Advances in the Protection of Critical Infrastructure by Improvement in Industrial Control System Security

Thomas H. Morris
Bagley College of Engineering
Mississippi State University
Mississippi State, Mississippi
morris@ece.msstate.edu

Rayford B. Vaughn
Office of Research and Economic Development
Mississippi State University
Mississippi State, Mississippi
vaughn@research.msstate.edu

Elena Sitnikova
School of Computer and Information Science
University of South Australia
Adelaide, South Australia
elena.sitnikova@unisa.edu.au

Abstract

This paper describes an international research effort that has been working toward identification of vulnerabilities in industrial control systems, mitigation strategies to address vulnerabilities, and development of tools to prevent intrusion on such systems. The research represents over five years of externally funded work within the United States and a strong partnership between the U.S. institution and both Queensland University of Technology (QUT) and the University of South Australia (UniSA). The authors introduce the security problem with such systems, discoveries to date, and the development of tools to provide intrusion detection, intrusion prevention, and forensic data capture within industrial control systems.

1 Introduction

The majority of industrialized nations have recognized the need for protection of those national infrastructures that society depends on to maintain their quality of life and to guarantee the safety and security of the citizenry. In the United States, a Presidential directive (PDD-63) issued in May 1998, set up a national program of "Critical Infrastructure Protection" (PDD 63, May 1998). In Europe, the 'European Programme for Critical Infrastructure Protection' (EPCIP, 2006) refers to the doctrine or specific programs created as a result of the European Commission's directive EU COM(2006) 786 which designates European critical infrastructure that, in case of fault, incident or attack, could impact both the country where it is hosted and at least one other European Member State. Australia has historically been cited as the first country that actually experienced an attack on an industrial control system that resulted in damage to critical infrastructure (Slay and Miller 2008) and recently released its Critical Infrastructure Resilience Strategy in 2010 and entered into a joint agreement with the United States on May 18, 2012 between Australia‟s Attorney General Nicola Roxon and U.S Secretary Napolitano. Attorney General Roxon commented, "Countries everywhere are increasingly reliant on critical infrastructure such as telecommunications, which enables „online‟ activities that contribute to global commerce and trade and play an increasingly important role in national security. Both Australia and United States recognize the considerable benefits delivered by these activities, but also the challenges of managing cyber security and resilience to counter malicious activity." However, while governments now recognize the threat that exists to their citizens, very little has actually been done to improve existing industrial control system security or to require that owner/operators do so. This was most recently witnessed in the inability of the U.S. Congress to pass a much debated "Cyber Security Bill" which would have established optional standards for the computer systems that oversee the U.S.‟ critical infrastructure - like power grids, dams and transportation.

Since 2007, the authors have been engaged in the research reported in this paper that has demonstrated both significant and exploitable vulnerabilities and strategies that are plausible, affordable, and reasonable to prevent or mitigate such attacks. While the research was initiated at the U.S. institution, we discovered similar interests by our collaborators at QUT and UniSA which has led to exchange visits by faculty and students as well as current collaborative research endeavor.

The remainder of this work is organized as follows. First, a section on the current research laboratory is provided. Sections follow that describe representative (but not comprehensive) vulnerability points and mitigation strategies. Finally, the paper offers future works and conclusions.

2 Development of the Research Laboratory

The authors established an initial industrial control system (ICS) security laboratory in 2006 with a grant from the National Security Agency (NSA) in the amount of $75,000. These funds were used to design and develop a working industrial control system laboratory with actual control devices (not models) and software commonly found in industrial critical infrastructure applications. The laboratory was designed such that its control function could be observed while controlling miniature devices. This laboratory[1] implemented control system instantiations for a gas/oil pipeline application, an industrial blower application, an oil storage tank application, a conveyer belt application, and a rural water association application (water tank). The laboratory, in its initial configuration is shown in Figure 1. Figure 2 shows

[1] The laboratory was designed, constructed, and installed by Mr. John Gordon, Control Systems, Inc., 391 Industrial Park, Montevallo, Alabama.

the cart arrangement with SCADAPack programmable logic controllers (PLC) mounted under the cart.

