core 1 only uses the SD Card to boot up and will not interact with the peripheral I/O controller unless the system reboots.

The PLC boots using the SD card. Both cores operate based on the configuration stored in the boot file on the SD card. While the PLCs only access to the outside is through the UART, the UART can also be used to change the boot file. Potential attackers can modify the boot file on the SD card or replace the boot file with a new one.

Boot process in the Zynq PS

The Zynq platform's default boot loader consists of two small programs, namely the Boot-ROM code and the First Stage Boot Loader (FSBL). Boot-ROM executes at start-up, loads the FSBL from the SD card to OCM and executes it. The FSBL executes and copies the application from the SD card to DDR memory, and transfer control to the application start address to execute it. In case of AMP, the FSBL always runs on CPU0, is responsible for programming the PL, and copies application executable and linkable format (ELF) files to DDR memory. After loading the applications to DDR memory, the

FSBL then starts executing the first application that was loaded. CPU0 is responsible for initializing shared resources and starting up CPU1.

BOOT.BIN generation

The boot file (BOOT.BIN) contains the FSBL, FPGA bitstream file, and the ELF for the application that runs on CPU0 and CPU1. There is a configuration file that contains the names of the files that will be copied to DDR memory. The order of these files is important. For this design, the order is:

1. FSBL ELF
2. Bitstream (optional)
3. CPU0 application
4. CPU1 application

BOOT.BIN can then be generated using the bootgen command:

From the Xilinx SDK, open a command window by selecting
`Xilinx_Tools->Launch_Shell` In the shell, `cd ../bootgen`
`bootgen -image bootimage.bif -o i BOOT.BIN -w on`

For the purpose of this lab, the BOOT.BIN file will be provided.

Module Tools:

Minicom: A text-based modem control and terminal emulation program used for setting up a remote serial console [59].

Module Assumptions:

This module assumes that the student has access to the RTU. The BOOT.BIN file will be provided and will be available on the Raspberry Pi.

Hands-on Exercise:

In this lab, the image (BOOT.BIN) is to be transferred from the RTU to the PLC. The PLC is configured to receive the file as described below:

1. Send "2" to configure boot image update mode.
2. Send the file size (size of BOOT.BIN).
3. Send the BOOT.BIN file over UART.

To send the BOOT.BIN file, perform the following steps on the RTU:

Step 1: **Write a program on the Raspberry Pi to transfer the system image**

The program can be written in Node.js or C to transfer the file from the Raspberry Pi to the Zynq PS. This will permit writing the file once it is received through the UART.

The following code written in Node.js can be used as a guide to send a binary file over serial port:

```
var DEV_TAIGA = "/dev/ttyACM0";

// Setup Serial Port
var SerialPort = require("serialport")
var serialTAIGA = new SerialPort.SerialPort(DEV_TAIGA, {
  baudrate: 115200, parser: SerialPort.parsers.readline("--\n",
  "binary")
}, false); // this is the openImmediately flag [default is true]


//FIle
```

```
12  fs = require('fs');
13
14  serialTAIGA.open(function (error) {
15    if ( error ) {
16      console.log(error);
17    }
18    else {
19      console.log('Sucessfully opened Serial Port');
20      fs.readFile("BOOT.BIN", function (err,data) {
21          if (err) {
22                  return console.log(err)
23          }
24          console.log(data);
25          serialTAIGA.write(data);
26      });
27    }
28  });
```

Step 2: **Boot the system**

Once, a modified boot file is transferred and successfully stored in the SD card, a reboot is issued and the modified boot file is accessed, thus affecting the system.

7.3 Summary

This chapter discusses how a remote diagnostics channel used for maintaining the system can be exploited to gain system knowledge. The memory debug module can give system insights to the adversary, whereas the boot image module can be exploited to modify and replace the system boot image. These attacks may allow algorithms to be identified by observing memory access patterns, and this can lead to both information leakage as well as physical process dsiruption.

Chapter 8

Evaluation and Discussion

The modules offer a hands-on experience, provides flexibility of timing and operation, and sharing the lab equipments. The experience gained from these hands-on modules can be categorized as follows:

- **Benefits:** These modules are set up using commercial-off-the-shelf hardware, and can thus be easily recreated without any large up-front investment. These modules give a comprehensive view of a CPS and an interactive approach to understanding the security vulnerabilities of such systems. The remote labs enhances the learning process for two-fold reasons: (1) these open interactions speed up the process of learning new techniques, and (2) these labs enforce students to focus on implementation and testing phase of their system, making the learning process more attractive. This process also exposes students to the heterogeneity of complex systems.
- **Limitations:** In the current implementation, students need access to the internal private static router to successfully complete all the modules.
- **Assessment:** Undergraduate computer engineering students were actively involved in developing these modules in order to collect feedback and implement changes to better suit the students targeted by this work.

