Virtual SCADA Systems for Cyber Security

Zach Thornton
Distributed Analytics and Security Institute (DASI)
Mississippi State University
Mississippi State, MS, USA
jzt3@msstate.edu

David Mudd
Distributed Analytics and Security Institute (DASI)
Mississippi State University
Mississippi State, MS, USA
dbm157@msstate.edu

Dr. Thomas Morris
Distributed Analytics and Security Institute (DASI)
Mississippi State University
Mississippi State, MS, USA
morris@ece.msstate.edu

Dr. Fei Hu
Electrical and Computer Engineering
University of Alabama
Tuscaloosa, AL, USA
fei@eng.ua.edu

Abstract—This paper describes a pair of virtual Supervisory Control and Data Acquisition (SCADA) systems. These virtual simulations were built using virtual devices that simulate industrial processes, emulate control system ladder logic functionality, utilize control system communication protocols, and implement industrial Human Machine Interfaces (HMI). The first of these focuses on a comprehensive virtual SCADA Laboratory using a gas pipeline simulation for research and teaching. The second details a procedure of exploiting the non-existent authentication methodology in Modbus/TCP to both hijack control and attack a virtual hydroelectric power system (VHPS).

Keywords; Human Machine Interface (HMI), Industrial Control System(ICS), Intrusion Detection System (IDS), MODBUS, Programmable Logic Controller (PLC), Supervisory Control and Data Acquisition (SCADA), Cyber security

I. INTRODUCTION

Industrial Control Systems, also known as Supervisory Control and Data Acquisition (SCADA) systems, are integral critical infrastructure for vital industries such as electricity, oil and gas, water, transportation, and chemical manufacture. Cyber security research has highlighted fundamental risks associated with SCADA systems and cyber security defensive techniques and countermeasures are required [1].

Fundamental risks in SCADA SYSTEMS can be identified and detected with research into the patterns, attack vectors, and impacts related to malicious activity. Traditionally, such research has required a test bed environment that includes a scaled physical model and the accompanying hardware, software, and information communications technologies (ICT) which form the complete cyber physical system. Such an environment presents two limitations for researchers. First, only researchers with access to the test bed environment can engage in SCADA intrusion detection research. Second, such test bed environments are expensive, difficult to expand, and difficult to maintain.

This paper describes 2 systems developed in response to these difficulties; the first is a virtual SCADA laboratory that is portable, distributable, and expandable. This environment closely models commercial SCADA products, is able to communicate with commercial SCADA products, can easily be expanded, is run in a virtual computing environment, and was developed in conjunction with classroom laboratory exercises to facilitate teaching laboratory concepts.

The second is a Virtual Hydroelectric Power System. Of primary focus is the exploitation of the Modbus/TCP protocol to implement and deploy attacks against the VHPS. The chief contribution of this work is to presents a conglomeration of previous SCADA research components into entire virtual SCADA systems that can be used for cyber security attack and defense research.

This paper described related work in this field, gives an outline of the virtual SCADA laboratory, attacks against it, and uses for the lab, describes the VHPS system, attacks against the VHPS, and results of the attacks.

II. BACKGROUND AND RELATED WORKS

Most modern industrial processes, such as electricity, oil and gas, water, transportation, chemical manufacture, etc., are controlled by SCADA SYSTEMS. These systems are made of numerous components, which can be broken down into 4 major categories. First, at the lowest level are sensors and actuators. Sensors include meters, gauges, calipers, and transmitters. Sensors are transducers that convert physical phenomenon into an electrical signal. Actuators, such as pumps, valves, and motors, receive a control signal and manipulate the physical process. Second, there are the controllers themselves. Distributed controllers include programmable logic controllers (PLC), programmable automation controllers (PAC), and intelligent electronic devices (IED). These are special purpose computer systems that interface with the sensors, implement custom control logic, and control actuators based upon the control logic and system state. Distributed controllers also include a network communications interface which connects to upstream systems including the supervisory control layer and HMI. Third, the supervisory control layer is used to store process data, to implement system level control schemes, and to control distributed controllers. Last, HMI allow the human operator to monitor and control the physical process.

