On SCADA Control System Command and Response Injection and Intrusion Detection

Wei Gao, Thomas Morris, Bradley Reaves, and Drew Richey
Department of Electrical and Computer Engineering
Mississippi State University
Mississippi State, MS 39762
{wg135, morris, bgr39, djr2}@ece.msstate.edu

***Abstract*—SCADA systems are widely used in critical infrastructure sectors, including electricity generation and distribution, oil and gas production and distribution, and water treatment and distribution. SCADA process control systems are typically isolated from the internet via firewalls. However, they may still be subject to illicit cyber penetrations and may be subject to cyber threats from disgruntled insiders. We have developed a set of command injection, data injection, and denial of service attacks which leverage the lack of authentication in many common control system communication protocols including MODBUS, DNP3, and EtherNET/IP. We used these exploits to aid in development of a neural network based intrusion detection system which monitors control system physical behavior to detect artifacts of command and response injection attacks. Finally, we present intrusion detection accuracy results for our neural network based IDS which includes input features derived from physical properties of the control system.**

Keywords: intrusion detection; SCADA control system; cyber security

I. INTRODUCTION

Supervisory Control and Data Acquisition (SCADA) systems are process control systems which interconnect and monitor remote physical processes. SCADA systems collect data from remote facilities about the state of the physical process and send commands to control the physical process creating a feedback control loop. SCADA systems are used in power transmission and distribution systems for situational awareness and control. Contemporary SCADA systems are commonly connected to corporate intranets which may have connections to the internet. SCADA communication protocols such as MODBUS, DNP3, Allen Bradley's Ethernet Industrial Protocol lack authentication features to prove the origin or freshness of network traffic. This lack of authentication capability leads to the potential for network penetrators and disgruntled insiders to inject false command and false response packets into a SCADA system either through direct creation of such packets or replay attacks. In this paper we have developed a set of command injection, response injection, and denial of service attacks, subjected commercial SCADA systems to these attacks, captured network traffic associated with these attacks using a SCADA network transaction data logger, and finally used the captured network traffic in combination with captured traffic from the SCADA control systems running normally to develop and validate a neural network based intrusion detection system which leverages knowledge of the physical properties of the controlled system to detect false command and false response injection attacks.

National Electric Reliability Council (NERC) Critical Infrastructure Protection (CIP) Standards 002-3 through 009-3 [2] require utilities and other responsible entities to place critical cyber assets within an electronic security perimeter. The electronic security perimeters must be subjected to vulnerability analyses, use access control technologies, and include systems to monitor and log the electronic security perimeter access. The Federal Energy Regulatory Commission (FERC) requires responsible entities involved in bulk electricity transmission to adhere to the NERC CIP 002-3 through 009-3 standards. No such regulation exists for the electric distribution systems and other critical infrastructure, such as water treatment and distribution, and gas distribution, in the United States. Electronic perimeter security will minimize the threat of illicit network penetrations, however, persons with electronic access to SCADA systems within the electronic security perimeter still remain a threat due to the lack of authentication capabilities in these systems. Additionally, the lack of authentication for process control system communication protocols means that if an attacker does penetrate the electronic security perimeter he will be able to inject false commands and false responses into the process control system without detection. As such, intrusion detection systems tailored for use in process control systems are needed to detect this behavior.

Team Cymru, a specialized Internet security research firm, released a briefing paper in 2008 [1] which discussed malicious port scan activity against their DarkNet (a honey pot) searching for open ports on port numbers commonly associated with SCADA system protocols. This report showed heavy scanning activity from four areas: The ASIA, USA, Western Europe and Eastern Europe. The report cited heavy scanning of DNP3 ports from Russia and Taiwan, and heavy scanning activities for MODBUS related ports in Western Europe and China. This port scanning is potentially indicative of attackers searching for SCADA systems for later attack.

Stuxnet is the first known worm to target an industrial control system. Stuxnet targeted PC's running the Siemens WinCC SCADA software product. Infected systems had a DLL replaced used by the WinCC Step7 tool. The worm then monitored communications between the WinCC tool and a remote terminal. If a specific signature related to the remote terminal was found firmware on the remote terminal was replaced with malicious code.

The remainder of this paper is organized as follows. Section 2 provides details on related works and an introduction to SCADA systems. Section 3 presents a set of exploits developed and tested in the Mississippi State University (MSU) SCADA security laboratory. Section 4 introduces a neural network based intrusion detection system (IDS) which leverages knowledge of the physical properties of the controlled system to detect false response injection attacks. Section 4 includes experimental results including intrusion detection accuracy, false positive rates, and false negative rates. Finally, Section 5 provides conclusions and proposed future works.

II. Background and Related Works

A. *Related Works*

Within the area of SCADA control system security researchers have developed pattern recognition systems which monitor MODBUS network transactions watching for signatures of known attacks and vulnerabilities [4]. Researchers have also used statistical techniques to monitor sensor data to identify system faults [5].

Traditional signature based IDS (e.g. Snort) focus on matching signatures stored in databases with network packets. This approach is efficient when detecting known attacks. However, the signature database depends on security experts' knowledge and experience. Furthermore, signature based IDS often cannot detect new attacks, so they are often behind attackers.

Statistical IDS use statistical models to classify network traffic as normal or abnormal (or into smaller sub-classes). Various model types or classifiers can be used to build the statistical model, including neural networks, linear methods, regression models, and Bayesian networks [6]. The main drawback of statistics based IDS is that they are not deterministic. Therefore, they suffer from more inaccuracies than signature-based intrusion detection systems. There are two types of inaccuracies in IDS: false positives and false negatives. False positives generate a false alarm when there is no intrusion, while false negatives will miss an actual intrusion. The accuracy of statistics-based intrusion detection systems relies on training dataset completeness, proper input feature development, and choice of classifier.