An evaluative survey was developed to gauge students' understanding in the subject of critical infrastructure cybersecurity. The idea is to understand student perceptions by administering a survey before and after module presentations. The survey was completed by participating students.

Few of the sample questions from the survey are as follows:

1. For a critical infrastructure under operation, what is more important; network security or process reliability?
   Network security (1), (2), equal (3), (4), process reliability (5)
2. How effectively can IT security measures protect a CPS?
   Not effectively (1), (2), somewhat effectively (3), (4), completely effectively (5)
3. How vulnerable are Internet-connected systems, how easily can such systems be detected?
   Not vulnerable (1), (2), no more vulnerable than other systems (3), (4), highly vulnerable (5)
4. How secure are network protocols to protect from attacks like man-in-the-middle attacks?
   Not secure (1), (2), somewhat secure (3), (4), very secure (5)
5. Is the software update process used for computers used in a CPS a good solution or a major issue in process reliability?
   Good solution (1), (2), (3), (4), major problem (5)
6. What is your understanding of CPS hardware software, communication and control of physical processes?
   Almost none (1), (2), some understanding (3), (4), very good understanding (5)
7. How likely are you to pursue a career that involves security of critical infrastructure?
   Not likely (1), (2), unsure (3), (4), very likely (5)

Here are some of the representative responses obtained from the students:

Student Response 1

1. For a critical infrastructure under operation, what is more important: network security or process reliability?

   Student response: ***4, process reliability***

   A process with good error handling and correction can overcome a network compromise, while a good network but unreliable process will usually yield unsatisfactory results.

2. How effectively can IT security measures protect a CPS?

   Student response: ***3, somewhat effectively***

   Network security is so easy to compromise nowadays, it almost seems like a non-issue for hackers.

3. How vulnerable are Internet-connected systems, and how easily can such systems be detected?

   Student response: ***5, highly vulnerable***

   If a device is open to the Internet, it gives a hacker a 24/7 window to remotely attempt any hacking technique.

4. How secure are network protocols to protect from attacks like man-in-the-middle?

   Student response: ***3, somewhat secure***

   Depending on the attack and the protocol, man-in-the-middle attacks may be completely blocked, or completely overlooked.

5. Is the software update process used by computers in a CPS a good solution or a major issue in process reliability?

   Student response: ***1, good solution***

   When tested, updates can effectively patch known exploits.

6. What is your understanding of CPS hardware, software, communication, and control of physical processes?

   Student response: ***3, some understanding***

I have limited experience outside of working with TAIGA over this past summer.

7. How likely are you to pursue a career that involves security of critical infrastructure?

   Student response: ***4, likely***

   The demand for jobs in embedded systems seems to be heading in the direction of security. With a large number of positions available, I'm sure that I will consider security in systems that I work with in almost all of my career positions.

Comments: *My experience over this past summer working with TAIGA was very informative. It taught me a lot about working in an environment that was unfamiliar. My first week or so of working with TAIGA was spent getting up to speed on the project they were working on. Once I had a grasp of the project, I was given the job of finding ways to break the system. It was a very open-ended job and allowed for a lot of creativity on my part. I learned much more than I would have in a classroom setting, as security and practicality of my solutions, and of TAIGA in general, were high priorities.*

Student Response 2

1. For a critical infrastructure under operation, what is more important; network security or process reliability?

   Student response: ***4, process reliability***

2. How effectively can IT security measures protect a CPS?

   Student response: ***3, somewhat effectively***

3. How vulnerable are Internet-connected systems, how easily can such systems be detected?

   Student response: ***4, vulnerable***

4. How secure are network protocols to protect from attacks like man-in-the-middle attacks?

   Student response: ***4, quite secure***

5. Is the software update process used for computers used in a CPS a good solution or a major issue in process reliability?
   Student response: ***4, big problem***
6. What is your understanding of CPS hardware software, communication and control of physical processes?
   Student response: ***3, some understanding***
7. How likely are you to pursue a career that involves security of critical infrastructure?
   Student response: ***4, likely***

Comments: *I think my summer research experience working on exploring security vulnerabilities in a cyber-physical system is enlightening. Before this experience, I only worked with network security and embedded systems individually. This experience opens the door for me to explore security issues that can be exploited on embedded systems with network enabled. I learned how to draw correct threat models to identify potential risks as well as design replay attacks on system modules. I also got the chance of exploring the security risks exist in the configuration channel of embedded systems. Overall, this summer research experience gave more ideas in designing future embedded systems.*

The survey helped us understand the students' assessment of the material. A lab setting with open-ended tasks helped the students learn on their own and evaluate the feasibility of their solutions. It helped us understand the basic awareness amongst the students about cyber threats to critical infrastructure, their interests in the field of security, and ideas about design of embedded systems. Also, it served as an aid in evaluating student understanding of a CPS system, differences between IT and CPS security, and the importance of process reliability in CPSes.