The test bed described in [2] is typical of small scale research environments which use commercial industrial equipment to implement model SCADA SYSTEMS. The test bed is consists of 7 systems including a gas pipeline, storage tank, water tower, industrial blower, assembly line conveyor, steel rolling process, and a chemical mixing system. Two of these physical systems were used as a basis for the virtual systems described in this paper; the gas pipeline and storage tank. Figure 1 shows pictures of physical systems and HMI screens from the test bed in [2].

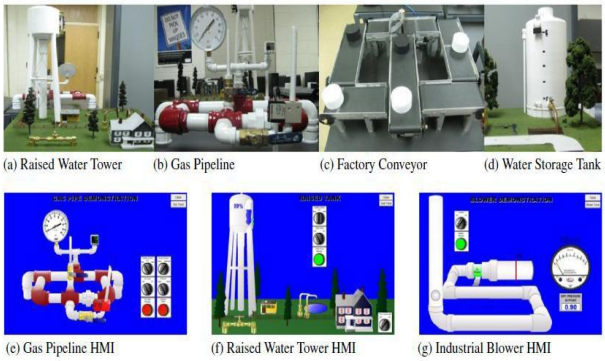

Figure 1: Laboratory Systems and HMIs

In [3], Bela Genge et al. propose a framework based on Emulab and Simulink to recreate cyber components and physical processes for a security analysis of networked industrial control systems. The proposed framework includes an architecture that intends to strikes a balance between test beds that consist entirely of physical components, such as [2], and test beds with entirely simulated components. A set of required functionalities for cyber-physical experimentation are provided. While the proposed architecture is helpful in envisioning a successful framework for SCADA experimentation, no realization of this framework is discussed in the paper.

In [4], 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.

In [5], Reaves and Morris describe an open virtual test bed for industrial control system security. This simulation was built in Python as both a process simulator and a PLC emulation and was designed to be interoperable with commercial SCADA equipment. The virtual test bed uses open source implementations of the MODBUS family of protocols, models PLC ladder logic programming, and models the physical behavior of the gas pipeline and storage tank systems using a curve fit for common system behaviors.

There are many simulators for both PLC and process simulation, such as those available from Rockwell Automation [6], Mathworks Simulink [7], MHJ Software, Modellica [8], and many others that are of commercially available. These are each significant for their modeling capabilities in their respective fields. However, none of these are designed for SCADA system modeling, and would require custom tools, such as what this paper describes, to create a comprehensive SCADA system.

This paper describes a pair of systems that combines the benefits of both physical system, and the simulators described. They closely model the behavior real systems, but are easily expandable, designed with SCADA modeling in mind, and can incorporate other simulators. They also benefits from being open source so that further development by other researchers is simple.

III. VIRTUAL SCADA LABORATORY

A. Laboratory Components

Three components comprise this virtual laboratory; a simulation of a gas pipeline process (sensors and actuators), PLC simulation, and an HMI. Also included are attack and detection systems. Each piece of the laboratory is run in a separate virtual machine (VM). This makes maintenance of the systems much easier such that when errors occur, the system can easily be restored to a previous working state. It also means that a virtual network can be utilized across the VMs. Thus all network traffic between the VMs is real network traffic that can be logged, disrupted, and modified as in a physical system.

1. Process Simulation

For this gas pipeline simulation, the components modeled were a gas compressor and a solenoid release valve. There are 4 states that are modeled for changes to the pressure of system:

1. If the compressor is compressing and the valve is closed, the pressure will indefinitely rise.
2. If the compressor is not compressing and the valve is open, the pressure will fall until it reaches zero.
3. If the compressor is compressing, and the valve is open, the pressure will rise, to an equilibrium pressure.
4. If the compressor is not compressing and the valve is closed, the pressure will remain constant.

The process simulation was implemented as a curve-fitted model of the system described by Morris et al [2]. For the first 2 states, the equations are quadratic. These are shown in the equations below where "*t"* represents time, relative to the current state and "*p*" represents pressure:

$$p = 2.0052 * t^2 \quad (1)$$

$$p = 0.098 * t^2 - 4.439 * t + 49.83 \quad (2)$$

The third state is modeled using a piecewise model, such that the pressure will converge to an equilibrium pressure, around 7.8. This value was discovered empirically from the system described in [2]. The third equation described the response as the pressure rises to 7.8, and the fourth describes the response once the pressure exceeds 7.8. The

This paper is based in part upon work supported by the National Science Foundation Secure and Trustworthy Cyberspace program under Grant No. 1315726

rand function below represents a Python uniform random number between -0.02 and 0.01:

$$p = t * (0.77319 - 0.0210857 * t) + 0.151637 \quad (3)$$

$$p = 7.3 + rand(-0.02, 0.01) \quad (4)$$

The fourth state requires no equation, as the pressure remains constant in this state.