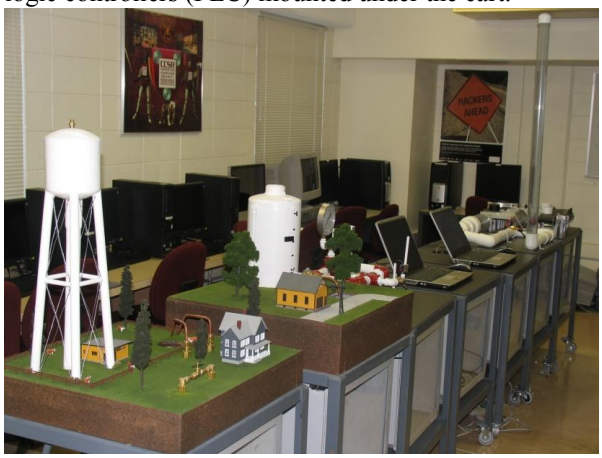

Figure 1: Initial configuration of the MSU SCADA Security Laboratory.

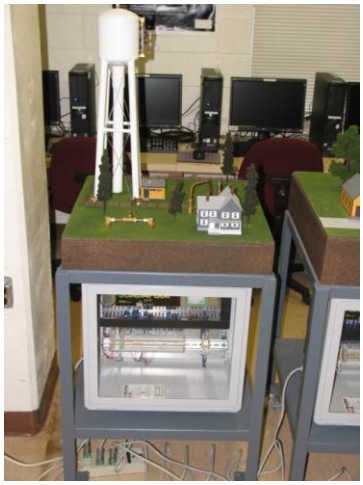

Figure 2: MSU SCADA Security Laboratory water tower application with view of SCADAPack programmable logic controller under cart.

This initial setup was not connected outside the boundary of the laboratory and used two laptops as operator stations running human-machine interface (HMI) software as an operator in a critical infrastructure application might be expected to do. Initially, the authors implemented an HMI product called GE Fanuc iFIX Proficy version 4.3. Based on research successes described later in this paper, the Department of Homeland Security and the National Science Foundation invested further research funding in the project to expand the laboratory, create an international effort, continue the vulnerability discovery effort, and to develop useful security tools which could be validated in an industrial partner‟s setting.

Further expansion resulted in the development of a steel rolling operation control system which models a steel manufacturing operation located close to the researcher‟s laboratory and shown in Figure 3.

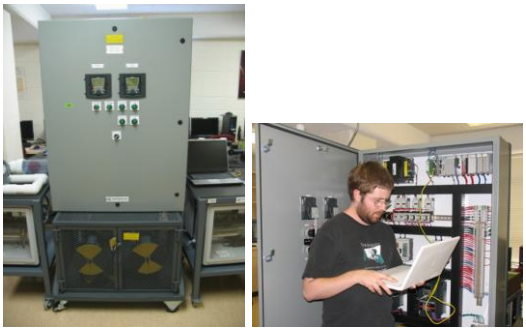

Figure 3: ICS for Model of Steel Plant Control Systems.

This steel rolling operation is used to press sheet metal to add strength via strain hardening and to improve surface finish for use in the automotive industry. Cyber-penetration of control systems monitoring and controlling a steel rolling operation can lead to loss of visibility and loss of control of the roller. Additionally, penetration of such a control system can lead to economic loss if an attacker is able to adjust process parameters which affect steel finish quality such as strength or surface finish by changing roll speed at either the entry or exit roll.

The steel rolling operation is controlled by a Master Terminal Unit (MTU), Remote Terminal Unit (RTU), and local and remote HMI sharing communication links. Both the MTU and RTU for the steel rolling operation are Allen Bradley Compact Logix L35E PLCs. The master terminal unit is connected to the Factory Talk View 5.0 HMI. Researchers recently added a wireless connection between this master terminal unit and a second lab space housing a Smart Grid transmission control system. Researchers developed new remote HMI screens in addition to the HMI and ladder logic to remotely monitor and control systems in the Smart Grid transmission control system portion of the testbed. The control system is also configurable. The testbed includes Allen-Bradley RSLogix 5000 programming software to compile and write ladder logic programs for use on the L35E Programmable Logic Controllers (PLCs). A wire bridge is as available on each system to add digital and analog inputs and outputs to the system. New monitoring and control logic associated with new inputs and outputs can be added using ladder logic. The Factory Talk View 5.0 HMI screens can be configured to add or remove visualizations or controls.

