8-15-2014

A Virtual Hydroelectric Power System for Distributable Industrial Control System Security Research

David Brian Mudd

A virtual hydroelectric power system for distributable industrial control system security research

By

David Brian Mudd

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical and Computer Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

August 2014

A virtual hydroelectric power system for distributable

industrial control system security research

By

David Brian Mudd

Approved:

Thomas H. Morris
(Major Professor)

Bryan A. Jones
(Committee Member)

John E. Ball
(Committee Member)

James E. Fowler
(Graduate Coordinator)

Jason M. Keith
Interim Dean
Bagley College of Engineering

Name: David Brian Mudd

Date of Degree: August 15, 2014

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Dr. Thomas Morris

Title of Study: A virtual hydroelectric power system for distributable industrial control system security research

Pages of Study: 72

Candidate for Degree of Master of Science

Cyber security for industrial control systems (ICS) has been a rapidly growing area of interest and research for the last several years. The lack of an easily distributable platform on which ICS components can be built for use in security testing and result comparison among researchers presents a major issue. This thesis details the use of a virtual testbed environment to build a representative virtual hydroelectric power system (VHPS). The VHPS generates realistic Modbus/TCP network traffic between two separate ICS devices, a Master and a Slave, located on separate VMs. For security testing purposes, a method of session hijacking has been implemented as well as a Function Code Scan attack and a Setpoint Manipulation attack. The virtual environment, the VHPS, and the attacks have been packaged into an LXDE-based Fedora Spin VM for easy distribution.

DEDICATION

To Jenna.

ACKNOWLEDGEMENTS

I would first like to thank my major professor, Dr. Thomas Morris, for all of the advice and assistance that he has given me. I learned a great deal from our discussions, and his opinions were vital to this thesis. I would also like to thank Bradley Reaves for developing the virtual environment and for aiding me in the initial phase of getting the environment working properly. The email discussions with him were insightful. Thirdly, I benefited greatly from Wei Gao's experience in working with SCADA systems.

This work was supported by the United States Army Core of Engineers.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADU - Application Data Unit

AGC - Automatic Generation Control

API - Application Programming Interface

ARP - Address Resolution Protocol

ASM - Algorithmic State Machine

AVD - Attack-Vulnerability-Damage

CIP - Critical Infrastructure Protection

CSSWG - Control System Security Working Group

FC - Function Code

GB - Gigabyte

GUI - Graphical User Interface

ICS - Industrial Control System

IDS - Intrusion Detection System

IP - Internet Protocol

LAN - Local Area Network

LXDE - Lightweight X11 Desktop Environment

MAC - Media Access Control

MBAP - Modbus Application

MTU - Master Terminal Unit

NERC - North American Electric Reliability Corporation

NIC - Network Interface Controller

NIPS - Network Intrusion Prevention System

OVF	- Open Virtualization Format
OS	- Operating System
PCAP	- Packet Capture
PDU	- Protocol Data Unit
PLC	- Programmable Logic Controller
RINSE	- Real-Time Immersive Network Simulation Environment
RTU	- Remote Terminal Unit
SCADA	- Supervisory Control and Data Acquisition
SNL	- Speed No Load
TCP	- Transmission Control Protocol
vdev	- Virtual Device
VHPS	- Virtual Hydroelectric Power System
VM	- Virtual Machine

CHAPTER 1

INTRODUCTION

This chapter provides an introduction for this thesis. It details the background information and motivation. In this chapter, the hypothesis for this thesis is presented, and the meaning of the requirements are outlined.

1.1 Motivation

The motivations for this work focus on the general problems faced in industrial control system (ICS) security and the problems that ICS security researchers face. These two problems are examined in further detail in the following subsection.

1.1.1 ICS Security Problems

Industrial Control Systems (ICS) are critical systems that combine various programmable logic devices and other computer based hardware devices to control and monitor large physical processes. These systems are often also referred to as Supervisory Control and Data Acquisition (SCADA) systems. Power generation systems, electrical transmission systems, and water treatment systems are examples of SCADA systems that are referred to as critical infrastructure. The security of systems that comprised the nation's critical infrastructure needs to be of highest importance and has been a topic of focused research for several years.

Any damage done to the nation's critical infrastructure may have a large, negative impact on the nation's economy. Additionally, a large, coordinated attack on the nation's power grid could leave many homes and businesses without power and, thus, further impacting the nation's economy. Traditionally, ICS devices communicated using serial protocols, and an attacker would need to physically infiltrate the plant site to cause damage. However, recently, more ICS have begun to incorporate TCP/IP and Ethernet networking into their network. The introduction of TCP/IP and Ethernet networking into ICS networks allows for attackers to potentially access the ICS network without physical infiltration.

There have been a number of documented instances of cyber attacks negatively affecting SCADA systems. One such instance is the Stuxnet worm [7]. Stuxnet is a complex piece of malware that was tasked with modifying programmable logic controllers (PLC) to act as the attacker intended and hide the changes from the system operator. A second documented instance is the Maroochy Water Services breach in Queensland, Australia [26]. In this instance, a former Maroochy Water Services engineer penetrated the system and caused roughly 264,000 gallons on raw sewage to flow into nearby rivers. These documented incidents show that a cyber attack on critical infrastructure can produce major environmental and financial damage as well as harmful effects on humans.

The North American Electric Reliability Corporation (NERC) is tasked with ensuring the reliability of North America's bulk power system. NERC develops and enforce reliability standards such as the NERC Critical Infrastructure Protection (CIP) standard [22]. The

NERC Control System Security Working Group (CSSWG) identified the top 10 vulnerabilities to control systems in the electric sector [5]. These vulnerabilities are translatable to other critical infrastructure sectors as well. The top 10, in no particular order, are presented as follows [21]:

1. Inadequate policies, procedures, and culture governing control system security.
2. Inadequately designed control system networks that lack sufficient defense-in-depth mechanisms.
3. Remote access to the control system without appropriate access control.
4. System administration mechanisms and software used in control systems are not adequately scrutinized or maintained.
5. Use of inadequately secured WiFi wireless communication for control.
6. Use of a non-dedicated communications channel for command and control and/or inappropriate use of control system network bandwidth for non-control purposes.
7. Insufficient application of tools to detect and report on anomalous or inappropriate activity.
8. Unauthorized or inappropriate applications or devices on control system networks.
9. Control systems command and control data not authenticated.
10. Inadequately managed, designed, or implemented critical support infrastructure

The above list of vulnerabilities illustrate the driving force behind ICS security research. Vulnerability numbers 3, 4, 5, 6, 8, and 9 identify the reasons attackers are able to gain access to these critical systems. ICS security research aims to exploit these vulnerabilities in order to improve upon vulnerability numbers 1, 2, 7, and 10.

1.1.2 ICS Security Research Problem

The development of best security practices and standards in ICS critical infrastructure has failed in comparison to the development seen in other areas of information technology.

ICS security researchers and industry professionals have taken notice and are developing secure protocols [9], standards [20], and various intrusion detection systems (IDS). However, further research efforts are impeded by a major problem.

It is difficult for researchers to compare solutions to ICS security issues to discern which is more effective. This result comes from the use of different testbeds to develop solutions. Often, ICS security solutions developed in one testbed can not be tested in another testbed. For IDS-based solutions, researchers often establish their own unique threat model and corresponding network traffic to train and verify their IDS [16]. This impedes the scientific process because it often prevents researchers from independently testing and verifying each other solutions. An easily distributed, expandable platform for the development and verification of ICS security solutions is needed.

1.2 Contribution

This thesis addresses the following hypothesis in response to the previously mentioned issues.

It is hypothesized that the following are attainable:

1. It is possible to build a simple, virtual hydroelectric power system (VHPS) in an available virtual environment.
2. It is possible for the virtual hydroelectric power system to generate real network traffic.
3. It is possible to capture the VHPS network traffic.
4. It is possible to build attacks against the virtual system.
5. It is possible to constrain a distributable package of the system to under 4GB.
6. It is possible for the VHPS to be extended.

For the first requirement in the hypothesis, a simple, virtual hydroelectric power system will be developed on top of an existing virtual environment tool [24]. The virtual system will have a single generator for power generation, and the method of power generation will be rudimentary. The second requirement indicates that the virtual system will contain device with unique IP address that communicate using an industry standard communication protocol. For the third requirement, the virtual system will be composed in such a way that the ability to capture the system's network traffic will exist. The satisfy the fourth requirement, cyber attacks against the virtual system will be implemented and executed. The fifth requirement indicates that the virtual system will be packaged in such a way that it will be easily distributable for further expansion and ICS security research. The final requirement specifies that the VHPS will allow for future extensions.

The virtual hydroelectric power system serves as a proof-of-concept for building virtual ICS systems in the existing virtual environment [24]. In the existing virtual environment, the created virtual ICS systems were developed using logic obtained from real, laboratory scale models of the systems. The VHPS, contrarily, did not develop from logic obtained from a real, laboratory scale model. The goal of the VHPS is to prove that realistic, virtual models of ICS system can be developed for ICS security research with and without an existing laboratory scale model from which to pull information.

The subsequent chapters detail the contributions for this thesis. Chapter 2 presents a summary of related works. The summary includes information about ICS in general, specifics of hydroelectric power systems, the virtual environment used to build the virtual hydroelectric power system, and related ICS security projects. Chapter 3 provides more detailed information on the specific requirements including sub-requirements and further motivations. Chapter 4 explains the developed virtual hydroelectric power system in detail. It also details the setup used to satisfy the second requirement in the hypothesis and the packaging method used to satisfy the fifth requirement. Chapter 5 describes the attacks developed against the virtual system and explains the network setup that allows for the network traffic to be captured. Finally, Chapter 6 provides a summary of how the hypothesis was satisfied and presents potential future work.

CHAPTER 2

RELATED WORKS

This chapter presents an overview of works related to this thesis. First, an overview of ICS is provided. Then, a general overview of hydroelectric power systems is given. Third, the virtual environment in which the virtual hydroelectric power system is built is described. Last, other related ICS security projects are detailed.

2.1 Industrial Control Systems

Industrial Control Systems (ICS) are critical systems that combine various programmable logic devices and other computer based hardware devices to control and monitor large physical processes. These systems are often also referred to as Supervisory Control and Data Acquisition (SCADA) systems. Power generation systems, electrical transmission systems, and water treatment systems are examples of SCADA systems that are referred to as critical infrastructure.

An example of an ICS is shown in Figure 2.1. ICS send commands to control the controlled physical process and collect data relayed back over the feedback loop. A system operator sits behind the human-machine-interface (HMI) to monitor and control the process. The HMI interfaces with the controller, often referred to as the Master Terminal Unit (MTU). In ICS, the MTU is normally a programmable logic controller (PLC). In addition to the HMI, the MTU interfaces with Remote Terminal Units (RTU), which are often referred to as slaves. Like the MTU, RTUs can also be PLCs. In Figure 2.1, the MTU and RTU are combined as the block named ‘Controller’. In addition to the MTU, the RTU interfaces with actuators and sensors.

2.1.1 Programmable Logic Controllers

PLCs are computer control devices designed for ICS. They monitor the state of input devices and control the state of output devices based upon decisions made through a customized program. PLCs are typically programmed with ‘ladder logic’, which is a graphical programming language that mimics the control system. As ICS rarely change, the ‘ladder logic’ with which the PLC is programmed rarely changes. So, in essence, a PLC follows a processing loop of reading inputs, executing the ladder logic, and updating outputs accordingly.

2.2 Hydroelectric Power Systems

Hydroelectric power systems provide roughly 7% of power generated in the United States. In the renewable energy sector, hydroelectric power accounts for roughly 66% of renewable generation [19]. There are three types of hydroelectric power plants: diversion, pumped storage, and impoundment [28]. A diversion (or run-of-river) facility routes a portion of a river through a canal and does not always require a dam. A pumped storage facility pumps water from a lower reservoir to an upper reservoir to store energy when electrical demand is low. When electrical demand is high, it release water from the upper reservoir to the lower reservoir to produce energy. The impoundment (or dam-based) is

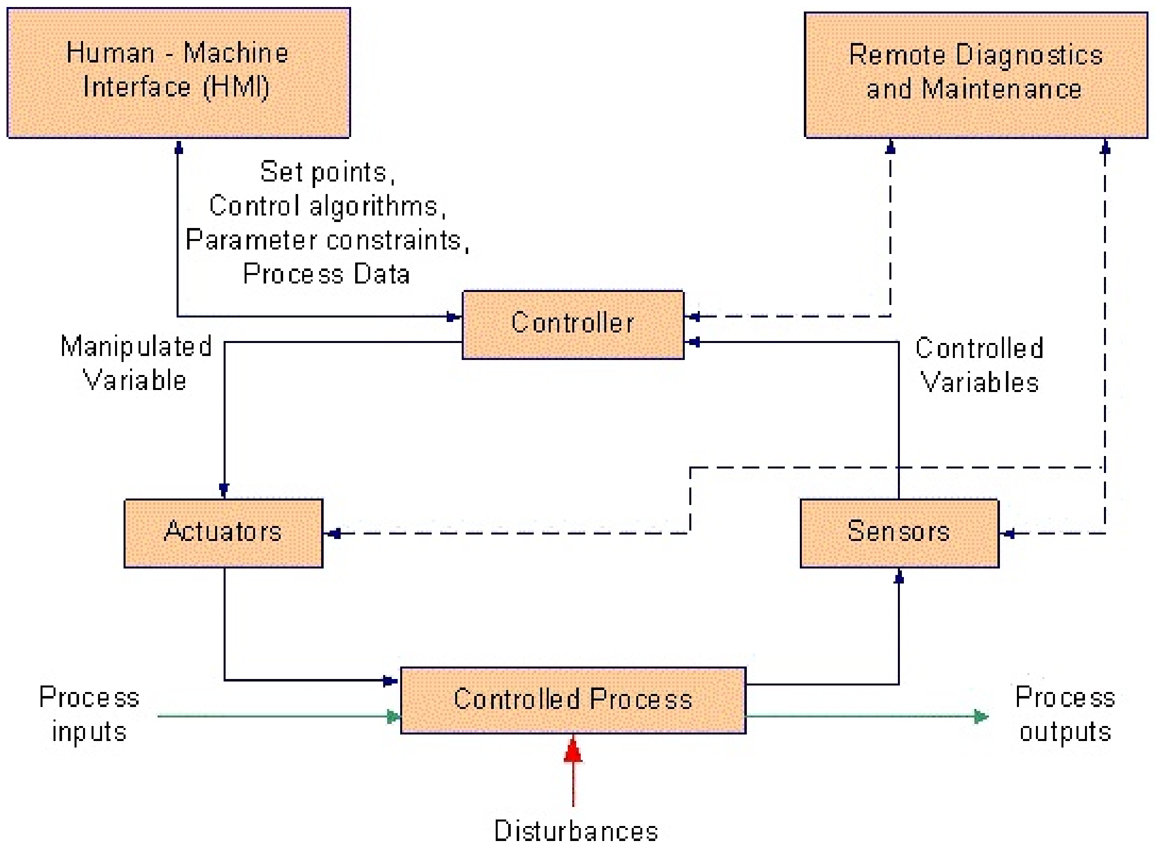

Human - Machine Interface (HMI)
Remote Diagnostics and Maintenance
Set points, Control algorithms, Parameter constraints, Process Data
Controller
Manipulated Variable
Controlled Variables
Actuators
Sensors
Process inputs
Controlled Process
Process outputs
Disturbances

Figure 2.1

ICS Diagram [4]

the most common type of hydroelectric power plant. Figure 2.2 presents a diagram of a dam-based hydroelectric power system. A dam-based facility stores water in a reservoir. The VHPS is a dam-base hydroelectric power system.

2.2.1 Components

As shown in Figure 2.2, a dam-based hydroelectric power facility is comprised of many components. First is the reservoir. The reservoir is an impoundment from the dam and is used to store water. The gate controls the release of water from the reservoir into the penstock. The penstock is a pipe that leads water from the reservoir to the turbine. The turbine is a mechanical device that rotates as water travels from the reservoir, down the penstock, and hits the turbine. The turbine converts the potential energy of the water in the reservoir into mechanical energy. As the turbine spins, so does the generator. The generator converts the turbine's mechanical energy into electrical energy.