These simulations are not built to model a complex response of the physical process, only a very simplified response. This should not be taken negatively. According to Reaves, an original architect of this simulation, the main functional goal of the testbed is to be able to emulate realistic serial and TCP/IP industrial control system protocol communications [5]. This process model is further discussed in [5], and serves as a starting point. Further work on expansion of this model to include more complex physical process models is described in the Future Work section.

2. PLC Simulation

A central part of this laboratory is the simulation of the PLC hardware and software. In real world systems, a typical PLC controller is programmed to perform 4 steps in an infinite loop: read inputs, analyze current state, calculate responses, and write outputs. This process is what the controller simulation seeks to emulate.

The simulations in this laboratory simulates the behavior of the PLC ladder logic of the lab described in [2]. These simulations emulate the process described above. Each data read, calculation, and output setting takes place one at a time, emulating each rung of ladder logic. The virtual PLC devices (VDEVs) communicate with the process simulator by emulating the analog and digital communication received from sensors and actuators. PLC simulation is implemented with Python.

The VDEVs communicate with other virtual devices using the MODBUS/TCP or MODBUS Serial Line protocols by utilizing the modbus_tk Python libraries. This enables the VDEVs to communicate with external devices such as physical PLCs and HMI using a standard SCADA communication protocol. It also allows researchers and students to view, capture, analyze, and route the traffic just as in real SCADA systems. The implementation of MODBUS within this simulation is indistinguishable from MODBUS traffic of real SCADA devices.

3. Human Machine Interface (HMI)

The third component of the laboratory is the HMI. As is the case in any SCADA system, an HMI is needed to visualize the changing process and interface with the control system. For this laboratory, 2 separate HMI were developed. The first HMI is the same HMI used with the modeled SCADA SYSTEM. Figure 2 below shows the commercial gas pipeline HMI screen. Using the commercial HMI maximizes test bed fidelity. The second HMI is an open source version. The open source version is similar to the commercial HMI, but, is freely distributable.

This paper is based in part upon work supported by the National Science Foundation Secure and Trustworthy Cyberspace program under Grant No. 1315726

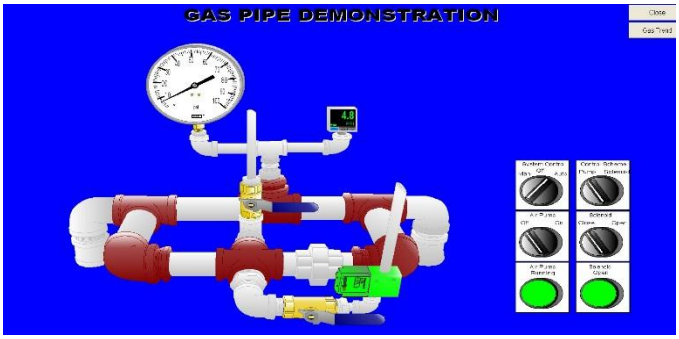

Figure 2: Commercial Gas Pipeline Human Machine Interface (HMI) Screenshot

The open source HMI was developed using the Python TkInter libraries. Figure 3 shows the open source HMI.

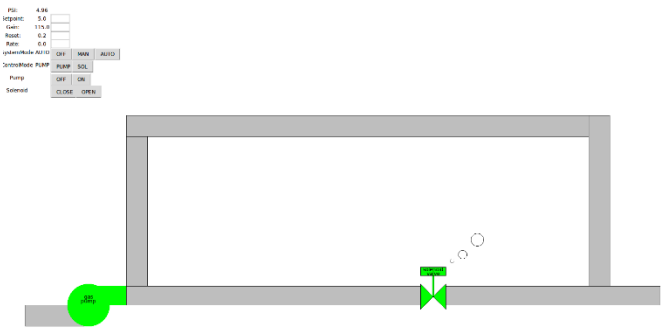

Figure 3: Open Source Python-based Human Machine Interface Screenshot

While these HMIs appear different, one primary way in which they are the same is the communication method they use. Both use MODBUS to request data from and write data to the PLC. All the data being displayed comes a MODBUS read request to PLC, and changes made by the user are translated into MODBUS write requests. Thus, the reads and writes from the HMI appear across the network, much like what is shown in Figure 3.