Lastly, a Smart Grid transmission control system was added to provide a second environment for ICS security research. Commercial devices in the Smart Grid transmission portion of the testbed include phasor measurement units, a phasor data concentrator, a synchrophasor vector processor, protection relays, controllers, a substation GPS clock, an Omicron relay test and calibration device, a Real Time Digital Simulator (RTDS), and OSISoft PI Historian. Additionally, the lab includes amplifiers to provide inputs needed for some of the PMUs.

The RTDS, protection relays, and phasor measurement units are connected to RTDS in a hardware-in-the-loop configuration. The RTDS is used to simulate transmission and distribution high voltage conditions and scenarios. The protection relays can be configured to

monitor bus voltage, current, frequency, and phase conditions to detect faults. This portion of the testbed also includes a MU Dynamics MU-4000 Analyzer. The MU-4000 is an appliance designed to test network appliances for cybersecurity vulnerabilities. The MU-4000 includes test suites for protocol mutation (also known as fuzzing), denial of service testing, and a test suite of published vulnerabilities. A Zenwall 10 Access Control Module and a Cisco 5510 Adaptive Security Appliance are in this portion of the test bed as well. Devices in the Smart Grid transmission control system are connected via a common Ethernet network. Industrial control system communication standards in use in the Smart Grid transmission control system include IEEE C37.118, MODBUS/TCP, DNP3, Generic Object Oriented Substation Events (GOOSE), and EtherNet/IP.

In 2011, our sponsors provided funding to include a smart meter laboratory representative of those that are rapidly being installed by utilities throughout the United States and elsewhere. Our smart meter installation includes 2 smart meters mounted in our laboratory environment connected wirelessly to a human machine interface software system for monitoring smart meter information. The installation also includes 4 additional smart meters mounted on buildings on our campus to allow monitoring of data related to actual loads. Figure 4 shows a collage of pictures of the smart meters installed across campus so that they record actual loads. This allows our researchers to study smart meter confidentiality, integrity, and availability vulnerabilities and develop security solutions.

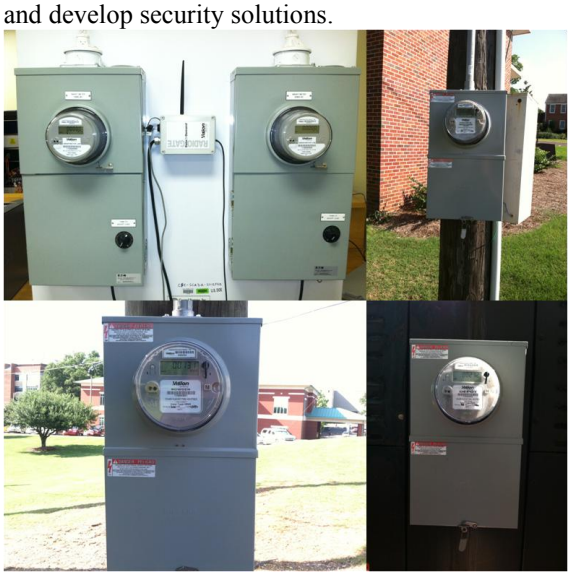

Figure 4: Collage of operational laboratory smart meters

We expanded our work and laboratory internationally through strong collaborations forged with Queensland University of Technology and the University of South Australia during 2010 through 2012. This involved exchanges of faculty and the establishment of laboratory facilities similar to our own at QUT as seen in Figures 5 and 6.

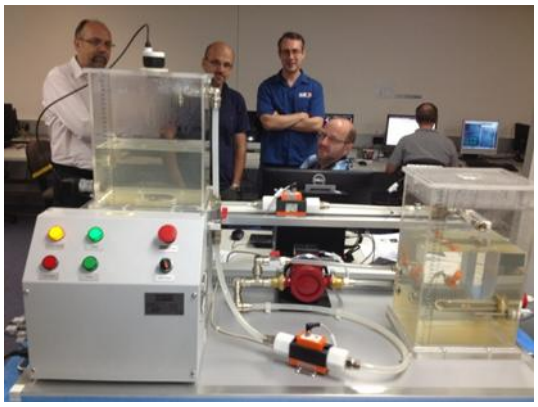

Figure 5: QUT ICS Security Module 1.