2.2.2 Start-up Process

Before the system begins to generate power, it must follow a start-up process. The first step in the start-up process is the sounding of an alarm. The alarm sounds for 30 seconds and serves as a warning for anyone on or near the reservoir that the system is about to begin operation. After the alarm finishes sounding, the brakes are released on the control gates and the gates are opened to the breakaway position. The breakaway position is the position the control gates are opened in order to overcome static friction and allow for relative motion [1]. As shown in Figure 2.3, the breakaway position may be near 35%. Then, the gates are slowly brought back down to the Speed No Load (SNL) position. As shown

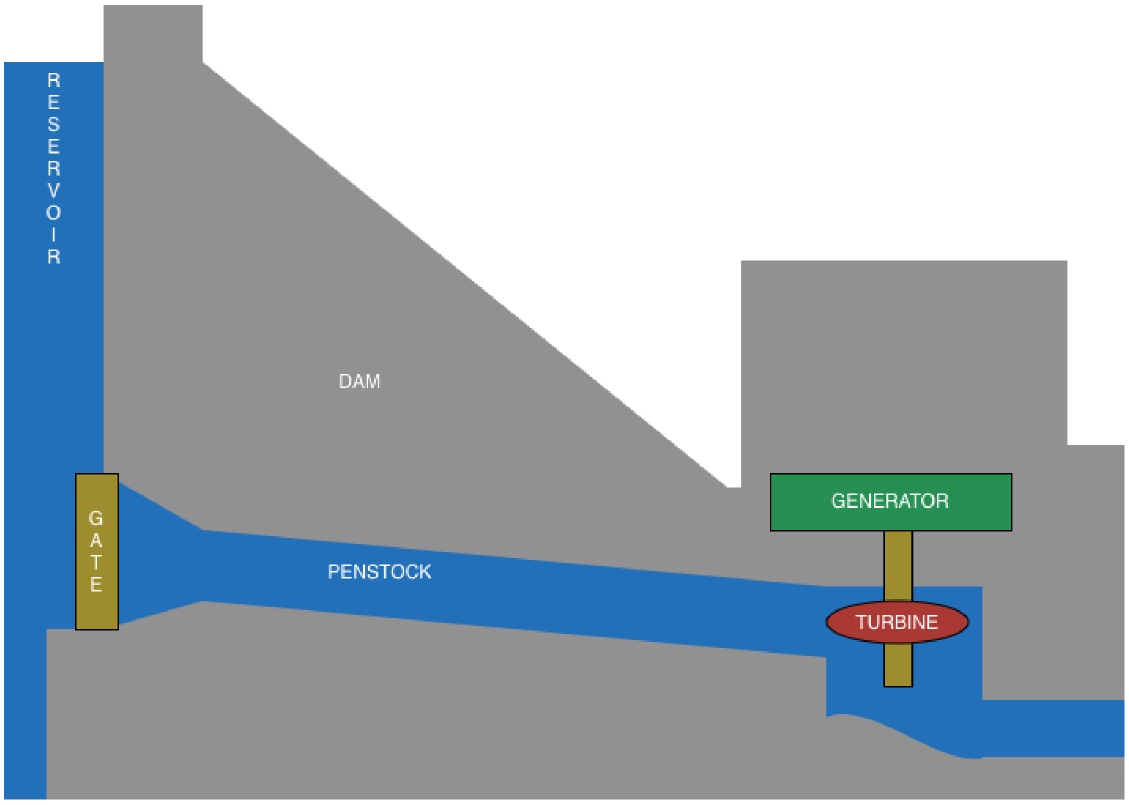


Figure 2.2

Dam-based Hydroelectric Power Overview

in Figure 2.4, this may be around 10%. During a normal day, there are times (usually hours) where the facility is not needed to handle the electrical load. The SNL position is the position the gates are kept when then the facility is on but not contributing to the grid. After the gates are in the SNL position and the system is ready to begin generation, the gates open and the system waits for the turbine to reach 95% maximum speed. Next, the exciter and the synchronizer are both switched on. With a terminal voltage reference point, the exciter sends an electrical current to the rotor. The electrical current provided by the exciter is needed to produce the rotor magnetic field and control the terminal voltage. The synchronizer is a system control device that adjusts the terminal frequency and voltage to match the power grid frequency and voltage. After this, the synchronizer closes the breaker and turns off. The breaker is an electrical switch that connects the facility to or disconnects the facility from the power grid. The hydroelectric power system can now contribute power to the grid.

2.2.3 Power Generation

In a dam-based hydropower system, water is stored in an upper reservoir behind the dam and control gates. When the system operator starts the system for power generation according to the given schedule, the start-up process begins. When the start-up process finishes, the system begins power generation. As the control gates open, the water from the upper reservoir travels through the control gates and into the penstock. The penstock is a pipeline structure that leads to the turbine. Figure 2.5 illustrates the turbine-generator architecture. As the water hits the blades of the turbine, the blades begin to spin. The turbine

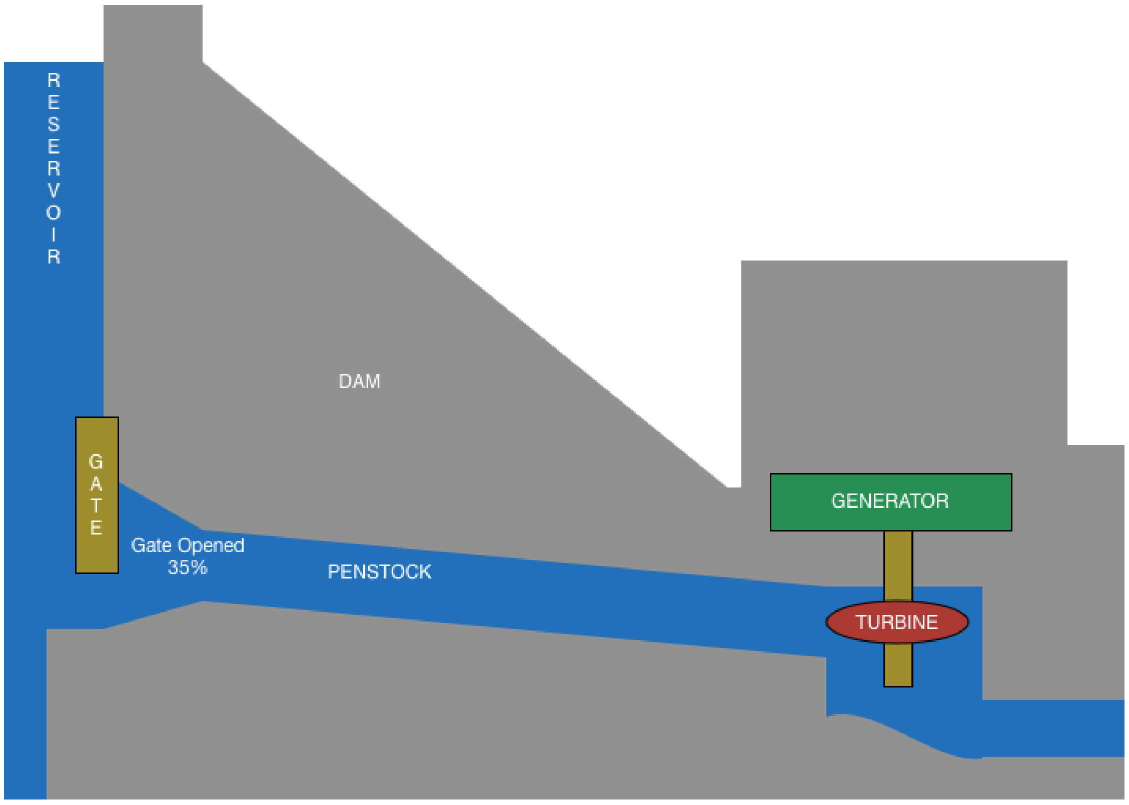


Figure 2.3

Breakaway Position

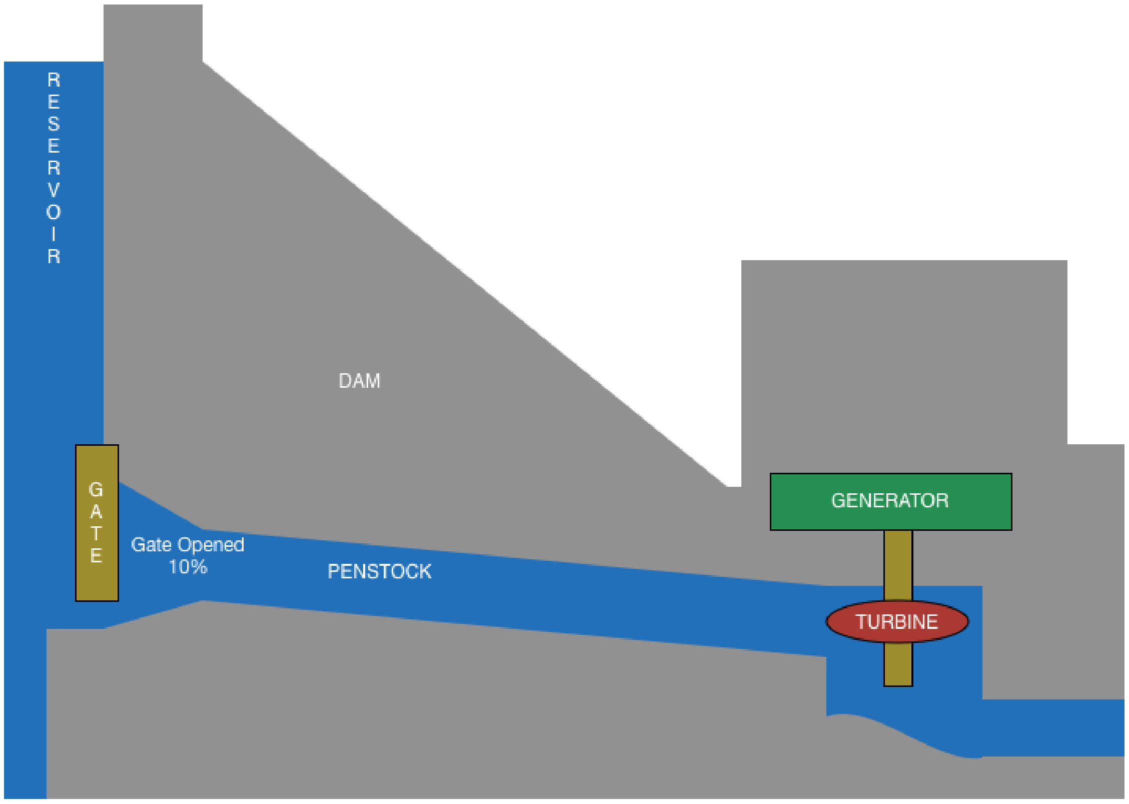


Figure 2.4

SNL Position

is connected to the generator by way of a shaft. Specifically, the shaft connects the turbine to the generator's rotor; therefore, the rotor spins along with the turbine. The exciter is a device that sends an electrical current to the rotor. The rotor is a series of electromagnets that spin within the generator's stator. The stator is a coil of tightly wound copper wire. The mechanical action of spinning the rotor within the stator creates a magnetic field between the two and generates an alternating current [23]. The synchronizer adjusts the frequency and voltage of the alternating current at the terminal to match the frequency of voltage of the grid.

2.2.4 Shutdown Process

To stop power generation, the system must follow a shutdown process. The breaker is opened to disconnect the terminal from the grid. Then, the exciter is turned off to stop providing current to the rotor. Also, the control gates are slowly closed to prevent water from the upper reservoir from flowing into the penstock. After these steps, the system has completed the shutdown process.

2.3 Virtual Environment

The Open Virtual Testbed [24] is an open source virtual environment written in Python that is both flexible and expandable. The architecture of the virtual environment is shown in Figure 2.6. The virtual testbed can realistically emulate both serial and TCP/IP ICS communications. It is comprised of three discrete components that allow for various virtual SCADA systems to be developed. The components are: the process simulator, the virtual device, and the configuration file.

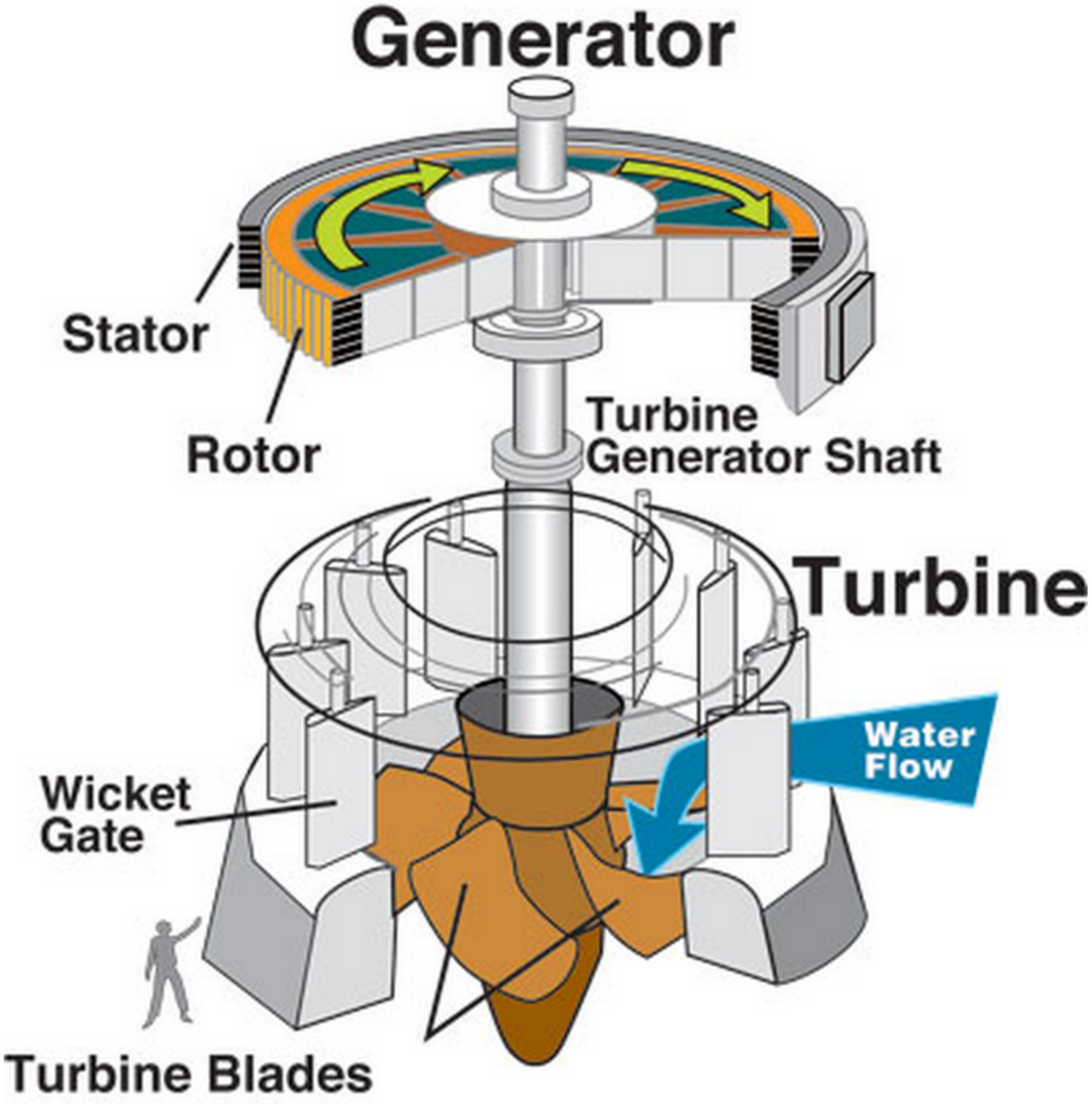


Figure 2.5

Turbine-Generator Architecture [23]

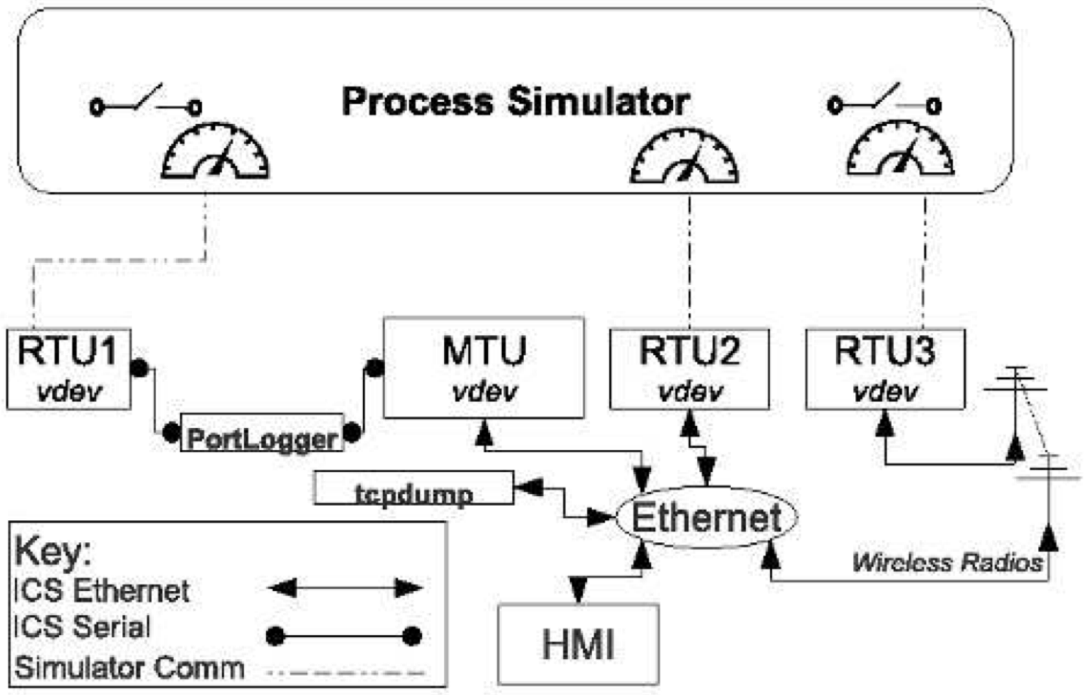


Figure 2.6

Open Virtual Testbed Architecture [24]

2.3.1 Process Simulator

The goal of the process simulator is to model the mechanics of the physical process. Additionally, the simulator is responsive to control inputs from the ICS. As in a real ICS, the simulator communicates with virtual devices. The virtual devices send control inputs to the simulator, and the simulator sends system measurements to the virtual devices. The control inputs received by the simulator are used to control actuators such as motors and pumps [24].