Because either commercial or experimental HMIs can be used, both major industry HMIs and developmental HMIs can be ported to this laboratory and tested for vulnerabilities. Research like that of Dr. Wesley McGrew in analyzing HMI vulnerabilities can take place without requiring access to a physical SCADA laboratory [23]

B. Attacks

In order to research the patterns and attack vectors of malicious activity, cyber-attacks were developed against the virtual laboratory SCADA systems. Developed attacks against the system include reconnaissance, command injection, and denial of service attacks.

Reconnaissance attacks are a category of attacks used by attackers to gain information about the control system before any destructive activity can take place. One of the reconnaissance attacks implemented is an Address Scan. In the implemented Address Scan attack, the attacker will send a request to every MODBUS device from 1 to 255 requesting to read a particular holding register. If a

MODBUS response is received, the attacker knows that a MODBUS devices exists at that address.

Command Injection attacks are another category of attacks used by attackers to maliciously adjust settings within the SCADA system. One such attack implemented is called an Altered Control Set Point attack. In this attack, the attacker purports to be a MODBUS device with a unique MODBUS device number, acts as a client, and sends a command to the server to alter the set point of the system. The server receives this response as any authentic command, alters the set point, and begins adjusting the process actuators to achieve this new set point.

A Denial of Service (DOS) attack is another type of attack used against SCADA systems. In a denial of service attack, the attacker seeks to interrupt network communication by any means possible. In one such attack developed, the attacker floods the virtual PLC with packets of pseudo-random data every 100ms, such that the PLC cannot respond to any other device due to the overwhelming amount of network traffic.

C. Laboratory Uses & Future work

This laboratory was developed for use in both teaching and research environments. In conjunction with the laboratory, a set of 4 laboratory exercises were developed for student classroom use.

In the first laboratory exercise, an overview is given of critical SCADA concepts using the virtual pipeline system as an example. This includes discussion of process components, a brief discussion of the MODBUS protocol, and a short discussion of virtualization. And 3 exercises to demonstrate the functionality of the HMI and the gas pipeline. In the second laboratory exercise, a more thorough discussion of the MODBUS protocol is given, and 3 attacks are composed by the students involving reconnaissance, command injection, and denial of service. In the third laboratory exercise, a discussion of common distinctions of IDSs is given, a discussion on the SNORT IDS, a dissection of a SNORT IDS rule, and 3 rules are written to defend the virtual pipeline system against the attacks from lab exercise 2. The fourth lab demonstrates the effects of distributed controls. This is done by explaining the basics of PID math, implementing the PID algorithm within the HMI, as opposed to within the PCL, then implementing a DOS attack against both the distributed and non-distributed control schemes, and the benefits of a distributed control system are illustrated.

In addition to teaching uses, this laboratory is written to be utilized for SCADA security research. Because the virtual devices emulate real processes and PLC systems, and use legitimate SCADA communication protocols, they are capable of generating SCADA network traffic data, which is useful for intrusion detection research. Due to the virtual and open source nature of the laboratory, it is easily distributable to other researchers. Given that the current simulations described implement a curve-fitted model, there is potential for replacing this model with a process model built using the Mathworks Simulink modeling software. Given that Simulink is an industry standard in process modeling, using Simulink also makes replacing a given Simulink model with a higher fidelity model easier for future researchers. Further development toward this end is ongoing. Other use cases include using this laboratory for the creation of SCADA datasets for use in machine learning algorithms. Because the implementation of MODBUS/TCP within this laboratory is indistinguishable from MODBUS traffic of real SCADA devices, large volumes of authentic SCADA traffic can be collected. Using the attacks described as well as further developed attacks in conjunction with normal operations, datasets of malicious and non-malicious SCADA network traffic can then be created. Such datasets are useful in the field of intrusion detection research to detect patterns in malicious network behavior [9]. Development of such datasets is ongoing.

This paper is based upon work supported by the National Science Foundation Secure and Trustworthy Cyberspace program under Grant No. 1315726

IV. VIRTUAL HYDROELECTRIC POWER SYSTEM

A. Overview

The VHPS is a simple representation of a single generator hydroelectric power system. It was developed using the open virtual SCADA testbed described [5]. Using the environment, the VHPS has virtual devices (VDEVs) in the place of PLCs.