Figure 6: QUT ICS Modules based on author's ICS laboratory.

We also hosted a collaborator, Dr. Elena Sitnikova, from UniSA during the summer of 2012 to work with our ICS program. Dr. Sitnikova was able to bring new perspectives to our overall research agenda in the areas of metrics and forensics – a research area that we continue to develop.

The laboratory facilities described above are unique within academia and have allowed the authors to proceed with vulnerability discovery, tool development, and mitigation strategies.

3 Human Machine Interface Vulnerabilities

Human machine interface (HMI) packages are the software used by operators in the "control center" of an ICS application. The HMI software serves a dual purpose of presenting the data acquired from various elements of the SCADA system and allowing the operator to manipulate parameters of the system that are under the operator's supervisory control. HMI software is often configured to mimic the look and feel of a tangible control panel, with elements like switches, dials, sliders, and readouts. Figure 7 provides an example of these elements. Our research published vulnerabilities that we discovered in these software packages that reflect a lack of robust security architecture and violate commonly accepted principles of software security engineering (McGrew and Vaughn, 2009).

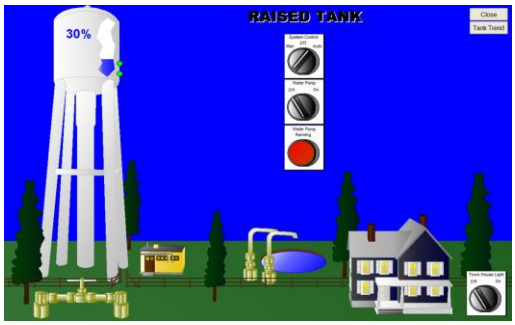

Figure 7: Typical view with an HMI package seen by the operator of an ICS

A representative finding of HMI vulnerabilities is best shown by our discovery of several in a popular HMI product. We refer to this product as HMI-1 to obfuscate the identity of the manufacture and specific product. The HMI-1 software serves as a HMI for end-user operators, and as an integrated development environment used by engineers to create the interfaces and scripts that make up the HMI. Software-enabled security features are also provided that allow for varying levels of access for different users. For example, an operator‟s account can be denied access to modify the interface, exit the full-screen interface, or shut down the system. A number of vulnerabilities in this product were found by experimentation in the author‟s laboratory that serve as useful examples of how design decisions and legacy code can affect the security of a modern control system. The vulnerabilities discovered are briefly described below. The authors are quite aware that similar vulnerabilities exist in other vendor‟s HMI products and we are currently investigating this hypothesis within the security laboratory.

One of the most surprising vulnerabilities discovered was that of password disclosure. Passwords in the HMI-1 product are not hashed securely for storage. The passwords for each user are obfuscated by an exclusive-or (XOR) operation against a static key before being stored in a file. The operation is obviously easily reversible by an attacker, resulting in the disclosure of all user passwords on an HMI-1 system. This is especially dangerous when the system is configured to authenticate users over a network, since the obfuscated passwords were not properly encrypted. An attacker with the ability to “sniff” network traffic can easily intercept and decrypt the users‟ passwords as we demonstrated to our government sponsors in our laboratory.

A second serious vulnerability was discovered which allowed an attacker to bypass authentication. This authentication bypass capability is especially serious since it automatically gives the attacker full administrator control and leaves no audit trail. The security architecture of HMI-1 does not prevent the modification and replacement of key security modules by the user. As a result, it is possible for an attacker to produce a copy of the HMI-1 login program and security manager library. The attacker can then modify the way key security modules operate. The modified modules can then be used on a target system to log in with no password, bypassing HMI-1‟s attempts at authentication and access control.

A third serious vulnerability often present in HMI software and within our HMI-1 case study is that of bypassing run-time restrictions. Restrictions can be placed on HMI-1 users in terms of what they are allowed to do in an HMI-1 system. This includes the ability to halt the system, run external programs, “alt tab” to other programs, and exit the full-screen interface. This prevents HMI-1 users from using the computer the HMI-1 system is running on for unauthorized tasks. This protection can be bypassed by an attacker through the use of a USB drive or CD configured to use Microsoft Windows‟ “Auto-Run” functionality. Using this method, an attacker can run malicious code designed to exit the HMI-1 interface, or leverage other vulnerabilities in the product. The authors‟ students have demonstrated this attack and have automated it on a USB drive.