Chapter 9

Conclusions and Future Work

This work explored conventional CPS topologies to understand cyber attack vectors, and defined various modules to illustrate a possible attack pathway to compromise a process controller. A modular approach provides a open-source, hands-on experience that emphasizes the unique aspects of a CPS and raises students' awareness and skills in this field.

This work is designed to serve two purposes: It includes principles of cybersecurity as applied to CPS, understanding the vulnerabilities and threat detection, principles of network design, protocol analysis. It also supports the direct implementation of CPS testbed; configuration of various CPS elements; and the exploitation of possible vulnerabilities.

The critical infrastructure includes power generation and distribution, telecommunications and several others we depend upon for secure functioning; and failure of these systems can result in loss of electricity supply, lack of water supply and other critical outages. The approach to solving these problems is through educating engineering students to facilitate cybersecurity awareness for critical infrastructure.

9.1 Scope of Modular Approach

This thesis explored one potential attack pathway in order to gain access to the leaf-node controller. It demonstrated several vulnerabilities present in each layer of CPS and methods to exploit these vulnerabilities and penetrate further into the system.

The effectiveness of the SHODAN search engine shows how various home automation systems, traffic lights, security cameras, and SCADA systems are connected to the Internet with very few or no security safeguards in place. While this search engines exposes the lack of security measures in Internet-connected devices, these vulnerabilities can be exploited to perform protocol downgrade attacks by using ARP poisoning. The regulatory layer provides the supervisory control and monitor functionalities. Protecting this layer is imperative since the overall security of a control system network could be weakened if the supervisory system becomes an open attack vector to the CPS. These modules discuss how an adversary can manipulate message routing and network coordination to perform replay and set-point violation attacks. A combination of these attacks can enable more stealthy damage if the adversary also gains reconfiguration resources. The memory debug channel can be exploited to gain system information, whereas the system boot image can be corrupted by maliciously accessing the update channel. These modules show how an adversary can ultimately gain access to the process controller.

These modules leverage the unique characteristics of a CPS to compromise a physical process. A mock testbed is created with commercial-off-the-shelf hardware, which makes it economical and easy to replicate. This courseware exposes students to a wide range of network tools and software, specific knowledge of device architectures, and low-level programming techniques for representative CPS hardware and software. This work emphasizes thorough understanding of threats and their impacts, and bridges the IT and CPS domains just as real attacker would. Cyber defense involves preventative measures to protect systems. We highlight the defense strategies by presenting appropriate network- and system-level topics.

In future, this courseware will be evaluated for pedagogical effectiveness. A comprehensive framework is being developed which can be integrated in a curriculum for critical infrastructure security. Future work will include setting up a virtual machine environment to allow experimentation to take place without physical access and introducing forensic data analysis and risk management.

Bibliography

[1] J. Slay and M. Miller, "Lessons learned from the Maroochy water breach," *Critical Infrastructure Protection*, vol. 253, pp. 73–82, 2007.

[2] D. Kushner, "The real story of Stuxnet," *Spectrum, IEEE*, vol. 50, pp. 48–53, March 2013.

[3] R. M. Lee, M. J. Assante, and T. Conway, "ICS CP/PE Cyber-to-Physical or Process Effects Case Study German Steel Mill Cyber Attack." https://ics.sans.org/media/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf, Sans ICS, Dec 2014.

[4] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pp. 355–366, ACM, 2011.

[5] E. Luiijf, "Understanding cyber threats and vulnerabilities," in *Critical Infrastructure Protection*, pp. 52–67, Springer, 2012.

[6] The White House, "Executive Order: Improving Critical Infrastructure Cybersecurity." https://www.whitehouse.gov/the-press-office/2013/02/12/executive-order-improving-critical-infrastructure-cybersecurity, Feb 2013.