1. Communication

The VHPS has two VDEVs that control the power generation process. Whereas real PLCs are programmed with either ladder logic or C, the control logic for the VDEVs is written in Python. Stemming from the open virtual SCADA testbed, the VHPS behaves according to a Master/Slave communication model. In this model, the Master sends commands to the Slave VDEV, and the Slave responds with system measurements. The Slave uses commands from the Master and system measurements from sensors and actuators to control the process operation.

The open virtual testbed allows for VDEVs to communicate in all forms of Modbus. For the VHPS, the VDEVs are configured to communicate using Modbus/TCP. To generate realistic SCADA communication traffic, each VDEV is housed within its own virtual machine (VM). Placing the VDEVs into their own VM gives each its own IP address, which models the realistic physical separation of PLCs in real systems.

2. System Overview

The VHPS has three operational processes: Start-up, Power Generation, and Shutting Down. The Start-up state is a transitional period from the system being off to the system being ready. Similarly, the Shutting Down state is a transitional period from the system being ready to the system being off. The Power Generation state represents the system being both ready and supplying power.

During the start-up process, there are a number of stages the VHPS steps through.

The first step of the start-up process is the sounding of a warning alarm for 30 seconds. The purpose of the alarm is to warn people that may be on the reservoir that the system is about to start. Next, the control gates open to the breakaway position, which is roughly 35%. The breakaway position is the position the control gates are opened in order to overcome static friction and allow for relative motion. After this, the gates are slowly brought back down to the Speed No Load (SNL) position. The SNL position is 10% for the VHPS. 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. After this, the synchronizer closes the breaker and turns off.

After the start-up process has completed, the VHPS enters the next operational process: 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 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. As per the schedule given to the operator, there are periods when the system does not need to be online. During this period, the VHPS enters the Shutting Down 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 considered offline.

B. Attacking The VHPS

The use of Modbus/TCP and the redundant communication nature of the VHPS creates a unique vulnerability, exploited by hijacking control over the Slave from the Master. Hijacking occurs through the exploitation of predictable TCP sequence numbers in redundant communication within SCADA systems. Hijacking control over the Slave consists of three parts: monitoring the SCADA communication stream, predicting the TCP sequence numbers, and submitting a Denial-of-Service (DoS) attack.

1. Monitoring Communication

In order to hijack the session, the attacker must have the ability to both monitor and capture network traffic between the Master and Slave. Switches have become more common in local networks than hubs in modern times. The method for monitoring traffic depends upon the local network implementation. When connected to a network using an unmanaged hub, the attacker can place his or her network interface controller (NIC) in promiscuous mode to view traffic. For a switched network, executing a Man-in-the-Middle (MITM) attack is necessary. Wireshark is used to view the communication stream between the Master and Slave VDEVs. It recognizes the Modbus/TCP protocol and allows for easy interpretation of the communication stream.

2. TCP Sequence Prediction

Modbus defines a set of function codes (FCs) that consist of various commands the Master can send the Slave [10]. In Modbus/TCP, the TCP sequence number increases according to the protocol data unit (PDU) size. In SCADA systems, the number of registers to be read from and written to does not change very often. As a result, PDU sizes for a specific command do not change very often either. Consequently, the next expected TCP sequence number can be easily predicted simply by observing the FC.

For the VHPS, the PDU size for FC 0x03 is 12 bytes, and the PDU size for FC 0x010 is 33 bytes. Fig. 5 provides an example for the VHPS communication stream between the Master and the Slave VDEVs.

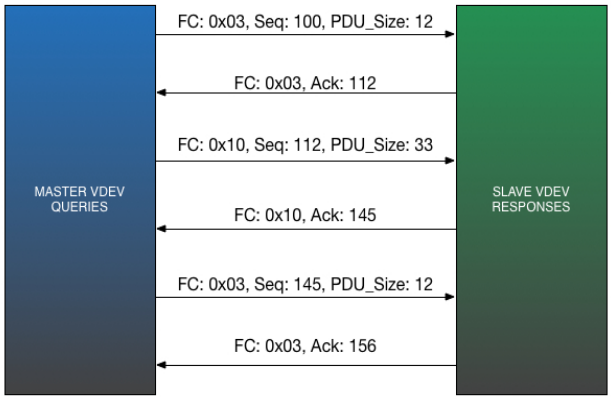

Fig. 5. TCP Sequence Number Example.