2.3.2 Virtual Device

In the virtual environment, virtual devices (vdevs) are the ICS devices that make and communicate control decisions. In real ICS, these are typically PLCs or Relays. Vdevs communicate with each other via real ICS protocols and follow a Master-Slave model. The Master vdev is the virtual equivalent of the MTU. It interfaces with the Slave vdev, which is the virtual equivalent of the RTU. The Slave vdev interfaces with the simulator. Vdevs are the most vital component of the virtual environment [24].

2.3.3 Configuration File

The configuration file describes the system characteristics and behavior that are not outlined in the simulator or vdevs. It includes the configuration information for both the simulator and the vdevs. For the simulator, the configuration file includes interface type, interface configuration information, and the initial values for simulator variables. For the vdevs, the configuration file includes the name and identification number of each vdev, the names and initial values of data objects (Points), interface timeout information, ICS protocol type, and ICS interface addresses and ports [24].

2.4 Testbed Projects

Many researchers have developed virtual testbeds for examining SCADA system security. This section outlines some of these virtual testbeds.

In [13], the authors present a SCADA testing environment called the ICS Sandbox. The ICS Sandbox is a compromise between a physical ICS implementation and a fully virtual simulator. It is an emulator that duplicates the behavior of a real system. The goal of the ICS Sandbox is to use known attacks to perform impact assessment and evaluate the effectiveness of network defenses in preventing the known attacks. While the ICS Sandbox duplicates the behavior of a real system, it is not easily distributable.

[17] describes Mississippi State University's SCADA testbed. The SCADA testbed uses commercial hardware and software and is comprised of 5 laboratory-scale ICS and an electrical substation. The 5 laboratory-scale ICS are a raised water tower, a gas pipeline, a factory conveyor, a water storage tank, and an industrial blower. The commercial equipment includes HMIs, MTUs, RTUs, and PLCs. While the testbed delivers real ICS operations and communication, it is expensive and not transportable.

In [2], a simulated testbed for electrical SCADA systems is presented. The Real-Time Immersive Network Simulation Environment (RINSE) was used to model communication between simulated devices, and the power grid was simulated using the PowerWorld software. The simulated devices communicate using the DNP3 communications protocol. The simulated testbed can integrate with real hardware. However, no comparison of simulated network traffic to real network traffic was performed.

In [10], the authors physically emulate the various states of a power plant using pipes, valves, sensors, and pumps. The system directly connects to a SCADA server through their field and process networks. Additionally, the authors have developed four attack scenarios. The first is Remote Authentication Dial In User Service Denial-of-Service. The second is Information Communications Technology Worm Infection. The third attack scenario is Process Network Malware Infection. Phising Attacks and Local Domain Name Server Poisoning is the final attack scenario. This work is also not easily distributable.

2.5 Cyber Attack Taxonomies

As a result of the recent surge in research surrounding ICS, methods of classifying attacks against SCADA systems have been developed. Attack taxonomies identify different types of vulnerabilities. Testbeds are needed to study vulnerabilities, test solutions, and discover new vulnerabilities. This section outlines recently developed taxonomies.

In [29], the authors present a method of classifying cyber attacks. The first is Cyber Attacks on Hardware. These attacks are any that can cause devices to fail or alarms to not go off at appropriate times. The second is Cyber Attacks on Software and includes buffer overflows and SQL injections. The final class is Attacks on the Communication Stack. The authors break down attacks on the Network, Transport, and Application layers of the TCP/IP model.

In [18], the authors present a catalog of 17 attacks against SCADA control systems. The 17 attacks are grouped into 4 attack classes: Reconnaissance, Response Injection, Command Injection, and Denial-of-Service. The Response Injection class is divided into two subclasses: Naive Malicious Response Injection and Complex Malicious Response Injection. The Command Injection class is divided into three subclasses: Malicious State Command Injection, Malicious Parameter Command Injection, and Malicious Function Code Injection.

In [8], the authors developed an Attack-Vulnerability-Damage (AVD) model of real cyber attacks against energy control systems. As indicated by the title, the model is comprised of three categories: the attack, the vulnerability, and the damage caused by the attack. The attack category is sub-categorized into three parts: the origin, the action, and the target. The vulnerability category includes three classes: configuration, specification, and implementation. The damage category is made of three subcategories: state effect, performance effect, and severity. The goal of the AVD taxonomy is to allow researchers and practitioners to further understand and share common knowledge about various cyber attacks.

In [6], a taxonomy of attacks against the DNP3 protocol is presented. The DNP3 protocol is used by more than 75% of the energy sector for control purposes. The taxonomy classifies attacks into four categories based upon the data link, pseudo-transport, and application layers of the DNP3 protocol. Common Attacks are the first category. Attacks of this category are common to all three layers. The authors identified three attacks for this category. The next category is Data Link Layer Attacks. The authors pinpointed 12 attacks for this category. Pseudo-transport Layer Attacks are the third category, and the authors list 5 attacks. The final category is Application Layer Attacks. For this category, 17 attacks were identified.

CHAPTER 3

REQUIREMENTS

This chapter details the various requirements necessary to create a comparable environment that is useful for ICS cyber security research. The first requirement is that the VHPS should follow normal hydroelectric power system operation. The second requirement is the VHPS should use normal network communication. Third, the ability to capture network communication should exist. As an important part of ICS security research, the fourth requirement is that cyber attacks should be implemented and deployed against the VHPS. The fifth requirement is that the VHPS be easily distributable for security research. The final major requirement is that the VHPS be extendable.

3.1 Normal Hydroelectric Power System Operation

A simple, virtual model of a dam-based hydropower system must be created. The virtual environment [24] in which the simple model will be developed includes virtual representations of laboratory-scale SCADA systems in Mississippi State University's SCADA testbed [17]. The creation of the simple, virtual hydroelectric power model will serve as a proof-of-concept for the development of virtual models of non-existing physical testbeds in the virtual environment. The model must include representations of the basic core components and the system logic needed for normal operation. The core physical components include: the control gate, the penstock, the turbine, the breaker, and the sensors. The system logic includes the working state and the transitional states from System Off to System On and from System On to System Off. The two transitional states are referred to as the start-up process and the shutdown process.

3.1.1 Simple GUI

For the sake of readability and usability, the system must also have a simple graphical user interface (GUI). The simple GUI should represent a simple human-machine interface (HMI). The HMI is an interface used by system operators to control and monitor the system. The simple GUI should contain methods to control the system as well as display important system values.

3.2 Normal Network Communication

It is important for the virtual system to generate real network traffic. Currently security solutions; such as Snort, Bro, and machine learning-based IDS; need normal network traffic in order to provide meaningful results. Snort is a network intrusion prevention system (NIPS). NIPS monitor network traffic for suspicious actions. Snort analyzed network traffic in real-time and uses defined rules to detect malicious activity [27]. Bro is a passive network analyzer that inspects network traffic in depth for suspicious activity [3]. Machine learning-based IDS use algorithms to statistically analyze datasets for patterns in order make autonomous decisions about new network packets [25]. The use of TCP/IP, UDP/IP, and serial communications allow for interaction with real devices and real software.

3.3 Capture Communication

For the purpose of analyzing network traffic, the ability to capture the network traffic between the virtual devices must exist. The ability to capture the network traffic allows for real traffic validation and communication analysis. Useful datasets for security research can be built from captured communication streams. Datasets containing real network traffic are vital to the advancement of ICS security research.

3.4 Deploy Cyber Attacks

For the virtual system to be useful in cyber security research, the ability to deploy cyber attacks against the system is a necessity. The cyber attacks should be realistic in nature. For meaningful results, the cyber attacks should, in theory, be potentially applicable to non-virtual systems.

3.4.1 Capture System Measurements during Cyber Attack

The ability to capture and extract system measurements from the network stream during a cyber attack must exist. During cyber attacks, the goal of the attacker may be to alter certain system variables in an effort to cause physical damage to the system. This ability to capture system measurements during an attack allows for the testing of current security solutions such as Intrusion Detection Systems (IDS). IDS use rules to aide in the identification of potentially malicious traffic. In order to effectively apply the rules, the IDS needs the ability to capture measurements for analysis. The ability to capture system measurements during an attack also allows for the building of datasets to be used for further testing. This also allows for the study of attack impact.

3.5 Distributable

One of the problems with ICS cyber security research is the lack of an easily obtainable and distributable platform. Often, researchers create their own physical testbeds for use in ICS security research. This causes problems when attempting to compare results of current security solutions. A security solution may work well on one virtual environment but not work as well for another. The scientific method requires independent, third party validation of results.

To mitigate these problems when comparing results, a reliable, open source virtual environment should be easily obtainable for all researchers to use. With the availability of an easily distributable virtual environment, a researcher could build a virtual ICS system, package it along with the virtual environment, and make the pair available for all to duplicate and verify.

To be better distributable, a compressed file size limit of no more than 4 gigabytes (GB) should be met. The file size of 4GB was chosen because computer devices that use the FAT32 file system cannot handle files that are larger than 4GB. This constraint ensures that the system can be transported to any machine.

3.6 Extendable

The VHPS must allow for extensions to be made. The requirement of Normal Hydroelectric Power System operation states that the system should be simple. In addition to being distributable, the VHPS must be extendable so a better model can be made in the future. Such extensions potentially include more generators and an automatic generation control (AGC) system. An AGC system adjusts the power output of multiple generators in response to electrical load changes.

CHAPTER 4

SYSTEM OVERVIEW

This section provides detail for the specific components and configurations that make up the VHPS system. First is the virtual environment in which the VHPS was developed. Next, the VirtualBox configurations are explained. Third, the Communication section explains how the normal communication requirement is satisfied. The specifics of the VHPS and how the requirement of normal operation is satisfied is detailed in the fourth section. The fourth section in this chapter explains how the VHPS is packaged for distribution and meets the size limit requirement. The final section describes how the VHPS is extendable.

4.1 Virtual Environment

The VHPS is built upon a previously developed open virtual environment [24]. The virtual environment is comprised of three major components: the vdevs, the simulator, and the configuration file.

4.1.1 Virtual Devices

The virtual devices within the virtual environment represent devices, such as PLCs, that are often found in ICS. These devices contain control logic that is used to execute decisions for the physical process. Each device is represented as it's own instance vdev within the virtual environment. For the VHPS, there are two vdevs. They are the Master vdev and the Slave vdev. The vdevs communicate between each other as real devices would via Modbus/TCP and Modbus over Serial Line, and they follow the Client-Server model.

Each vdev also has its own set of Points. A Point is a register class that is used to store various system information such as varying states or measurements. In the VHPS, the Points are either input registers, holding registers, or single-bit output registers (coils).

4.1.1.1 Master

In the Client-Server model, the Master vdev is the client. The Master vdev sends queries to the Slave vdev and waits for a response. For the VHPS, these queries are typically either read commands or write commands. The logic within the Master vdev instance controls which of the Slave Points the Master needs to read into its own set of Points as well as which of the Slave Points the Master needs to write values to. The job of the Master vdev is to monitor and control the Slave. Table 4.1 and Table 4.2 name the Slave Points that the Master reads and writes respectively.

4.1.1.2 Slave

In the Client-Server model, the Slave vdev is the server. It receives queries from the Master vdev and reacts accordingly. Every execution cycle, the Slave copies the values of its internal Points to those named in Table 4.1 and Table 4.2. When the Master sends a read request, the Slave responds with the values contained in the points from Table 4.1.

The Slave vdev also contains the logic for system operation. Real PLCs use either ladder logic or C to define the logic. The logic for the Slave vdev (and the Master) is written

Table 4.1

Read Slave Points

Point Name	Description
MstrRPenstockGate	The control gate is open this amount (percent)
MstrRWaterFlow	Flow rate of water within the penstock
MstrRTurbineSpeed	RPM of the turbine
MstrRPowerOutput	Amount of power currently generated
MstrRAlarmWaterFlow	Alarm for flow rate out of bounds
MstrRAlarmTurbineSpeed	Alarm for RPM out of bounds
MstrRMANmotor_off	Flag for control gate motor off
MstrRMANmotor_fwd	Flag for control gate motor opening gate
MstrRMANmotor_rvs	Flag for control gat motor closing gate
MstrRGovernor	Flag for Governor on/off
MstrRExciter	Flag for Exciter on/off
MstrRSynchronizer	Flag for Synchronizer on/off
MstrRTerminalVSetPoint	Terminal voltage
MstrRSystemVoltage	System voltage
MstrRTerminalFreq	Terminal frequency
MstrRSystemFreq	System frequency

Table 4.2

Written Slave Points

Point Name	Description
MstrWPowerNeeded	Amount of power needed to be produced
MstrWHH_WaterFlow	Flow rate high limit
MstrWHH_TurbineSpeed	RPM high limit
MstrWBreakerStatus	Flag for breaker open/closed
MstrWSystemMode	System in automatic or manual
MstrWgateSetPoint	Control gate setpoint (for manual mode)
MstrWstart	Starts or stops the system

in Python. This includes the execution flows of the start-up process, power generation, and the shutdown process. It controls the motors for the control gates to satisfy the power needed requirement. While the Master controls the Slave, the Slave is in control of the physical process execution.

4.1.2 Simulator

The simulator models the physical mechanics of the ICS. As in real ICS, the simulator is also responsive to control inputs. The simulator is characterized by states. The states represent the various states that the physical process can be in while in operation.

In the VHPS, the simulated mechanical operation is the opening and closing of the control gates. For the VHPS, the simulator has three states: 'motor_fwd' (motor forward), 'motor_rvs' (motor reverse), and 'motor_off'. All three states are boolean values. When 'motor_fwd' is true, the other two states are false and the control gates open wider. When 'motor_rvs' is true, the other two states are false and the control gates begin to close. When 'motor_off' is true, the other two states are false and the control gates to not move. Figure 4.1 presents an algorithmic state machine (ASM) of the simulator when the VHPS is in automatic mode.