[7] Department of Homeland Security, "Homeland Security Presidential Directive 7: Critical Infrastructure Identification, Prioritization, and Protection." http://www.dhs.gov/homeland-security-presidential-directive-7, Dec. 2003.

[8] "National SCADA Test Bed." http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf, 2003.

[9] Daniel Noyes , "Cyber Security Testing and Training Programs for Industrial Control Systems ." https://inldigitallibrary.inl.gov/sti/5411182.pdf, Mar. 2012.

[10] Michael J. Assante, "Testimony on Securing Critical Infrastructure in the Age of Stuxnet." http://www.hsgac.senate.gov/hearings/securing-critical-infrastructure-in-the-age-of-stuxnet, Nov. 2010.

[11] Trusted Information Sharing Network for Critical Infrastructure Protection, "Achieving IT Resilience." http://www.tisn.gov.au/Documents/ITSEAG+Resilience+Paper+CIO+Report+(PDF).pdf, May 2010.

[12] J. Slay and E. Sitnikova, "Developing SCADA Systems Security Course within a Systems Engineering Program," in *12th Colloquium for Information Systems Security Education*, Nov 2011.

[13] J. C. Foreman, J. H. Graham, J. L. Hieb, and R. K. Ragade, "A Curriculum Model for Industrial Control Systems Cyber-Security with Sample Modules," in *27th International Conference on Computers and Their Applications*, Mar 2012.

[14] E. Crowley, "Information system security curricula development," in *Proceedings of the 4th conference on Information technology curriculum*, pp. 249–255, ACM, 2003.

[15] D. C. Rowe, B. M. Lunt, and J. J. Ekstrom, "The role of cyber-security in information technology education," in *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, (New York, NY, USA), pp. 113–122, ACM, 2011.

[16] R. K. Raj, S. Mishra, C. J. Romanowski, and T. M. Howles, "Cybersecurity as General Education." http://cisse.info/resources/archives/category/16-papers?download=187:16-2011.

[17] T. Howles, C. Romanowski, S. Mishra, and R. K. Raj, "A holistic, modular approach to infuse cybersecurity into undergraduate computing degree programs," in *Annual Symposium On Information Assurance (ASIA), Albany, NY*, pp. 7–8, 2011.

[18] J. E. Huss, "Laboratory projects for promoting hands-on learning in a computer security course," *SIGCSE Bull.*, vol. 27, pp. 2–6, June 1995.

[19] V. J. H. Powell, C. T. Davis, R. S. Johnson, P. Y. Wu, J. C. Turchek, and I. W. Parker, "Vlabnet: The integrated design of hands-on learning in information security and networking," in *Proceedings of the 4th Annual Conference on Information Security Curriculum Development*, InfoSecCD '07, (New York, NY, USA), pp. 9:1–9:7, ACM, 2007.

[20] D. Navarro, J. C. Mendez, K. Berrios, E. Ortiz-Rivera, and E. Arzuaga, "Using cybersecurity as an engineering education approach on computer engineering to learn about smart grid technologies and the next generation of electric power systems," in *Frontiers in Education Conference (FIE), 2014 IEEE*, pp. 1–8, IEEE, 2014.

[21] T. Yardley, S. Uludag, K. Nahrstedt, and P. Sauer, "Developing a smart grid cybersecurity education platform and a preliminary assessment of its first application," in *Frontiers in Education Conference (FIE), 2014 IEEE*, pp. 1–9, Oct 2014.

[22] InfoSec Institute, "SCADA Security Course." http://www.infosecinstitute.com/courses/scada_security_training.html#learn/.

[23] Eric Cole and Eric Cornelius and Justin Searle, "ICS/SCADA Security Essentials." https://www.sans.org/course/ics-scada-cyber-security-essentials.

[24] J. Weiss, *Protecting Industrial Control Systems from Electronic Threats*. Momentum Press, 2010.

[25] T. Chiluvuri, "A trusted autonomic architecture to safeguard cyber-physical control leaf nodes and protect process integrity," Master's thesis, Virginia Tech, Blacksburg, VA, 2015.

[26] O. Harshe, "Preemptive detection of cyber attacks on industrial control systems," Master's thesis, Virginia Tech, Blacksburg, VA, 2015.

[27] F. Hu, *Cyber-Physical Systems: Integrated Computing and Engineering Design*. CRC Press, 2013.

[28] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pp. 1–6, Nov 2011.

[29] J. Depoy, J. Phelan, P. Sholander, B. Smith, G. Varnado, and G. Wyss, "Risk assessment for physical and cyber attacks on critical infrastructures," in *Military Communications Conference, 2005. MILCOM 2005. IEEE*, pp. 1961–1969, IEEE, 2005.