Once a packet is captured, the TCP sequence number and the Modbus Transaction Identifier are extracted. The purpose of the Modbus 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 control hijacking: the DoS attack.

3. Denial-of-Service

In this methodology, the attacker attempts to hijack through TCP sequence prediction. The attacker predicts the next sequence number by capturing the most recent command and examining the FC. In a TCP session, if a recipient receives a packet with a sequence number that it has already seen, it will ignore the packet. Therefore, whichever sender's (Master or attacker) packet reaches the Slave first will be accepted and the other ignored. This is called a race condition. The role of the DoS attack in hijacking control is to eliminate the race condition between the Master and the attacker.

Due to the nature of TCP and resulting from a lack of authentication in Modbus, the attacker only needs to successfully send one packet to the Slave to deny the Master and take control over the Slave. If the fabricated packet reaches the Slave before the legitimate packet, the legitimate packet will be ignored by the Slave. The DoS attack for hijacking control simply replays the packet captured during sequence number prediction.

4. Function Code Scan

After eliminating the race condition, the attacker is now free to send commands to the Slave. To effectively launch an attack, some knowledge about the system or devices is necessary.

The Modbus Protocol Specification defines a set of commands that a Master device can send a Slave device [10]. Each command is assigned an FC. With each FC, there is a formation for both the command and the response. All responses will either be formed normally as specified by the protocol specification or will be formed as an error response. An error response contains two bytes. The first byte is the FC plus 0x80. The second byte is known as the exception code and provides information about the nature of the error. Although Modbus provides these commands, those that are not needed can be disabled and, therefore, not supported by the Slave device. Sending commands with unsupported FCs results in wasted effort. Thus, it is beneficial to know which FCs are supported and which are not.

The Function Code Scan attack is a Reconnaissance attack. Its purpose is to determine supported and unsupported FCs. It is built from the packet captured for sequence number prediction. As needed with the DoS attack, the TCP sequence number and the Modbus Transaction Identifier are updated accordingly. The data field in the Modbus PDU does not need to change with the FCs for the attack. Changing the FC field is enough to determine which are supported and which are not. The simplest way the update the FC field is through the use of a loop. With each iteration of the loop, the FC is incremented by 0x01, and the command packet is built and sent to the Slave.

From the results of the FC Scan attack, the attacker knows that the Slave VDEV supports various write commands: FCs 0x05, 0x06, 0x0F, and 0x10. Using any of these FCs allows the attacker to change certain system operation set points.

5. Set point Manipulation

After executing a FC Scan the attacker can now launch a variety of attacks. From the results of the previous attack, the attacker knows that the FC 0x10 Write Multiple Registers is supported. With this command, the attacker can choose to alter the values of one or more Slave input Points. The Slave's input Points are referred to as set points.

The Set point Manipulation attack is a Command Injection attack. Whereas the FC Scan attack's goal is information retrieval, the goal of this attack is to alter system behavior in some manner. For the VHPS, the attacker has the ability to start or shutdown the system, disconnect the system from the power grid, adjust the position of the control gate, or alter the amount of power the system is providing. The fabricated packet is built from examining communication traffic and analyzing legitimate packets with FC 0x10. From the analysis, the set point memory locations and order can be determined. Depending upon the desired outcome of the attack, the attacker can change one or multiple set points with a single packet.

For this attack, FC 0x10 'Write Multiple Registers' was used to alter the system set point that changes the amount of power the system needs to produce to maintain power grid stability. Fig.10 shows the results of this attack on the power generated by the system. During control hijacking, a DoS attack is executed in which the attacking VM takes over control of the Slave VDEV from the Master VDEV. As a result, commands changing Slave set point values are not reflected on the HMI. Using FC 0x10, the attack alters the 'Power Needed' set point on the Slave VDEV to 48 kW. Upon receiving this fabricated command, the Slave begins closing the control gates to meet this set point, and the Master does not receive any communication of this action.