4.1.3 Configuration File

The configuration file contains the configuration information for all of the vdevs in the VHPS as well as the configuration information for the simulator. The configuration information for the vdevs includes the names, the identification numbers, the Point names and metadata, interface timeout length, and ICS interface configuration information. Ta-

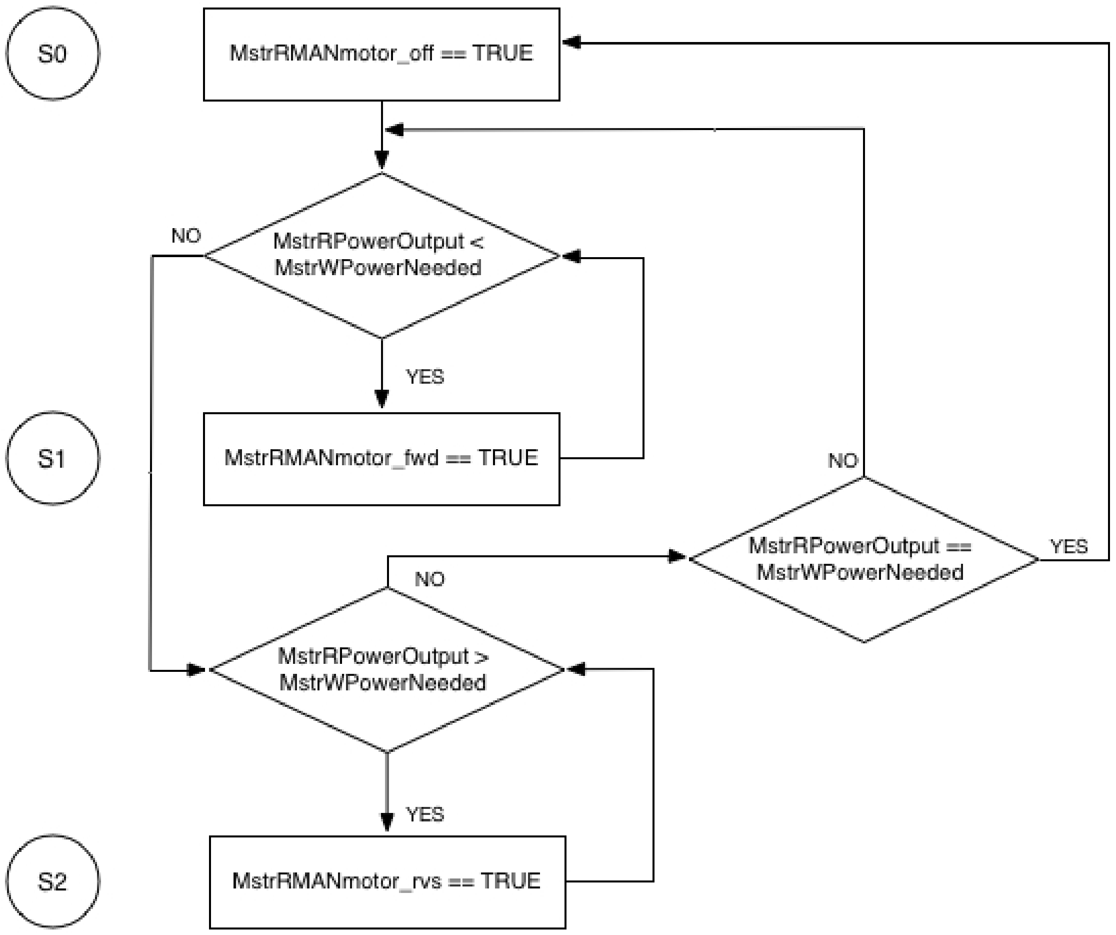

S0
MstrRMANmotor_off == TRUE
NO
MstrRPowerOutput < MstrWPowerNeeded
YES
S1
MstrRMANmotor_fwd == TRUE
NO
MstrRPowerOutput == MstrWPowerNeeded
YES
NO
MstrRPowerOutput > MstrWPowerNeeded
YES
S2
MstrRMANmotor_rvs == TRUE

Figure 4.1

Simulator ASM

ble 4.3 shows an example of a Point for the VHPS Slave vdev. The simulator configuration information includes the simulator interface information and the simulator variable initialization.

Table 4.3

Example Slave Point

Parameter	Value
name	start
typ	0
value	0
addr	12
blockname	inputreg
blocktype	3
datatype	boolean

There are two methods of modifying the system configuration. The first way is to manually edit the file named ‘config’. This file is the actual text-based configuration file that configuration information is pulled from. The second method is to edit the file named ‘mkconfig.py’. It is a python script that generates the text-based configuration file.

4.2 VirtualBox Configuration

The use of virtualization software provides many benefits. One benefit of virtualization software is the ability to have multiple operating systems with differing Internet Protocol (IP) addresses on a single host system. In SCADA systems, each device has its own IP address. So, for the VHPS, virtualization software allows for each vdev to execute in separate VMs with each having its own IP address. The VHPS executes upon a VirtualBox configuration.

4.2.1 Virtual Device VMs

The VHPS has two vdevs. They are the Master (client) vdev and the Slave (server) vdev. Both of the vdevs are contained in their own VM in order to have their own IP address. Both of the VMs have a copy of the virtual environment and contain all the files needed to execute either the Master vdev or Slave vdev.

4.2.2 Host-Only Network Configuration

In order to generate realistic traffic, both the Master and the Slave vdevs have their own unique IP address within Virtualbox. To achieve this, the VMs are connected to a host-only network within VirtualBox. VirtualBox's host-only network configuration allows for the VMs connected to it to speak to each other without a physical network interface. VirtualBox, instead, creates a software network interface on the host machine. Currently, the configuration of the host-only network interface for the VHPS uses a DHCP Server to assign IP addresses. However, the DHCP server can be disabled to assign static IP address to the vdevs.

The mkconfig.py file located on the VMs can be executed to adjust the IP addresses of the vdevs in the configuration file if needed. However, all changes need to be made in all of the VMs. For example, if the IP address for the Master vdev is changed, then the mkconfig.py will need to be executed and altered on both VMs to reflect the change of IP address. If the alteration is not made on the Slave VM, then the two will not communicate properly because the Slave vdev will not recognize the Master's IP address as valid.

4.3 Communication

Normal network communication is a requirement for the VHPS. To accomplish this requirement, the VHPS must include two sub-requirements. The first is that the vdevs have their own unique IP address. The separation of the vdevs onto different VMs connected to a host-only network interface accomplishes this sub-goal. The second is that the vdevs communicate using a standard ICS protocol. The vdevs of the VHPS communicate using the Modbus/TCP protocol.

4.3.1 Modbus/TCP

In SCADA systems, Modbus is a very commonly used protocol. Modbus adheres to the Client-Server model, specifically a Master-Slave model. The Modbus protocol has two flavors: Modbus over Serial Line and Modbus/TCP. Modbus over Serial Line is used for serial connections, and Modbus/TCP is used for TCP/IP connections. The VHPS uses Modbus/TCP for this reason.

In Modbus, the Points are classified as four tables: discrete inputs, coils, input registers, or holding registers. A discrete input is a single bit, read-only register. A coil is a single bit, read-write register. Input registers are read-only, 16-bit registers. The last table, holding registers, are read-write, 16-bit registers [15].

Modbus/TCP has two types of frames: the Protocol Data Unit (PDU) and the Application Data Unit (ADU). The PDU contains two fields: the function code and the data

[14]. The Modbus function code is a single byte field that specifies an action that the Slave should follow. The Modbus protocol defines many function codes; however, not all function codes are valid for all devices. For the VHPS, there are two function codes that are mainly used to send requests to the Slave. The first function code is 0x03 and is named "Read Holding Registers". The data field for this request from the Master vdev includes the number of registers to read and the starting address. The second function code is 0x10 and is named "Write Multiple Registers". The data field for this request includes the first register address in which to write, the number of registers that are going to be written, the number of bytes that will be written, and the register values to be written.

The ADU contains the PDU and the Modbus Application (MBAP) Header. The MBAP Header is 7 bytes long and is added to the beginning of the frame. The MBAP Header is comprised of four fields. The first field is the Transaction Identifier. The Transaction Identifier makes up two of the 7 bytes of the MBAP Header. It is set by the Master and increments by one with each request. The Protocol Identifier is the next field. It is also two bytes in length. It is always 0x0000. The third field is the Length field. The Length field is two bytes long and identifies the total number of bytes following this field in the rest of the frame. The last field is a single byte long. It is named the Unit Identifier. The Unit Identifier is used to identify the Slave [14]. Figure 4.2 shows the components of the Modbus/TCP ADU.

Figure 4.3 shows a screen capture of the communication between the Master and Slave vdevs. Each vdev also has its own unique IP address. As a result, the VHPS satisfies the

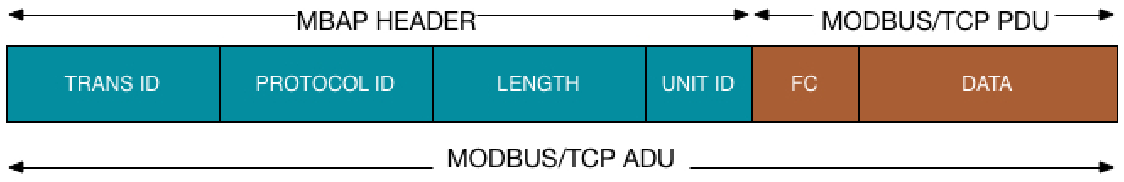


Figure 4.2

Modbus/TCP Application Data Unit

requirement of generating normal network communication. Figure 4.4 shows the normal network communication flow between the Master and Slave vdevs.

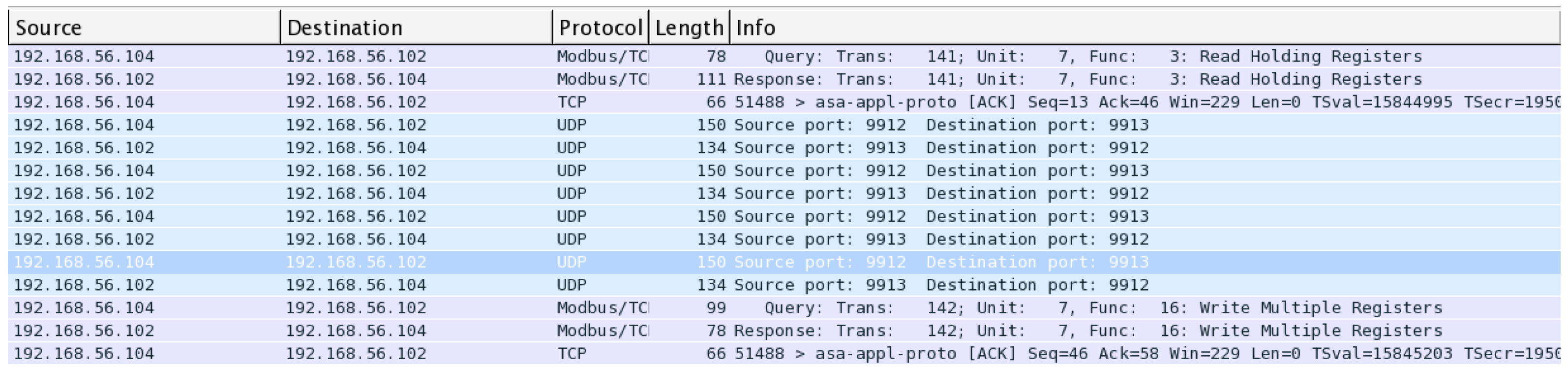
4.4 VHPS

In order to satisfy the requirement of following normal operations for a hydroelectric power system, the VHPS follows the processes of starting up, generating power, and shutting down. The following section provides the details of the three processes.

4.4.1 Start-up Process

Figure 4.5 shows an ASM for the start-up process. When the operator of the VHPS presses the Start button, the VHPS begins the start-up process. The first part of the start-up process is to sound the alarm for thirty seconds.

In normal operation, after the alarm finishes sounding, the control gates are opened to the breakaway position. For the VHPS, the breakaway position for the control gates is 35%. The breakaway position value does not change. Once the gates reach the breakaway position, they are brought back down to the SNL position. The SNL position for the control gates is 10%. Figure 4.6 shows the alarm sounding for 30 seconds after the start button is



Source	Destination	Protocol	Length	Info
192.168.56.104	192.168.56.102	Modbus/TC	78	Query: Trans: 141; Unit: 7, Func: 3: Read Holding Registers
192.168.56.102	192.168.56.104	Modbus/TC	111	Response: Trans: 141; Unit: 7, Func: 3: Read Holding Registers
192.168.56.104	192.168.56.102	TCP	66	51488 > asa-appl-proto [ACK] Seq=13 Ack=46 Win=229 Len=0 TSval=15844995 TSecr=1956
192.168.56.104	192.168.56.102	UDP	150	Source port: 9912 Destination port: 9913
192.168.56.102	192.168.56.104	UDP	134	Source port: 9913 Destination port: 9912
192.168.56.104	192.168.56.102	UDP	150	Source port: 9912 Destination port: 9913
192.168.56.102	192.168.56.104	UDP	134	Source port: 9913 Destination port: 9912
192.168.56.104	192.168.56.102	UDP	150	Source port: 9912 Destination port: 9913
192.168.56.102	192.168.56.104	UDP	134	Source port: 9913 Destination port: 9912
192.168.56.104	192.168.56.102	UDP	150	Source port: 9912 Destination port: 9913
192.168.56.102	192.168.56.104	UDP	134	Source port: 9913 Destination port: 9912
192.168.56.104	192.168.56.102	Modbus/TC	99	Query: Trans: 142; Unit: 7, Func: 16: Write Multiple Registers
192.168.56.102	192.168.56.104	Modbus/TC	78	Response: Trans: 142; Unit: 7, Func: 16: Write Multiple Registers
192.168.56.104	192.168.56.102	TCP	66	51488 > asa-appl-proto [ACK] Seq=46 Ack=58 Win=229 Len=0 TSval=15845203 TSecr=1956

Figure 4.3

Communication Capture

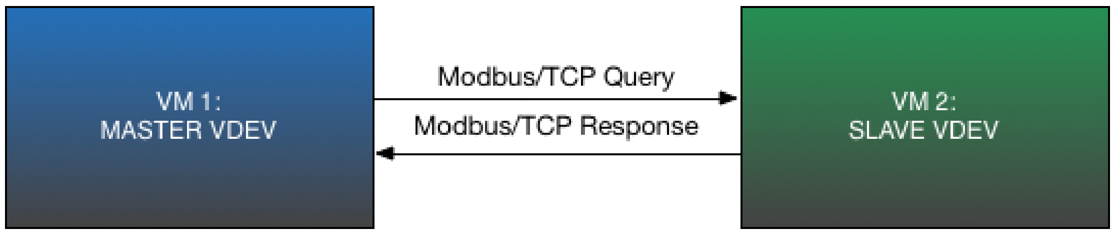


Figure 4.4

Normal Network Communication

pressed followed by the control gates in the VHPS opening to the breakaway position and closing to the SNL position.

At this point, the system is ready to begin generating power and opens the gates back up to increase water flow in the penstock and speed the turbine up to 95% maximum speed. Figure 4.7 shows the turbine speeding up to 95% maximum speed. The values in Figure 4.7 are normalized. Since the normalized values of the control gate percentage, turbine speed, and water flow in penstock are linearly correlated, they are very close in value and overlap on the graph; therefore, the control gate percentage and water flow are excluded from the figure.

Figure 4.8 shows the final stages of the start-up process. For the VHPS, the exciter and the synchronizer are represented as ON or OFF flags. After the turbine speeds up to 95% maximum speed, the exciter turns on. Very soon after the exciter turns on, the synchronizer turns on. After the synchronizer adjusts the terminal voltage and terminal frequency to match the grid, the breaker closes. With the breaker closed, the synchronizer is no longer needed and turns off.