Prevention of these and other related vulnerabilities largely rests in implementing correct software architectures and secure coding principles. For nearly 30 years, the software engineering community has had at its disposal a number of well known and important security engineering principles. They were first documented in 1970‟s in a then U.S. government classified report which established the need for security measures within the software engineering community as well as the untrusted nature of computing systems. As described in early fundamental papers, security engineering principles that directly relate to the vulnerabilities described in this paper include the principles of:

- **Complete mediation** - in which every access to a system‟s resources must be checked for authorization;
- **Security through obscurity** - where it is inadvisable to depend on obscurity for system security; and,
- **Least privilege** - in which each element of the system should operate at the lowest level of access possible to perform its task.

While such principles are easy to engineer into a product, they generally cannot be patched into a product that did not consider them in the first place. That was the case with our HMI-1 case study and the eventual recommendation was to re-engineer and re-code the product in its entirety. While space limits us to one case study, we do have others and the vulnerabilities presented here are representative examples that can easily apply to other HMI packages too (GLEG 2011 and Auriemma 2011).

4 Intrusion Detection Systems

While intrusion detection and intrusion prevention systems are often found in traditional IT systems as a normal protection mechanism, this is not the case in industrial control systems. Generally it is accepted in ICS communities that such tools are not practical in that reaction to a “false positive” alert could have catastrophic consequences. Our research has shown otherwise and our prototypes have been introduced with success into a large scale public utility company that provides gas and electric services.

The objective of our research in this area was to demonstrate that signature based intrusion detection

systems are an effective tool for monitoring serial port based industrial control system communication traffic to detect cyber-attacks. In our work, we described four types of attacks (denial of service, response injection, command injection, and reconnaissance attacks) which are known threats to serial port based industrial control systems. Our research was able to demonstrate a methodology for monitoring MODUS RTU and MODBUS ASCII traffic using an existing signature based intrusion detection system, Snort, commonly used to monitor Ethernet network traffic.

Denial of Service (DOS) attacks attempt to break the communication link between the remote terminal and master terminal or human machine interface. Breaking the communication link between the master terminal or human machine interface and the remote terminal breaks the feedback control loop and makes process control impossible. This sort of attack can be effective on standalone process control systems as well as those relying on internet connectivity. DOS attacks take many forms. Many DOS attacks attempt to overwhelm hardware or software on one end of a communication link so that it is no longer responsive. Other DOS attacks send ill-timed or malformed network packets which cause errors in a remote device‟s network stack and cause the remote device to become unresponsive. MODBUS devices include configuration settings to allow operators to remotely force a RTU into listen only mode, remotely restart communications, and remotely clear RTU counters and diagnostic registers. Each of these features can be used by an attacker to execute a denial of service attack.

With Serial Snort MODBUS, serial based control systems can also be protected against these threats. In (Reaves and Morris, 2009) the authors discuss how to discover and connect to a proprietary SCADA radio used to form a MTU to RTU communication link. This penetration enables an attacker to execute a denial of service attack against industrial control systems. The industrial radio used to demonstrate this attack uses a carrier sense and back off arbitration scheme to regulate slave device wireless transmissions. If a single device continually transmits without breaks other slave radios in the network will back off and wait to transmit until the offender stops transmitting. In practice such an attack stops communication between the MTU and RTU. This causes the HMI to no longer be updated with fresh physical process measurements and causes the operator to see stale process measurements. Without an intrusion detection system in place, the operator may not know that such an attack is in progress. The only symptom is that HMI inputs seem to have no effect on the controlled process. This attack may be executed via continuous transmission of invalid data such as streams of random numbers, or as continual transmission of otherwise valid MODBUS traffic. In either case Snort rules can be used to detect the attack. Such a rule can be based upon packet length, such as exceeding MODBUS/TCP maximum length, or on the volume of traffic in a given time frame. Serial Snort times out on continuous transmission of greater than 300 bytes and transmits the captured traffic as a TCP/IP packet to allow Snort to detect the large continuous streams of data.