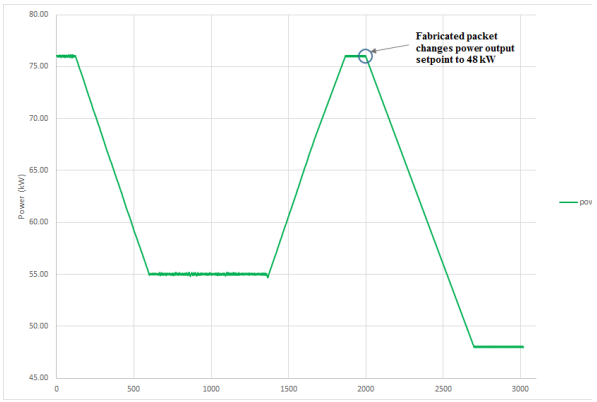

Fig. 10. Power Output Set point Change by Attacker.

V. CONCLUSION

This paper has described 2 systems: First, a Virtual SCADA Laboratory, built by modeling small scale industrial processes, modeling commercial PLC programming, employing the widely used MODBUS/TCP network protocol, and utilizing popular commercial HMIs. The system is capable of interfacing with physical commercial SCADA devices, such as PLCs and HMIs. Attacks against this system have been developed, and are included in the laboratory. As well as attacks, detection rules have been developed for the included attacks. This laboratory is portable due to its small size and virtual nature and is a useful research tool. It is also a useful tool for teachers in the cyber-physical system classroom.

The second is a simple, virtual hydroelectric power system (VHPS) that uses the common SCADA communication protocol Modbus, specifically Modbus/TCP, for communication between virtual PLCs or VDEVs. The system is comprised of a Master VDEV, which issues commands, and a Slave VDEV, which alters system operations based upon the commands. A control hijacking scheme combining a TCP sequence prediction attack and a DoS attack was presented. Following control hijacking, a FC Scan attack was developed with the purpose of garnering knowledge about supported and unsupported function codes. Spawning from the FC Scan attack, a Set point Manipulation attack was developed with the goal of altering system operation.

VI. REFERENCES

[1] Dept. Homeland Security. "Recommended practice: Improving industrial control systems cyber security with defense-in-depth strategies", 2009

[2] Morris, T., Vaughn, R., Dandass, Y. *A Testbed for SCADA Control System Cybersecurity Research and Pedagogy*. The 7th Annual ACM Cyber Secruity and Information Intelligence Research Workshop (CSIIRW). October 12-14, 2011. Oak Ridge, TN.

[3] Genge, B., Siaterlis, C., Fovino, I., Masera, M., *A cyber-physical experimentation environment for the security analysis of networked industrial control systems.* Computers and Electrical Engineering. Elsevier. Volume 38 (2012) Page 1146–1161. DOI: 10.1016/j.compeleceng.2012.06.015

[4] A. Lemay, J. Fernandex, and S. Knight, "An isolated virtual cluster for SCADA network security research," *Proceedings of the 1st Internationsl Symposium for ICS and SCADA Cyber Security Research*, 2013.

[5] Reaves, B., Morris, T. *An Open Virtual Testbed for Industrial Control System Security Research*. International Journal of Information Security (IJIS). Springer. Volume 11, Issue 4 (2012), Page 215-229. DOI: 10.1007/s10207-012-0164-7

[6] *RSLogix Emulate 500 Getting Results Guide*, Version 21.00, Rockwell Automation, Milwaukee, WI, Publication LGEM5K-GR016G-EN-E, January 2013

[7] *Modeling Guidelines for High-Integrity Systems*, Rev 1.10, TheMathWorks Inc, Natick, MA, October 2014

[8] Mattsson, S. Elmqvist, H. Otter, M. *Physical System Modeling with Modelica*. Control Engineering Practice, Volume 6, Issue 4, April 1998, pg. 501-510, DOI: 10.1016/S0967-0661(98)00047-1

[9] Beaver, Justin M., Borges-Hink, Raymond C., Buckner, Mark A., "An Evaluation of Machine Learning Methods to Detect Malicious SCADA Communications," in the Proceedings of 2013 12th International Conference on Machine Learning and Applications (ICMLA), vol.2, pp.54-59, 2013. doi: 10.1109/ICMLA.2013.10

[10] Modbus Organization, "Modbus Application Protocol Specification," 2006

[11] McGrew, R.W. *Vulnerability Analysis Case Studies of Control Systems Human Machine Interfaces*, Doctoral Dissertation, Mississippi State University, 2012