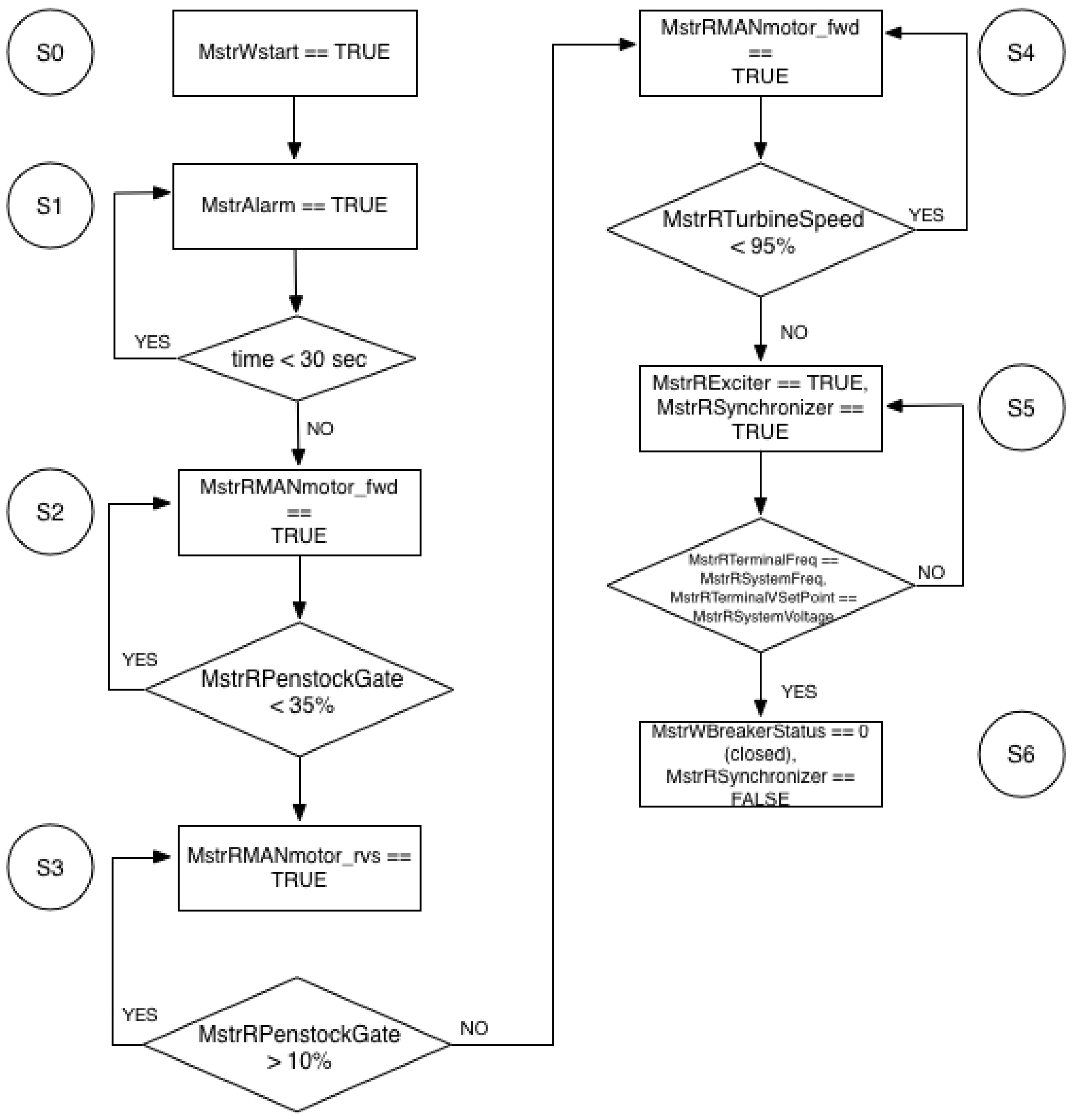

S0
MstrWstart == TRUE
S1
MstrAlarm == TRUE
YES
time < 30 sec
NO
S2
MstrRMANmotor_fwd == TRUE
YES
MstrRPenstockGate < 35%
S3
MstrRMANmotor_rvs == TRUE
YES
MstrRPenstockGate > 10%
NO
MstrRMANmotor_fwd == TRUE
S4
MstrRTurbineSpeed < 95%
YES
NO
MstrRExciter == TRUE, MstrRSynchronizer == TRUE
S5
MstrRTerminalFreq == MstrRSystemFreq, MstrRTerminalVSetPoint == MstrRSystemVoltage
NO
YES
MstrWBreakerStatus == 0 (closed), MstrRSynchronizer == FALSE
S6

Figure 4.5

Start-up ASM

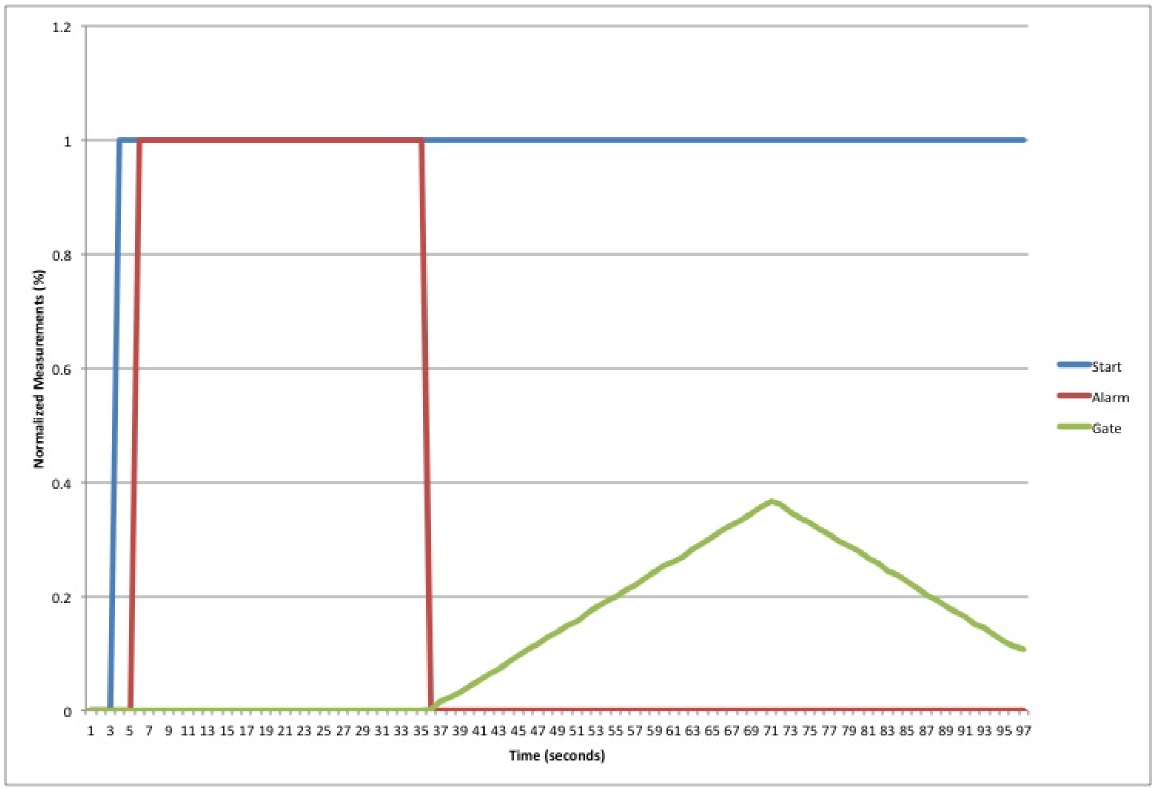


Figure 4.6

Breakaway Position to SNL Position

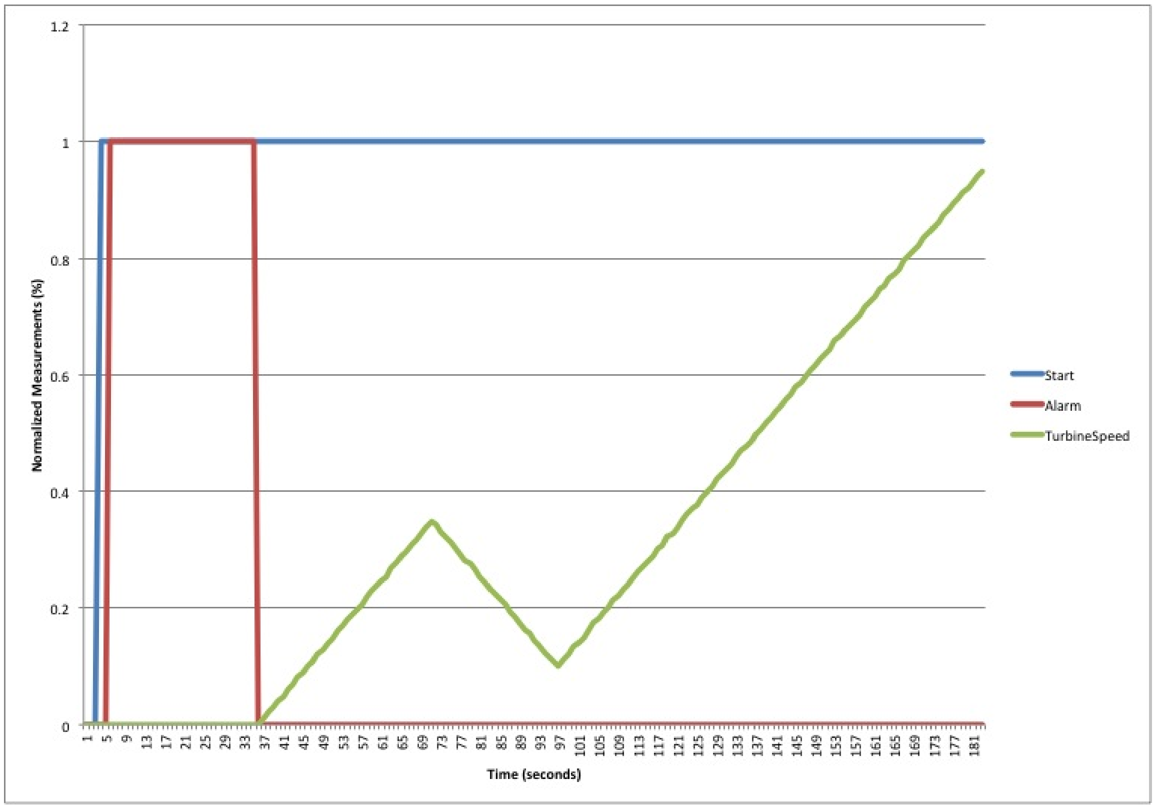


Figure 4.7

Turbine Speed Reaching 95% Maximum

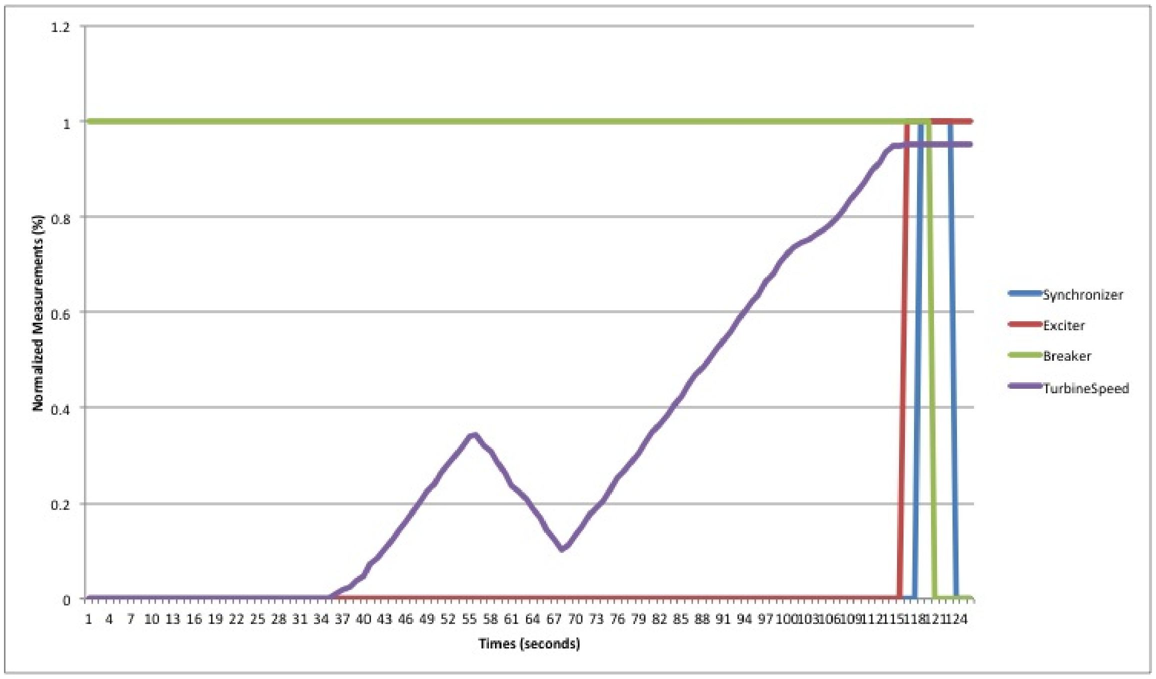


Figure 4.8

Final Stage of Start-up Process

4.4.2 Power Generation

In hydroelectric power systems, the system operator receives a schedule for how much power the facility is expected to produce for given times during the day. The system is expected to generate more power at peak load times than at non-peak load times. The operator can either manually adjust the power output or allow for an AGC system to regulate power output. AGC has not been modeled for the VHPS. The VHPS assumes peak load times at the final stage of the start-up process.

An important function of the VHPS is that the operator can change the amount of power that needs to be generated at any given time. Whether the amount of power needed changes because of the daily schedule or because of an unexpectedly high load at non-peak times, this capability is vital. Figure 4.9 shows the ability of the VHPS to change the amount of power generated by the process.

4.4.3 Shutdown Process

Figure 4.10 provides an ASM for the shutdown process. First, the breaker opens to disconnect the VHPS from the power grid. Next, the exciter is turned off and the control gates begin to close. The closing of the control gates slows down the turbine and the flow rate of water within the penstock. Once the control gates are completely shut and all the water has drained from the penstock, the system is officially offline. Figure 4.11 shows the process of the VHPS shutting down. The turbine speed and water flow rate are not shown in Figure 4.11 because they overlap with control gate percentage when normalized.

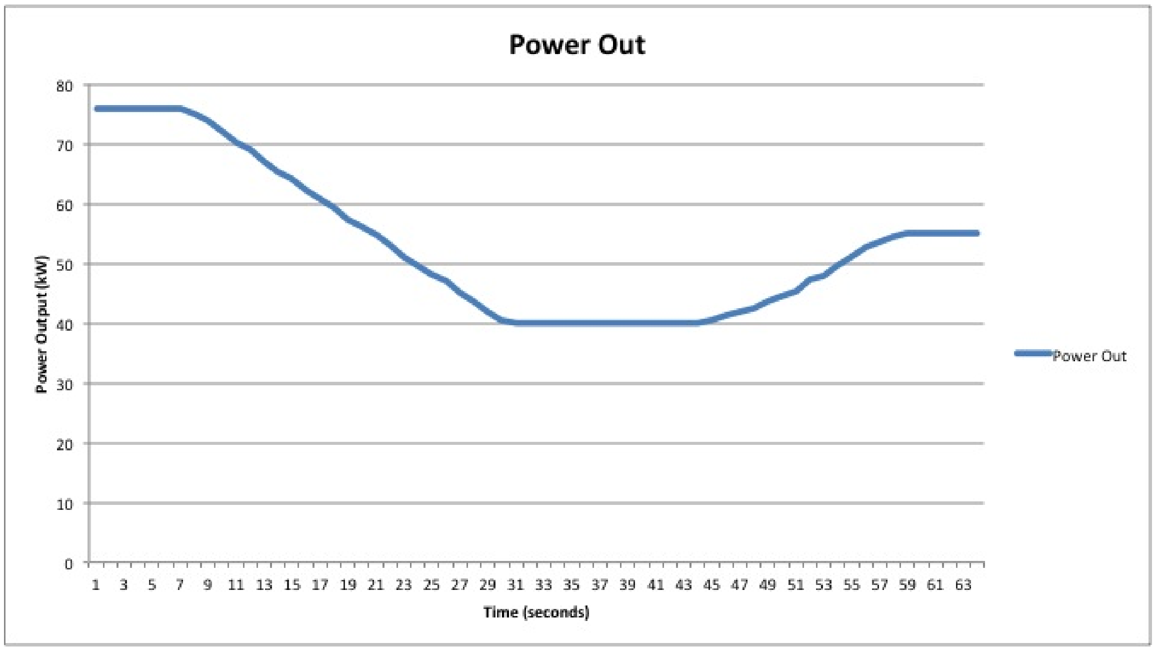


Figure 4.9

Power Generation Changes

4.4.4 GUI

One of the requirements of the VHPS is the provision of a simple GUI for increased readability and usability. In SCADA systems, the system operator monitors and controls the systems through the HMI. For the VHPS, a simple GUI was created to represent the HMI. The GUI was developed using the Python programming language's Tkinter toolkit. Without the GUI, the VHPS would simply be text-based. This hinders the readability and usability of the system. The operator of the VHPS would have to physically type in commands to change input Point values. The operator would also have to continuously type in a specific command to see updated system measurements. The GUI makes the VHPS usable for non-experts, and also makes the VHPS useful in the classroom.