In another representative attack, the network stack implemented in MODBUS MTU and RTU devices may not be sophisticated enough to handle receipt of malformed communication traffic. Some devices will reset themselves or hang when a malformed packet is received. This problem has been recognized by many equipment vendors and contemporary devices often include more robust network stacks. However, many serial systems in existing control systems are legacy devices with no support for malformed packet checks. A Snort rule to detect non MODBUS/TCP traffic on the network can be developed to detect this sort of attack. Such a rule checks a field which is not used in MODBUS RTU/ASCII protocols and therefore this rule is not applicable for these protocols. However, the length check rules described above serve as a minimal malformed packet check. Also, some malformed packets are possible with MODBUS RTU/ASCII protocols. For example, only addresses in the range 0-247 are allowed in a field which can hold numbers greater than 247. A Snort rule could check for a packet with an illegal address. Many other rules related to detecting malformed packets are possible and space limitations prohibit us from describing them here.

Response injection attacks inject false responses into a control system. Since control systems rely on feedback control loops which monitor physical process data before making control decisions, protecting the integrity of the sensor measurements is critical. False response injection can be used by hackers to cause control algorithms and operators to make misinformed decisions. In a MODBUS RTU or MODBUS ASCII network, the MTU regularly queries the RTU to read registers associated with the physical process state. In a normal operation the RTU responds to the MTU queries with data from the requested registers. If two responses to a single query are received, the MTU accepts the first received response and rejects subsequent responses. The proprietary wireless vulnerability described in (Reeves and Morris, 2009) can be exploited to eavesdrop on MTU to RTU communications and to inject false RTU to MTU communication packets. This exploit can be used to inject falsified MODBUS response packets after a query is issued. The attacker monitors the radio network for MODBUS queries. If the attacker‟s falsified response arrives at the MTU before the valid response, the false response is accepted. Responding faster than the RTU can be achieved in two ways. First, the attacker can take advantage of the periodic nature of MTU queries and queue a false response before the MTU query transmission completes. In this case a Snort rule can be developed to alert when multiple MODBUS responses are received with the same address and same function code within a time period too small to normally receive two responses. Two responses in less than the normal query period would indicate an attack.

In a second version of this attack there is only one response because the attacker stops all transmissions from the radio connected to the legitimate RTU via the continuous transmission DOS attack mentioned above. The attacker inserts the falsified MODBUS response within the continuous stream of DOS traffic. The proprietary radio network exploited for these attacks uses

slotted communications with dedicated slots for MTU to RTU communication and RTU to MTU communication. The MTU continues to query the RTU during the DOS attack. In this case, the Snort rule mentioned above which alerts for packets larger than legal MODBUS packets would detect this attack. Control systems also often read measurement values such as pipe pressure, voltage, or current. Often these values are transmitted in fields which can hold values which are physically impossible measurements. For instance, a numerical float type field can hold negative numbers; however, a pipeline pressure can never be negative. Other measurements may have legal ranges which correspond to the physical properties of a specific system. In both cases, Snort rules can be developed to alert when measurements fall outside predefined ranges.

Command injection attacks inject false control and configuration commands into a control system. Control injection can be classified into two categories. First, human operators oversee control systems and occasionally intercede with supervisory control actions, such as opening a breaker. Hackers may attempt to inject false supervisory control actions into a control system network. Second, remote terminals and intelligent electronic devices are generally programmed to automatically monitor and control the physical process directly at a remote site. This programming takes the form of ladder logic, C code, and registers which hold key control parameters such as high and low limits gating process control actions. Hackers can use command injection attacks to overwrite ladder logic, C code, and remote terminal register settings. System specific rules can be developed to block commands not supported for a given control system. MODBUS supports RTU addresses in the range 0-247; however, most control systems will use a small portion of this allowed set. MODBUS supports function codes in the range 0 to 255. Many function codes are reserved for common functions, while others are available for custom functions. Finally, MODBUS read and write calls include a start address, within RTU memory, and the number of (bytes, coils, inputs) to be read or written. Snort rules can be developed to detect unsupported RTU addresses, unsupported function codes, and unsupported read/write address and size combinations. RTU often have registers which hold system set points; such as target measurement values for a PID scheme, limits to generate a system alarm, or limits to generate on/off control actions. Snort rules can be developed to detect invalid register values which may cause incorrect or unsafe process functionality.