[30] "Sustainable Security for Infrastructure SCADA." http://www.tswg.gov/tswg/ip/SustainableSecurity.pdf.

[31] L. W. Lerner, *Trustworthy Embedded Computing for Cyber-Physical Control*. PhD thesis, Virginia Tech, 2015.

[32] "Can We Trust the Chips of the Future?," *IEEE Design & Test of Computers*, vol. 28, no. 5, pp. 96–103, 2011.

[33] M. Tehranipoor, H. Salmani, X. Zhang, X. Wang, R. Karri, J. Rajendran, and K. Rosenfeld, "Trustworthy hardware: Trojan detection and design-for-trust challenges," *Computer*, no. 7, pp. 66–74, 2010.

[34] E. Chien, L. OMurchu, and N. Falliere, "W32. duqu: the precursor to the next stuxnet," in *Proc. of the 5th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2012.

[35] Symantec Security Response, "W32.Duqu: The precursor to the next Stuxnet." https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf, Nov. 2011.

[36] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 277–293, Feb 2013.

[37] N. T. Chiluvuri, O. A. Harshe, C. D. Patterson, and W. T. Baumann, "Using heterogeneous computing to implement a trust isolated architecture for cyber-physical control systems," in *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*, CPSS '15, (New York, NY, USA), pp. 25–35, ACM, 2015.

[38] O. A. Harshe, N. Teja Chiluvuri, C. D. Patterson, and W. T. Baumann, "Design and implementation of a security framework for industrial control systems," in *Industrial Instrumentation and Control (ICIC), 2015 International Conference on*, pp. 127–132, IEEE, 2015.

[39] Trusted Computing Group, Incorporated, "TPM Main Specification Version 1.2 Revision 116 Part 1 Design Principles," Mar. 2011.

[40] M. Roman, E. Bobasu, and D. Sendrescu, "Modelling of the rotary inverted pendulum system," in *Automation, Quality and Testing, Robotics, 2008. AQTR 2008. IEEE International Conference on*, vol. 2, pp. 141–146, May 2008.

[41] C. Reas and B. Fry, *Processing: a programming handbook for visual designers and artists*, vol. 6812. Mit Press, 2007.

[42] M. M. Lombardi, "Authentic learning for the 21st century: An overview," *Educause learning initiative*, vol. 1, no. 2007, pp. 1–12, 2007.

[43] P. M. Stohr-Hunt, "An analysis of frequency of hands-on experience and science achievement," *Journal of research in Science Teaching*, vol. 33, no. 1, pp. 101–109, 1996.

[44] J. Ma and J. V. Nickerson, "Hands-on, simulated, and remote laboratories: A comparative literature review," *ACM Computing Surveys (CSUR)*, vol. 38, no. 3, p. 7, 2006.

[45] J. Mezirow *et al.*, "How critical reflection triggers transformative learning," *Fostering Critical Reflection in Adulthood*, pp. 1–20, 1990.

[46] K. Hinett and T. Varnava, *Developing reflective practice in legal education*. Citeseer, 2002.

[47] "How Stuxnet Spreads: A Study of Infection Paths in Best Practice Systems." White Paper, Feb. 2011.

[48] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proceedings of the 1st International Symposium on ICS & SCADA Cyber Security Research 2013*, pp. 22–29, BCS, 2013.

[49] John Matherly, "SHODAN Search Engine." https://www.shodan.io/.

[50] T. Ylonen and C. Lonvick, "The secure shell (ssh) protocol architecture," 2006.

[51] Massimiliano Montoro, "Cain and Abel." Available at http://www.oxid.it/cain.html.

[52] Ulf Lamping and Richard Sharpe and Ed Warnicke, "Wireshark 2.0." Available at https://www.wireshark.org/.

[53] Python Software Foundation, "Python Language Reference, version 2.7." Available at http://www.python.org.

[54] Microsoft Developer Network, "Replay attacks." Security Guidance and Best Practices.

[55] "MJPG-Streamer." Available at http://sourceforge.net/projects/mjpg-streamer/.

[56] Alberto Ornaghi and Marco Valleri, "Ettercap." Available at https://ettercap.github.io/ettercap/.

[57] R. C. Pinto and J. Rufino, "Exploiting non-intrusive monitoring in real-time embedded operating systems.," in *EWiLi*, 2014.

[58] "Zynq-7000 All Programmable SoC Technical Reference Manual." http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[59] Wikipedia, "Minicom."