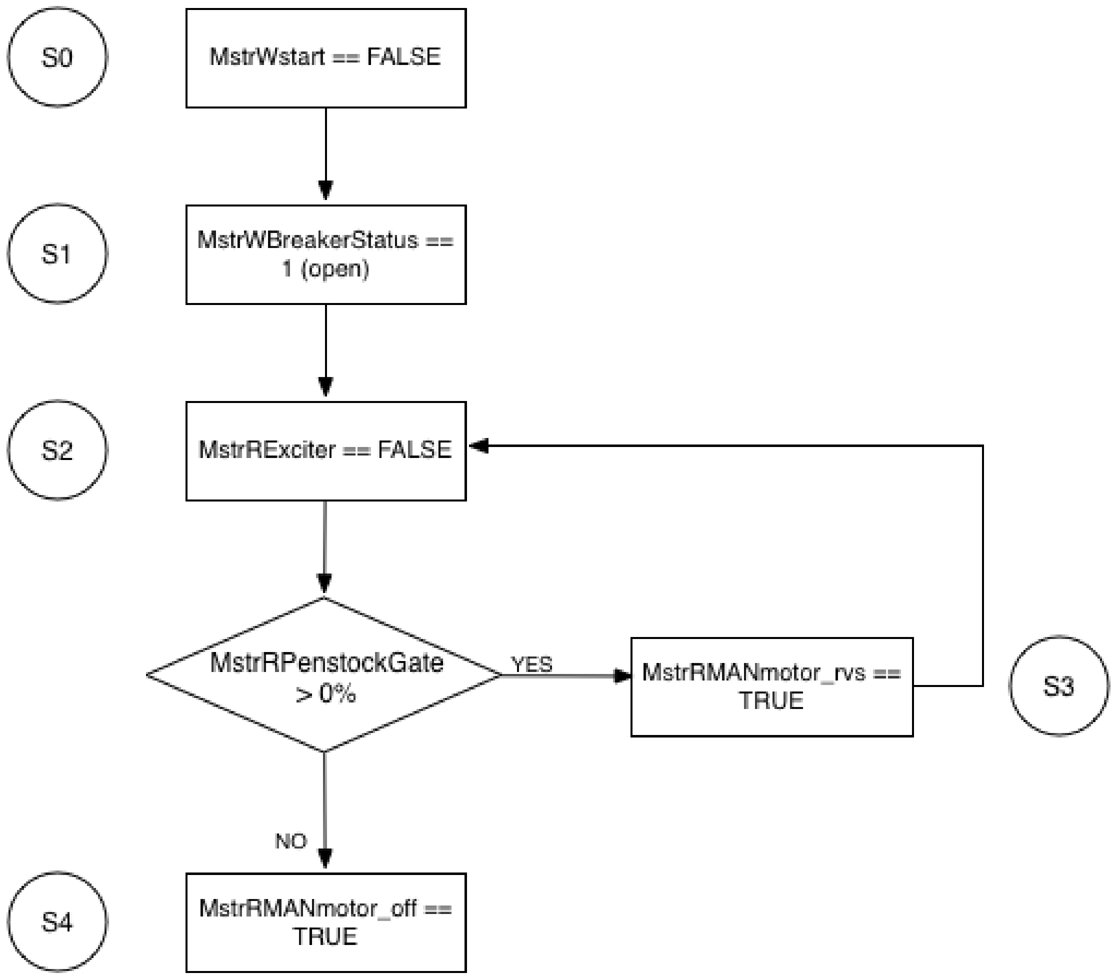

S0
MstrWstart == FALSE
S1
MstrWBreakerStatus == 1 (open)
S2
MstrRExciter == FALSE
MstrRPenstockGate > 0%
YES
MstrRMANmotor_rvs == TRUE
S3
NO
S4
MstrRMANmotor_off == TRUE

Figure 4.10

Shutdown ASM

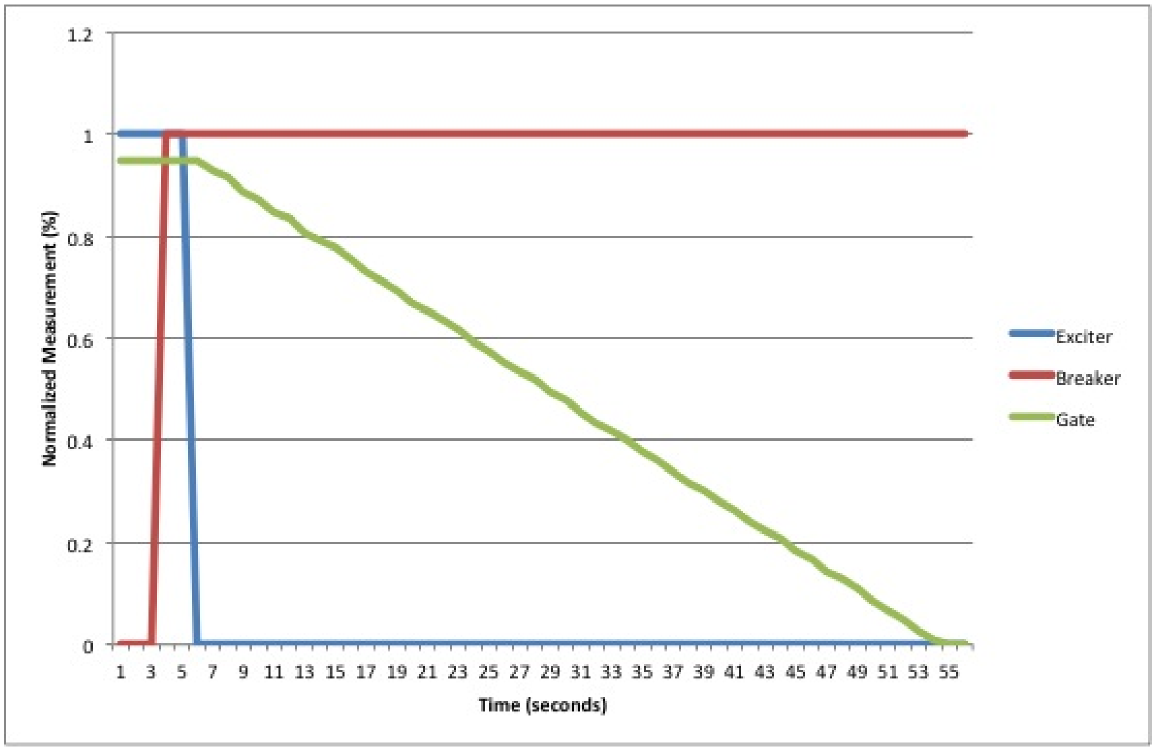


Figure 4.11

VHPS Shutdown Process

Figure 4.12 illustrates the virtual architecture with the addition of the GUI. In the virtual environment, the vdevs are executed as threads. The thread definitions are located in a file names ‘vdev.py’. This file allows for the addition of threads and corresponding definitions to the list of threads. It also provides the definition for a purely text-based user interface. The VHPS GUI is written as a thread definition within this file and is set to only execute when the Master vdev executes. The ‘vdev.py’ file extracts the Points from either vdev into a local ‘points’ list. Through this local list, the GUI writes and reads Point values to and from the Master vdev.

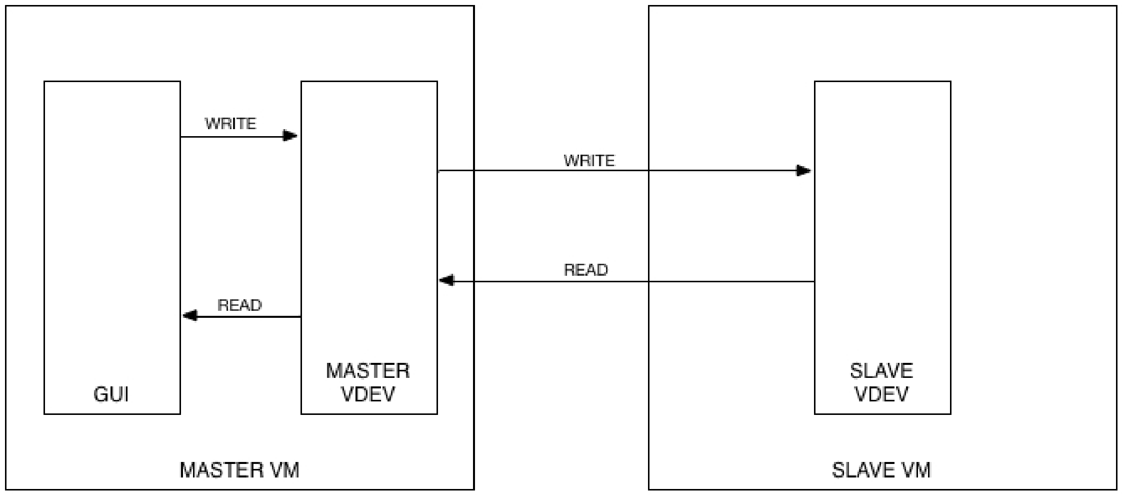
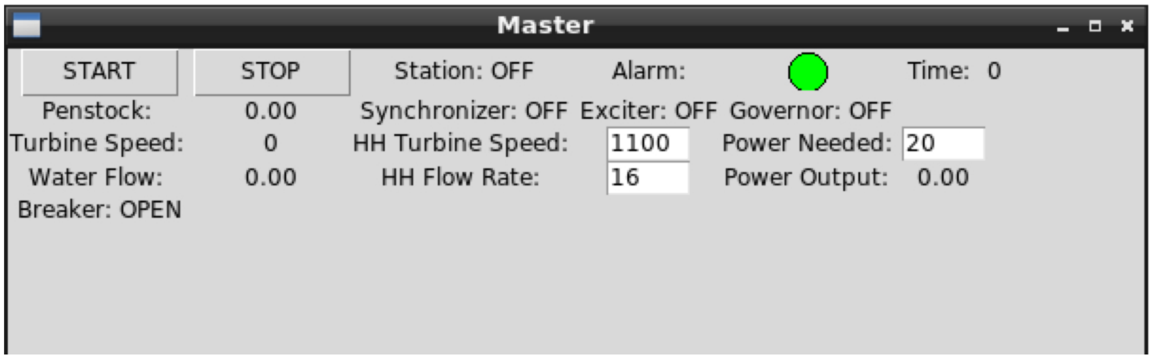


Figure 4.12

GUI Architecture

The provided GUI satisfies the requirement of a simple GUI that represents an HMI. The GUI interfaces with the Master vdev. Control of the input Points of the system can be accomplished through the GUI, and the system measurements are displayed on the GUI without the operator having to manually refresh a Points list. This allows for the operator to easily monitor the system. Figure 4.13 shows the simple GUI used to interface with the VHPS. From the GUI, the operator can start or stop the system as well as change certain system variables such as the amount of power that is needed.



Station: OFF
Alarm: 🟢
Time: 0

Penstock:	0.00	Synchronizer: OFF	Exciter: OFF	Governor: OFF	
Turbine Speed:	0	HH Turbine Speed:	1100	Power Needed:	20
Water Flow:	0.00	HH Flow Rate:	16	Power Output:	0.00
Breaker: OPEN					

Figure 4.13

VHPS GUI

4.5 Capturing Communication

A network interface controller (NIC) is a computer hardware component that connects a computer to a network. A NIC can either operate in non-promiscuous mode or promiscuous mode. In non-promiscuous mode, the NIC only passes frames it is intended to receive to the OS network stack. In promiscuous mode, the NIC passes all frames it receives to the OS network stack regardless of the intended receiver. However, for promiscuous mode to work for monitoring network traffic, the router needs to be a hub or a managed switch with port mirroring capabilities. In an unmanaged switched network, promiscuous mode does not allow for one to monitor all network traffic.

To satisfy the requirement of capturing network traffic, promiscuous mode is enabled for the VMs. The host-only network interface is a virtual hub and, therefore, allows for any connected VM in promiscuous mode to view all network traffic for any VM connected to the interface. The host-only network interface provides two exclusive options for promiscuous mode: Allow All and Allow VMs. With the Allow All option selected, the VMs can sniff all traffic between the VMs and all traffic from physical machines as well. With the Allow VMs option selected, the VMs can only sniff traffic between the VMs connected to the host-only network.

The packet analyzer Wireshark is used to view the communication stream between the Master and Slave virtual devices. It recognizes the Modbus/TCP protocol and allows for easy interpretation of the packets. Additionally, the packet capture (PCAP) library can be used to capture the Modbus/TCP packets and extract important measurements for dataset creation.

4.6 Distribution Packaging

For the requirement of making the VHPS distributable and easily obtainable, various variables must be considered. The VHPS needs at least two VMs (one for the Master vdev and one for the Slave vdev) in order to generate realistic traffic between the devices. Running more than one VM on some less powerful host machines can cause a lot of system strain and lag. The VHPS itself does not need a lot of processing power; therefore, the operating system (OS) chosen for the VMs that the virtual environment and VHPS are housed on should also be lightweight in terms of resource usage.

Fedora is a popular Linux distribution and was used to build the virtual environment. Therefore, the initialization file used to set up the virtual environment is formated to install Fedora development packages. However, less powerful host machines have trouble running more than one VM with the standard Fedora OS. As a result, the OS chosen for the VMs is Fedora LXDE Spin 32-bit. LXDE stands for Lightweight X11 Desktop Environment. It is an alternative version of Fedora that is not designed to be powerful but, instead, lightweight on resource usage and fast. The use of the LXDE spin allows for efficient use of the VHPS on computers with various hardware specifications. This feature increases the range of researchers that can effectively use to system.

The other variable that needs to be considered is packaging size. The packaging should contain everything necessary to get the system up and running as quickly as possible. The packaging size should be smaller than 4GB. The file size limit was chosen because a majority of flash drives are formatted with FAT32. One of the limitations of FAT32 is the file size limit of 4GB. Keeping the packaging size under 4GB circumvents the need to reformat flash drives for sharing purposes.

The simplest and most effective way to package the VHPS and the virtual environment is to use VirtualBox's 'Export Appliance' feature. This feature exports a selected VM as an Open Virtualization Format (OVF) file. In VirtualBox the default extension when exporting is OVA. OVA stands for Open Virtualization Format Archive, and the use of this extension creates a single file. If the OVF extension is chosen instead of OVA, the result will be a series of files. For the VHPS, the OVA file extension is used in order to create a single distributable file. When the Master VM is exported to an OVA file, its file size is roughly 1.73GB. This is well below the 4GB limit. The Slave VM is simply a linked clone of the Master VM, so it does not need to be exported.

4.7 Extendable

The VHPS is a simple representation of a hydroelectric power system. The VHPS consists of a single generator while real system contain many. The VHPS also lacks the multiple Slave devices that are present in real hydroelectric power systems. As a result, the VHPS must be extendable.

Since the VHPS is built in the open virtual environment discussed in [24], additional virtual devices can be added to the VHPS. Only the configuration information and the logic of the devices would need to be added to the current VHPS. Additionally, the use of a host-only network interface in VirtualBox allows for any added devices to have their own unique IP address when executed in separate VMs.

CHAPTER 5

ATTACKS

An important aspect of ICS cyber security research is the ability to implement and deploy cyber attacks. Cyber attacks not only test the vulnerability of the system, but also allow for the develop of training datasets. This chapter outlines how the requirement of deploying cyber attacks is satisfied. Four attacks were implemented: a Denial-of-Service (DoS) attack, session hijacking, function code scan, and setpoint manipulation. It also details how the ability to capture network communication is satisfied.

5.1 DoS

In some instances, an attacker may wish to make a server or other resources unavailable to its clients. An attack that successfully prevents clients from accessing needed services is known as a DoS attack.

With the current TCP sequence number and Modbus Transaction Identifier known, the attacks can now be injected into the communication stream. However, a race condition is now created. If a normal packet reaches the destination before the attack packet, the attack packet will be ignored. To prevent the race condition and to ensure that all of the attacker's packets reach the destination first, a DoS attack must be performed against the source device. The DoS attack eliminates the race condition and ensures the attack packet is accepted.

Figure 5.1 illustrates the methodology used to get the source device out of sequence. The simplest way to perform a DoS attack against the source is to get it out of sequence with the destination. Theoretically, one way of accomplishing this is to send a packet to the destination immediately after predicting what the next sequence number should be valued. This causes the destination to ignore the real source's packets because they will have a different sequence number than what is expected. Since the system is a repeated process of reads followed by writes, the PDU size sent to get the source device out of sequence does not need to be arbitrary.