Reconnaissance attacks allow cyber attackers to reconnoiter a system before attacking. For example, NMAP is used to identify connected systems and then finger print the system. Finger printing allows an attacker to learn which ports are open, the identity and version of the remote operating system, and the identity and version of remote network stack daemons. With this information the attacker can plan a more effective attack. MODBUS systems can be scanned to learn connected RTU addresses, supported function codes of connected RTU, and supported address ranges of MODBUS inputs and coils. Quickdraw (Peterson, 2009) includes rules for detecting function code scans and scans to learn supported address ranges of MODBUS inputs and coils. These rules also apply to MODBUS RTU and MODBUS ASCII systems. Serial Snort places the MODBUS RTU/ASCII RTU address value in the MODBUS/TCP Unit Identifier field. This allows rules to be developed to detect RTU address scans.

The vulnerabilities, exploits, and associated rules described in this section are relevant for MODBUS and other serial port protocols. Snort can be used for intrusion detection and prevention on systems using other serial port protocols as well and our research continues to discover methods to safely employ such rules.

5 Related efforts

This work is not an exhaustive treatment of the tools and techniques we have developed. In the accompanying presentation we can directly offer more examples of our development of forensic capture devices, a working simulation model of industrial control systems that provides accurate responses to sets of inputs (thus allowing some research experimentation to take place without the use of a physical laboratory) and the development of FPGA based devices that implement our intrusion prevention and detection code. We intend to continue our collaboration with UniSA and QUT and within the next year, demonstrate attacks via TCP/IP connections from the U.S. to Australia (and vice versa), along with the means and methods to prevent such attacks. We will also further test and enhance our intrusion detection and prevention tools as briefly described here.

6 Conclusions

Industrial control system operators often do not perform intrusion detection on PCs and enterprise systems in control rooms. These operators most often do not monitor communication traffic on the operation networks interconnecting industrial control system devices such as master terminal units, PLC, remote terminal units, and intelligent electronic devices. This paper provides an overview of tools and techniques developed to detect malicious activity on industrial control system MODBUS communication networks, describes typical vulnerabilities we are finding in HMI control software, and discusses our intention to foster out international collaborations. The sample rules described in this paper are intended for use with the SNORT Intrusion Detection System. We have additional papers under review that will publish the explicit format for the SNORT rules we allude to here.

7 Acknowledgments

The work described herein has been supported by grants from the U.S. National Security Agency, the Department of Homeland Security, and the National Science Foundation. Their support is gratefully acknowledged.

8 References

The European Programme for Critical Infrastructure Protection (EPCIP), Dec 2006,

http://europa.eu/rapid/press-release_MEMO-06-477_en.htm

GLEG Inc., "SCADA+ Pack Latest Updates," http://gleg.net/agora scada upd.shtml.

Auriemma, L., "Advisories," http://aluigi.altervista.org/adv.htm.

McGrew, W. and Vaughn, R., "Discovering Vulnerabilities in Control System Human-Machine Interface Software," Journal of Systems and Software, Elsevier 82 (May 2009) 583–589

Morris, T., Vaughn, R., Dandass, Y. A Retrofit Network Intrusion Detection System for MODBUS RTU and ASCII Industrial Control Systems. Proceedings of the 45th IEEE Hawaii International Conference on System Sciences (HICSS – 45). January 4-7, 2012. Grand Wailea, Maui.

Peterson, D. "Quickdraw: Generating Security Log Events for Legacy SCADA and Control System Devices," Conference for Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology, vol., no., pp.227-229, 3-4 March 2009

Presidential Decision Directive 63, May 22, 1998, http://www.fas.org/irp/offdocs/pdd/pdd-63.htm

Reaves, B., Morris, T. Discovery, Infiltration, and Denial of Service in a Process Control System Wireless Network. IEEE eCrime Researchers Summit. October 20-21, 2009. Tacoma, WA

Slay, J., Miller, M., Lessons learned from the Maroochy water breach, in Critical Infrastructure Protection, E. Goetz and S. Shenoi (Eds.), Springer, Boston, Massachusetts, pp. 73-82, 2008.