In Figure 5.1, the attacker successfully removes the Master virtual device from sequence with the Slave virtual device. The packet with the Modbus Transaction Identifier of 255, sequence number of 259, and FC of 0x03 is captured for sequence number prediction purposes. The attacker fabricates a new Modbus/TCP packet with the Modbus Transaction Identifier incremented by one and the sequence number incremented by 12. The sequence number is incremented by 12 because the PDU size of queries with a FC of 0x03 for the VHPS is 12. To ensure that the FC used in the fabricated packet is valid for the Slave virtual device, the fabricated packet is essentially a replay of the captured packet with modified Modbus Transaction Identifier and TCP sequence number. Once the fabricated packet is assembled, it is immediately sent to the Slave virtual device. In Figure 5.1, the fabricated packet is in bold and directed towards the Slave virtual device with the values FC equal to 0x03, sequence number equal to 271, and transaction identifier equal to 256.

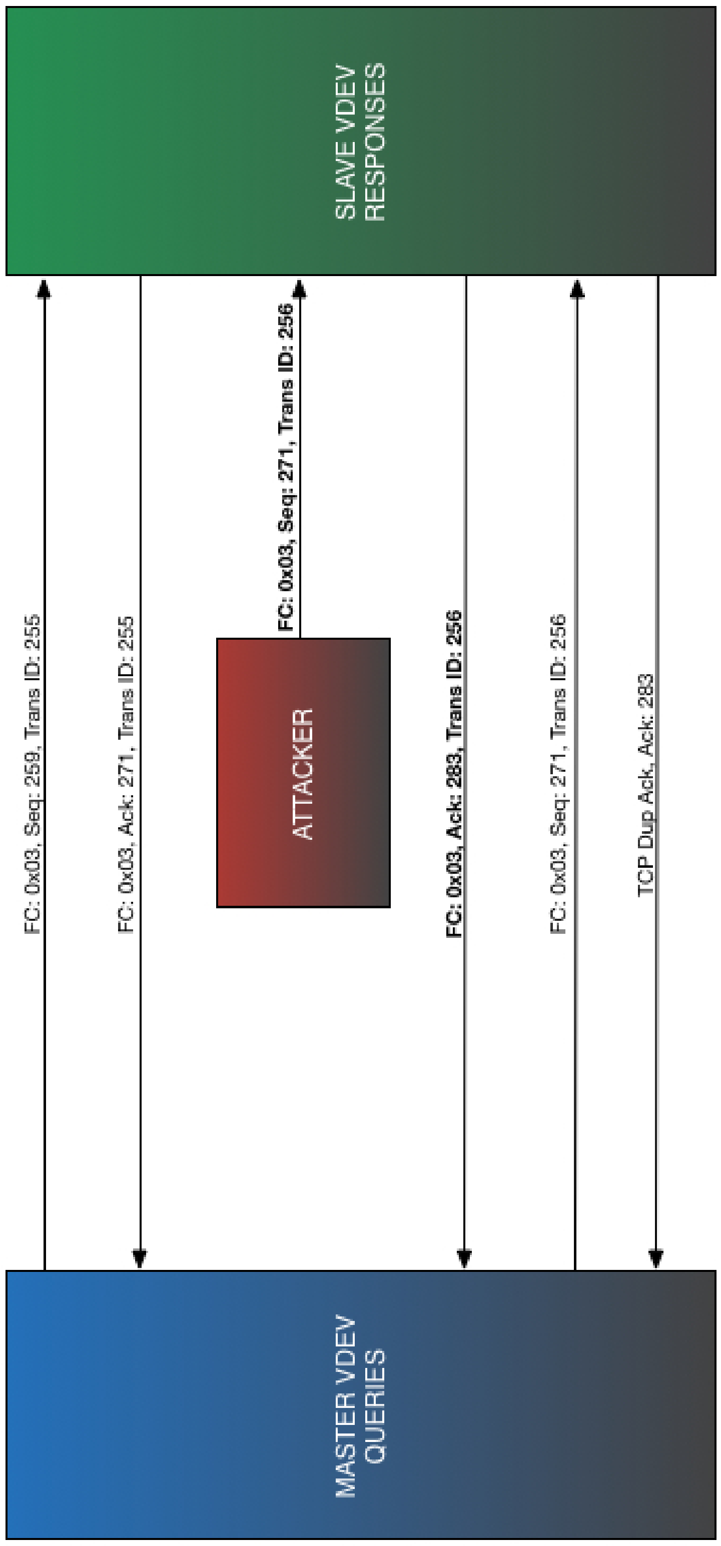

MASTER VDEV QUERIES
SLAVE VDEV RESPONSES
ATTACKER
FC: 0x03, Seq: 259, Trans ID: 255
FC: 0x03, Ack: 271, Trans ID: 255
FC: 0x03, Seq: 271, Trans ID: 256
FC: 0x03, Ack: 283, Trans ID: 256
FC: 0x03, Seq: 271, Trans ID: 256
TCP Dup Ack, Ack: 283

Figure 5.1

Denial of Service

Now, when the Master vdev sends out its packet with the sequence number equal to 271, the Slave vdev ignores it and responds with duplicate packet notification. The Master vdev will continue to send the same packet again and again and each time it will be ignored. The attacker is now free of the race condition.

5.2 Session Hijacking

For the VHPS, the Master and the Slave virtual devices communicate using the industry standard Modbus/TCP protocol. In order to inject packets into the communication stream from a third party, the TCP session must be hijacked. Session hijacking involves three steps: the ability to monitor and capture network activity, the ability to perform a TCP sequence prediction attack, and the ability to perform a Denial-of-Service (DoS) attack. This method of attacking has not previously been implemented for the virtual SCADA platform and demonstrates a realistic attack against ICS. Session hijacking allows an attacker to inject both Modbus queries and responses. This breaks the feedback control loop in two ways. An attacker can inject malicious control commands or malicious sensor measurements.

5.2.1 Monitoring and Capturing Network Activity

In order to hijack the session, the attacker must have the ability to both monitor and capture network traffic between the Master and Slave vdevs. Therefore, the NIC must be in promiscuous mode. Similar to the Master and Slave vdevs, the attacker exists as a VM connected to VirtualBox host-only network interface. The use of promiscuous mode on a VM means that any VM connected to the host-only network can capture packets and log the data. As a result, the sub-requirement of ability to capture network traffic during a cyber attack is satisfied.

Wireshark is used to view the communication stream between the Master and Slave virtual devices. It recognizes the Modbus/TCP protocol and allows for easy interpretation of the packets. For this implementation of session hijacking, the most important piece of information are the TCP port numbers used by the virtual devices. The port numbers are essential for the next step: TCP sequence number prediction.

5.2.2 TCP Sequence Prediction

The Transmission Control Protocol (TCP) is one of the foundational protocols in the TCP/IP model of network communication. The purpose of the TCP protocol is to offer reliable transmission of data between programs on computers communicating over the network. The computer sending the data is known as the source, and the computer receiving the data is known as the destination. TCP also ensures that the data is delivered in the order it is meant to be delivered. To accomplish this, TCP headers include a 32-bit sequence number parameter. At the beginning of data transmission, a sequence number is randomly chosen by the source computer. All subsequent sequence numbers after the first increase by the amount of data that is sent to the destination. If a received sequence number is incorrect, the destination responds to the source with a ‘TCP Out-of-Order’ packet.

The goal of the attacker is to inject falsified commands or responses into the communication stream between the Master and Slave virtual devices. Without knowing the correct, expected TCP sequence number, all of the attacker’s false Modbus/TCP packets would be ignored. With Wireshark, the current sequence number can be viewed, and through packet analysis, a pattern for the next expected sequence number can be determined.

In Modbus/TCP, the sequence number increases by the size of the Modbus PDU. The Modbus PDU size does not fluctuate very much due to the nature of ICS's. They rarely change. They are repeated processes that have a constant set of points. For the VHPS, the system is a repeated series of read commands followed by write commands. Specifically, the read commands are Modbus function code 0x03 Read Holding Registers, and the write commands are Modbus function code 0x10 Write Multiple Registers. For the VHPS, the repeated process and constant set of points lead to a PDU size of 12 bytes for Modbus command Read Holding Registers and a PDU size of 33 bytes for Modbus command Write Multiple Registers. The repeated process of read commands followed by write commands simplifies sequence number prediction.

Figure 5.2 provides a graph of the communication stream for the following example. If the Master virtual device sends a read command with a sequence number of 100, then the PDU size will be 12 bytes. The next command will be a write command with a sequence number of 112 and a PDU size of 33 bytes. Since the read and write commands alternate, the third command in this example will be a read command. This packet will have a sequence number of 145 and a PDU size of 12 bytes.

The Packet Capture (PCAP) library is used to capture one of the Modbus packets. As the name suggests, the PCAP library is made up of an Application Programming Interface (API) for capturing network packets. Once a packet is captured, the TCP sequence number and the Modbus Transaction Identifier are extracted. The purpose of the Modbus

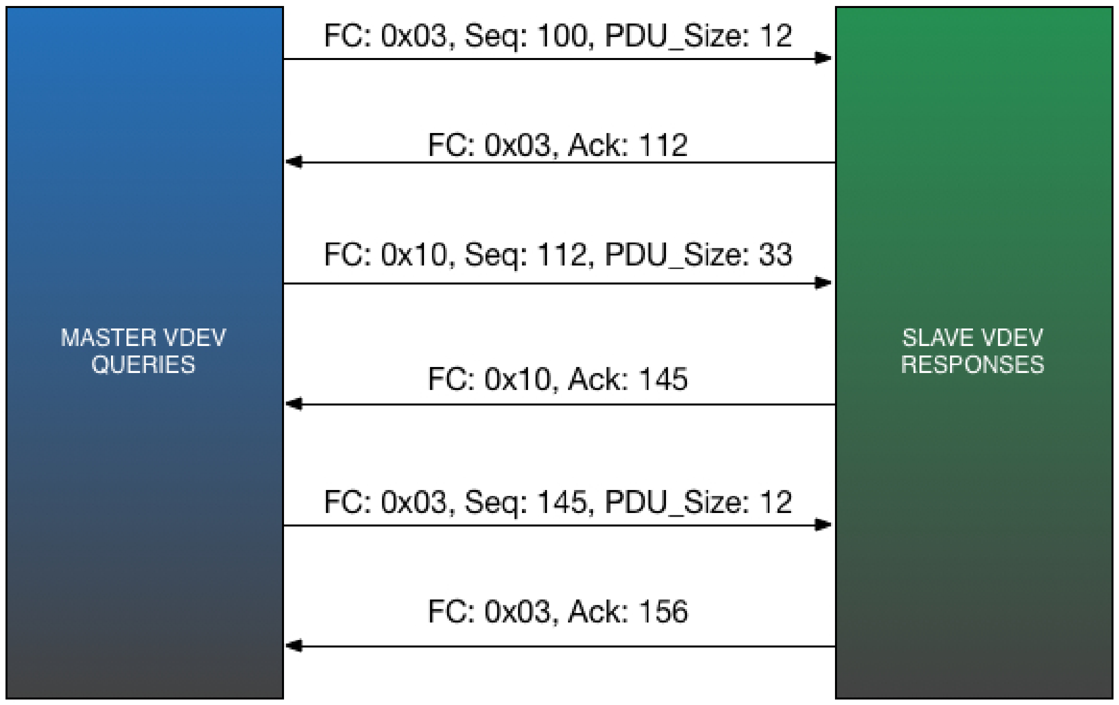

MASTER VDEV QUERIES
FC: 0x03, Seq: 100, PDU_Size: 12
FC: 0x03, Ack: 112
FC: 0x10, Seq: 112, PDU_Size: 33
FC: 0x10, Ack: 145
FC: 0x03, Seq: 145, PDU_Size: 12
FC: 0x03, Ack: 156
SLAVE VDEV RESPONSES

Figure 5.2

TCP Sequence Number Example

Transaction Identifier is to keep the command order. It is simply an integer value that is incremented by one with every command. These two elements are necessary for the final phase of session hijacking: the DoS attack.

5.2.3 DoS for Session Hijacking

The last step needed to complete the session hijacking is the DoS attack. The purpose of the DoS attack is to eliminate the race condition between the source vdev and the attacker. The DoS attack described earlier is used to enable this attack.

5.3 Function Code Scan

Gathering information about a system is often one of the primary incentives when attempting to attack the system. Reconnaissance attacks tell the attacker valuable information about the system. For example, an attacker can glean IP addresses of servers and clients for systems that use Modbus/TCP, valid function codes used by the Modbus servers, or specific points used by the Modbus servers to store register values. Reconnaissance attacks are vital in developing more complex and targeted attacks.

The purpose of the function code scan attack is to identify supported function codes for a Modbus server. The Modbus function code field has a single byte length. For example, the Modbus function code for 'Read Holding Registers' is 0x03. Performing a function code scan allows for the attacker to realize various other attacks by exploiting the knowledge of supported function codes.

The attacker is able to differentiate between supported and unsupported function codes by examining the response from the Slave device. For any given query, the data field of the response will either contain normal response data or an error message. In Modbus, error messages are comprised of two bytes. The first byte is called the error code. The error code is merely the function code added to 0x80. For example, if the function code that produces the error is 0x03, the returned error code will be 0x83. The second byte is the exception code. The exception code indicates the general nature of the error. Table 5.1 lists the Modbus exception codes and their names [15].

Table 5.1

Modbus Exception Codes

Exception Code	Name
0x01	Illegal Function
0x02	Illegal Data Address
0x03	Illegal Data Value
0x04	Server Device Failure
0x05	Acknowledge
0x06	Server Device Busy
0x08	Memory Parity Error
0x0A	Gateway Path Unavailable
0x0B	Gateway Target Device Failed to Respond

Figure 5.3 shows the state diagram for Modbus FC 0x03 'Read Holding Registers'. In all Modbus FC state diagrams, the first decision the server makes is whether the FC is supported or not. If the FC is not supported, the server returns an error message with exception code 0x01. A response containing either normal data or any exception code that is not 0x01 indicates that the FC is supported [15].

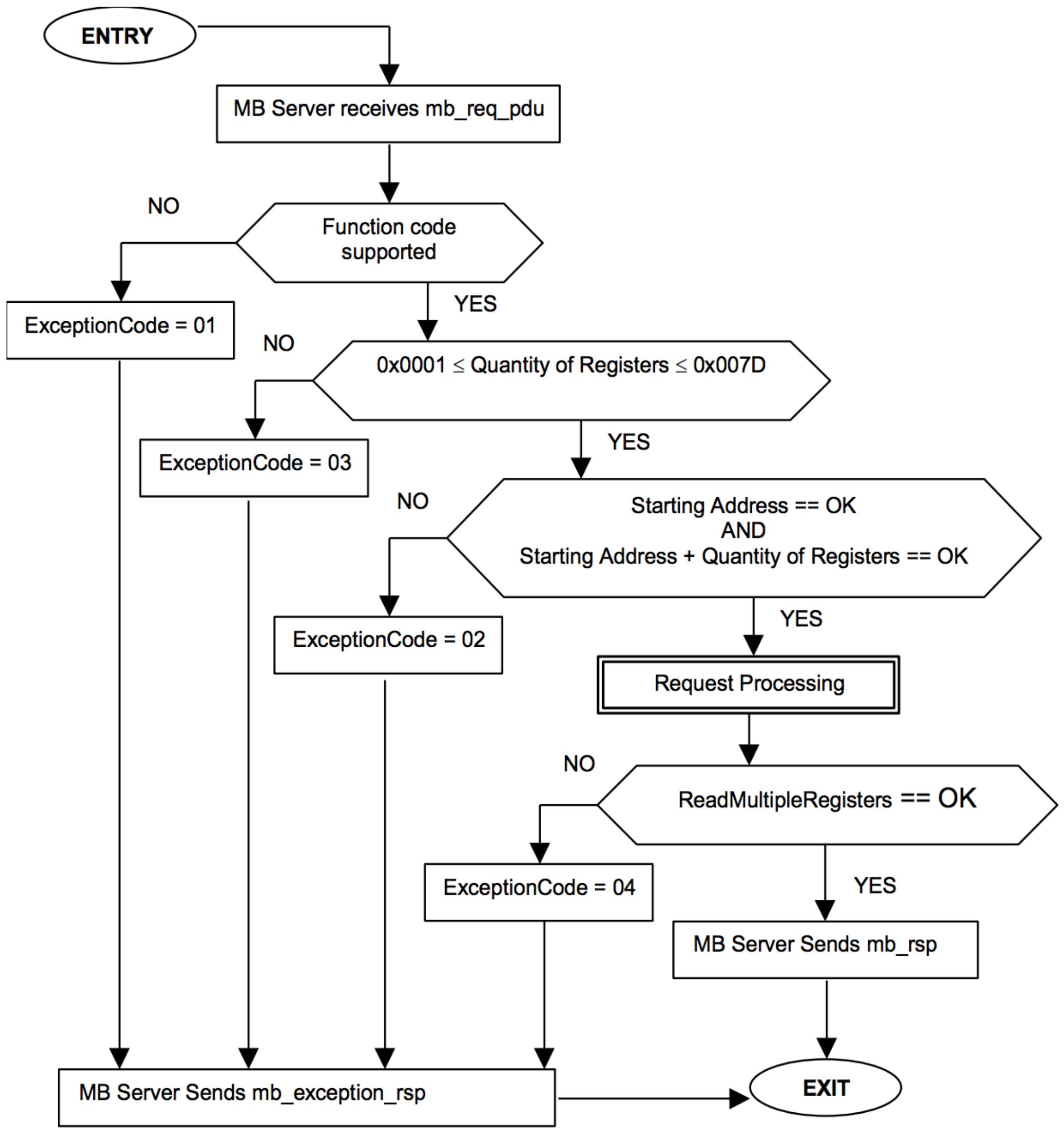

ENTRY
MB Server receives mb_req_pdu
Function code supported
NO
YES
ExceptionCode = 01
0x0001 ≤ Quantity of Registers ≤ 0x007D
NO
YES
ExceptionCode = 03
Starting Address == OK AND Starting Address + Quantity of Registers == OK
NO
YES
ExceptionCode = 02
Request Processing
ReadMultipleRegisters == OK
NO
YES
ExceptionCode = 04
MB Server Sends mb_rsp
MB Server Sends mb_exception_rsp
EXIT

Figure 5.3

State Diagram for Read Holding Registers [15]

To perform a function code scan, the attacker merely needs to send queries with the varying function codes. The content of the queries does not matter as long as the packets are formed correctly for a specific function code. A simple way to perform the attack is to use the same data for every fabricated packet to be sent but increment the function code value by one inside of a loop.

A Function Code Scan attack was performed against the VHPS. Table 5.2 shows the results of the attack. The support of the function codes is determined through the response from the Slave. If the response from the Slave is not an error message, the FC is supported. If the Slave responds with an error message with exception code 0x01, the function code is not supported. If the Slave returns an error message with an exception code other than 0x01, the function code is supported.

5.4 Setpoint Manipulation

In the VHPS, setpoints are variables in the Slave vdev which are changed by the master. Setpoints allow the Master to control the Slave. By hijacking the session, the attacker becomes in complete control of the system. From the results of the Function Code Scan attack, the attacker knows that the 'Write Multiple Registers' (FC: 0x10) function is supported. Using this function, the attacker can choose to change any or all setpoint values at once.

In the VHPS, there are a number of setpoints the attacker can manipulate in an effort to sabotage the system. By changing the 'Mstrstart' setpoint, the attacker can start or stop the system. If the system is in automatic mode, the attacker can change the 'PowerNeeded'

Table 5.2

Function Code Scan Attack Results

FC	Name	Valid	Exception Returned	Exception Code
0x01	Read Coils	Yes	No	N/A
0x02	Read Discrete Inputs	Yes	No	N/A
0x03	Read Holding Registers	Yes	No	N/A
0x04	Read Input Registers	Yes	Yes	0x02
0x05	Write Single Coil	Yes	Yes	0x03
0x06	Write Single Register	Yes	No	N/A
0x07	Read Exception Status	No	Yes	0x01
0x08	Diagnostics	No	Yes	0x01
0x09	Unknown Function	No	Yes	0x01
0x0A	Unknown Function	No	Yes	0x01
0x0B	Get Comm. Event Counters	No	Yes	0x01
0x0C	Get Comm. Event Log	No	Yes	0x01
0x0D	Unknown Function	No	Yes	0x01
0x0E	Unknown Function	No	Yes	0x01
0x0F	Write Multiple Coils	Yes	Yes	0x04
0x10	Write Multiple Registers	Yes	No	N/A

setpoint to adjust the amount of power the system generates. The attacker is also able to change the system mode from from automatic to manual by changing the ‘MstrSystemMode’ setpoint. After changing to manual mode, the attacker can alter the ‘MastergateSetPoint’ to adjust the positions of the control gates, allowing for more or less water to enter the penstock and affecting the amount of power generated.

After the plant (VHPS) begins to generate power and the voltage and current are synchronous with the grid, a breaker is closed to connect the plant to the system. Using the ‘MstrBreakerStatus’ setpoint, the attacker can disconnect the VHPS from the grid by opening the breaker. System operators are sent schedules of how much power to provide to the grid for various times throughout the day. By disconnecting the plant from the grid, the attacker prevents the plant from supplying the needed power to the grid. Figure 5.4 shows the breaker being opened by the attacker soon after the breaker is closed to connect to the power grid.

The Setpoint Manipulation attack is possible because there is no authentication in Modbus. The Modbus protocol does not check the MAC address of the sender to ensure sender validity.

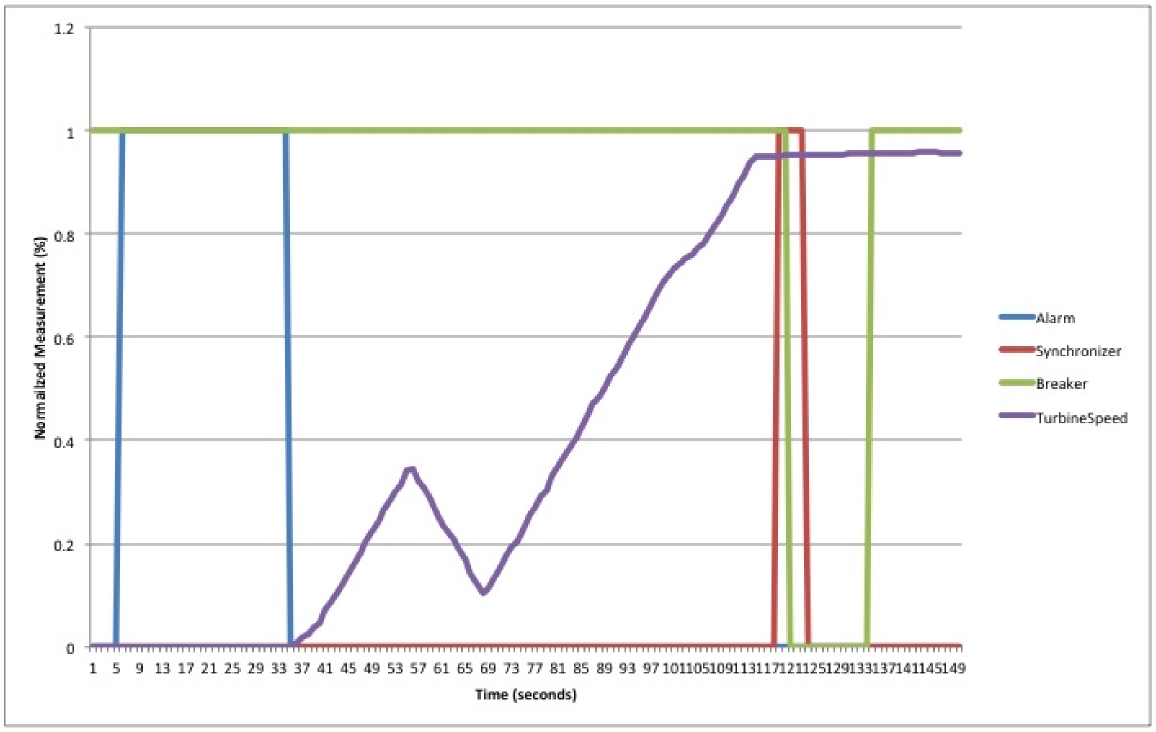


Figure 5.4

Breaker Opened by Attacker

CHAPTER 6

CONCLUSIONS

Currently, cyber security research for industrial control systems suffers from a lack of ability to share methods and results of research. Researchers in this field typically have their own unique method of security testing of ICS. This hampers the ability of researchers to effectively test and compare results. This thesis addresses the following hypothesis in relation to these issues.

It is hypothesized that the following are attainable:

1. It is possible to build a simple, virtual hydroelectric power system in an available virtual environment.
2. It is possible for the virtual hydroelectric power system to generate real network traffic.
3. It is possible to capture the VHPS network traffic.
4. It is possible to build attacks against the virtual system.
5. It is possible to constrain a distributable package of the system to under 4GB.
6. It is possible for the VHPS to be extended.

6.1 Contributions

A VHPS was developed in an available virtual environment [24]. The VHPS follows normal operations for a hydroelectric power system that includes system start-up, power generation, and system shut-down. The VHPS includes two virtual devices known as the

Master and the Slave. The Master vdev sends queries or commands to the Slave vdev. The Slave vdev responds in the appropriate manner to the Master's queries or commands. The VHPS also includes a simulator that represents the mechanical process of opening and closing the control gate. For ease of use, the VHPS also includes a GUI that represents the HMI that would be used by plant operators to monitor and control the system.

The logic for the Master and Slave virtual devices is located and executed on separate VMs in VirtualBox. The VMs are connected to VirtualBox's host-only network interface. Connecting the VMs to this interface allows for the vdevs to have their own unique IP addresses. The vdevs communicate with each other using the Modbus/TCP protocol. Modbus/TCP is an industry standard that follows the Client-Server model. In this case, the Master vdev is the Client, and the Slave vdev is the server. With the combination of each vdev having their own IP address and the vdevs communicating using Modbus/TCP, the VHPS is able to generate real network traffic.

VirtualBox's host-only network interface, to which the VMs are connected, has three promiscuous mode options. The first option is to deny promiscuous mode, which is the same as putting the network interface in non-promiscuous mode. The second option is to enable promiscuous mode with the Allow All option set. The Allow All option means the VMs can view all traffic between each other as well as from physical machines. The third option is to enable promiscuous mode with the Allow VMs option set. This option means the VMs can view all traffic between each other but not from physical machines. For the VHPS setup, promiscuous mode is enabled with the Allow VMs option chosen. As a result, all VMs connected to the host-only interface can view all network traffic between each other. The network analyzer Wireshark is used to monitor and capture network traffic.

Various attacks were developed against the virtual system. A session hijacking method was developed in order to take over control of the VHPS from the Master vdev. The session hijacking attack involves monitoring and capturing Modbus/TCP transactions between the Master and Slave vdevs and executing a TCP sequence prediction attack in preparation for the final step of hijacking the session: a DoS attack. The purpose of the DoS attack is to eliminate the race condition with the Master vdev.

In addition to the session hijacking attack, a function code scan attack and a setpoint manipulation attack were developed. The function code scan attack is a type of reconnaissance attack in which the Slave device is scanned to determine which Modbus function codes are supported and which are not. The setpoint manipulation attack allows for the attacker to change various operational setpoints that are normally manipulated by the plant operator.

The VHPS has been condensed into a single, distributable file that is well below the 4GB constraint. Within VirtualBox, the Master VM was exported to a 1.73GB OVA file. An OVA file is a compressed file containing all files and settings for a VM. It can be imported into VirtualBox on a different machine. This allows for the VHPS to be easily distributed and quickly functioning on other host machines.

The VHPS is a simple model of a hydroelectric power system. The combination of the virtual environment [24], VMs, and the host-only network interface allow for the VHPS to be extended and improved upon.

6.2 Future Work

There are many expansions that can be made on this work in the future. First, the VHPS can be expanded to include multiple generators. With multiple generators, automatic generation control should be implemented [11]. In hydroelectric power systems, the AGC monitors and controls the power output of each generator in the system [12]. Second, a larger catalog of attacks should be developed against the VHPS. The catalog should include more reconnaissance attacks such as a point scan and an address scan. The catalog should also include response injection attacks. The response injection attacks should make the Master vdev believe that is still communicating properly with the Slave vdev after the session hijacking attack. Third, useful datasets should be built for security research purposes. The VMs are connected to a host-only network interface in promiscuous mode, so any VM connect to the same network interface can capture system data to create the datasets. Fourth, current security solutions, such as Snort, should be developed for the VHPS. Another VM running Snort rules as an intrusion detection system could be added to the network of VMs in an effort to detect the malicious activity.

REFERENCES

[1] ASCE, *Water Control Gates: Guidelines for Inspection and Evaluation*, American Society of Civil Engineers, 2012.

[2] D. C. Bergman, *Power Grid Simulation, Evaluation, and Test Framework*, master's thesis, University of Illinois, May 2010.

[3] *Bro - Documentation and Training*, The Bro Project, http://www.bro.org/documentation/index.html, 2014.

[4] J. Caswell, *Survey of Industrial Control Systems Security*, Washington University in St. Louis.

[5] D. Dickinson, *Protecting Water Industrial Control and SCADA Systems from Cyber Attacks*, Phoenix Contact.

[6] S. East, J. Butts, M. Papa, and S. Shenoi, "A Taxonomy of Attacks on the DNP3 Protocol," *Critical Infrastructure Protection*. IFIP, 2009, vol. 3.

[7] N. Falliere, L. Murchu, and E. Chien, *W32.Stuxnet Dossier*, Tech. Rep., Symantec Security Response, 2011.

[8] T. Fleury, H. Khurana, and V. Welch, "Towards A Taxonomy of Attacks Against Energy Control Systems," *International Conference on Critical Infrastructure Protection*. IFIP, 2008.

[9] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, *Critical Infrastructure Protection*, vol. 3, chapter Design and Implementation of a Secure Modbus Protocol, IFIP International Federation for Information, 2009.

[10] I. N. Fovino, M. Masera, L. Guidi, and G. Carpi, "An Experimental Platform for Assessing SCADA Vulnerabilities and Countermeasures in Power Plants," *Human System Interactions*. IEEE, 2010, pp. 679–686.

[11] N. Jaleeli, D. N. Ewart, and L. H. Fink, "Understanding Automatic Generation Control," *IEEE Transactions on Power Systems*. IEEE, 1992, vol. 7, pp. 1106–1122.

[12] G. L. Kusic, J. A. Sutterfield, A. R. Caprez, J. L. Haneline, and B. R. Bergman, "Automatic Generation Control for Hydro Systems," *IEEE Transactions on Energy Conversion*. IEEE, 1988, vol. 3, pp. 33–39.

[13] A. Lemay, J. Fernandez, and S. Knight, "An Isolated Virtual Cluster for SCADA Network Security Research," *Proceedings of the 1st International Symposium for ICS and SCADA Cyber Security Research*, 2013.

[14] *Modbus Messaging On TCP/IP Implementation Guide*, Modbus Organization, October 2006.

[15] *Modbus Application Protocol Specification*, Modbus Organization, April 2012.

[16] T. Morris and W. Gao, *Industrial Control System Network Traffic Data Sets to Facilitate Intrusion Detection System Research*, Mississippi State University.

[17] T. Morris, A. Srivastava, B. Reaves, W. Gao, K. Pavurapu, and R. Reddi, "A Control System Testbed to Validate Critical Infrastructure Protection Concepts," *International Journal of Critical Infrastructure Protection*, 2011, vol. 4, pp. 88–103.

[18] T. H. Morris and W. Gao, "Industrial Control System Cyber Attacks," *Proceedings of the 1st International Symposium for ICS and SCADA Cyber Security Research*, 2013.

[19] *NHA FAQ*, National Hydropower Association, http://www.hydro.org/tech-and-policy/faq/, 2014.

[20] *CIP Standards*, NERC.

[21] *Top 10 Vulnerabilities of Control Systems and Their Associated Mitigations*, NERC, 2007.

[22] *NERC*, NERC, http://www.nerc.com/Pages/default.aspx, 2013.

[23] H. Perlman, *Hydroelectric Power: How It Works*, U.S. Geological Survey, http://water.usgs.gov/edu/hyhowworks.html, 2014.

[24] B. Reaves, *An Open Virtual Testbed for Industrial Control System Security Research*, master's thesis, Mississippi State University, 2011.

[25] J. Singh and M. J. Nene, "A Survey on Machine Learning Techniques for Intrusion Detection Systems," *International Journal of Advanced Research in Computer and Communication Engineering*, November 2013, vol. 2.

[26] J. Slay and M. Miller, *Critical Infrastructure Protection*, chapter Lessons Learned from the Maroochy Water Breach, Springer US, 2008.

[27] *Snort FAQ/Wiki*, Sourcefire, https://github.com/vrtadmin/snort-faq/blob/master/README.md, June 2013.

[28] *Types of Hydropower Plants*, U.S. Department of Energy, http://energy.gov/eere/water/types-hydropower-plants.

[29] B. Zhu, A. Joseph, and S. Sastry, *A Taxonomy of Cyber Attacks on SCADA Systems*, Tech. Rep., University of California at Berkeley, 2011.