Artificial neural networks (ANN) model the biological nervous system, and are widely used in pattern classification, detection, and statistical analysis. ANN is one of the most commonly used classifiers for intrusion detection systems. In [7], Choudhary and Swarup describe research to apply neural network approach for intrusion detection systems with very high detection accuracy. However, implement artificial neural network in intrusion detection system in SCADA environment is still new research area.

This paper describes an IDS which uses device address, MTU command contents, RTU response contents, command and response frequency, and physical properties about the control process as input features to detect command and response injection attacks.

B. *SCADA System Overview*

SCADA control systems are distributed cyber-physical systems. These systems consist of Remote Terminal Units(RTU), Master Terminal Units (MTU), Human Machine Interface software (HMI) and the sensors and actuators that interface with the physical system. Remote terminal units (RTU) are connected to sensors and actuators to interface directly with the physical process. RTU commonly store control parameters and execute programs which directly control the physical process. For instance, an RTU may be used to control the water level in a tank. It will be programmed with a high water level and a low water level. The RTU continuously monitors the water level with a connected water level sensor. If the water level reaches the programmed high level, the RTU turns off a pump which fills the tank. If the water level reaches the low level the RTU turns on the pump to add water to the system. The RTU, the ladder logic (or other programmation), and the attached sensors and actuators form a feedback control loop.

Master terminal units (MTU) are connected to the RTU via communication links. The MTU polls the RTU periodically to read physical quantities of the controlled system such a voltage, pressure, water level etc. Typically this information is displayed on a Human Machine Interface (HMI) to allow operators to monitor the physical process. HMI typically allow the operator to interact with the physical process. Operators may change operating parameters. For example an operator may change the high and low water levels, from the previously mentioned water system example. The MTU, RTU, communication link, HMI, and operator form a second supervisory feedback control loop.

The communication links in SCADA systems can be thought of occupying layers 1, 2 and 7 of the OSI model; these are the physical, data link (and media access control), and application layers, respectively. The physical layer may be wired or wireless.. Wired networks may use leased lines, category 5 or 6 cable,, serial cable, and/or fiber optic cable. Wireless networks may use standardized communication systems such as IEEE 802.11, ZigBee, and/or WirelessHART. Wireless links may also use proprietary non-standard protocols. Finally, wireless links may include very long-distance solutions such as satellite and microwave links. There are many standards for layer 2 and 7 SCADA communication including Fieldbus, Profibus, MODBUS, DeviceNet, and Distributed Network Protocol version 3 (DNP3). DeviceNet includes the common Ethernet-based protocol Ether Industrial Protocol, abbreviated Ethernet/IP or E/IP. All of these protocols have versions that have been adapted to use TCP/IP as network and transport layers for use with commodity internet equipment. One common security flaw with all of these communication protocols is that they do not include cryptographic authentication, which means, RTU and MTU cannot validate the origin of commands and responses respectively.

C. *Mississippi State University SCADA Security Laboratory*

The MSU SCADA security laboratory (henceforth called the lab) was used to develop a set of SCADA control system exploits. The lab contains 5 laboratory scale SCADA control systems [8]. Each SCADA control system includes Control Microsystems, INC SCADAPack Light PLCs as MTU and RTU. The MTU is connected via RS-232 serial port to a PC running the GE IFIX human machine interface software. The MTU and RTU are connected wirelessly using integrated Freewave FGR 900MHz radios. The MTU and RTU can be configured to communicate using DNP3, MODBUS ASCII, or MODBUS RTU communication standards. The 5 laboratory scale control systems model a gas pipeline, a factory assembly line, a water tower, a water storage tank, and an industrial blower. All 5 control systems are mechanically functional models. The IDS system, accompanying training data, and exploits discussed in this paper were developed and validated using the water storage tank control system.

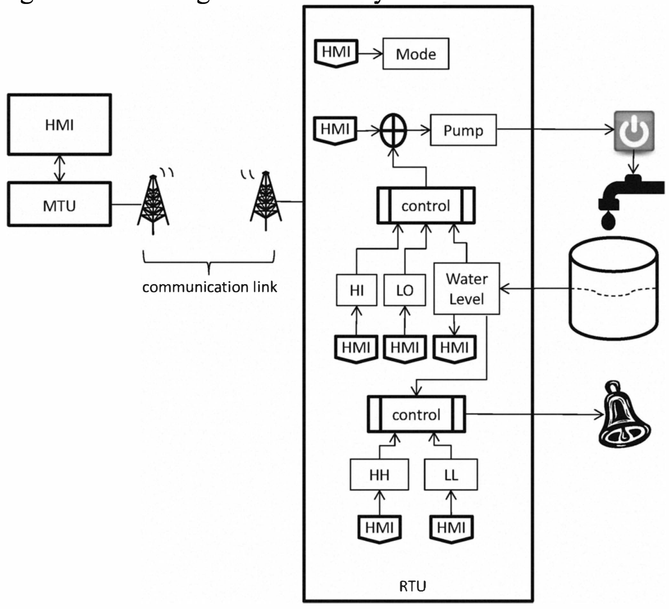

Fig. 1: Water Tank Control System Schematic

The water storage tank control system contains a 2 liter storage tank, a pump to force water into the tank, and a variable level water sensor. The RTU contains HI and LO water level registers which are programmed by the HMI. The RTU contains a water level register which is set by a sensor monitoring water level in the water tank. The RTU also contains a pump state register which may be programmed by the RTU or control logic. The RTU control logic attempts to maintain water level in the water tank between the HI and LO levels by turning on and off the water pump. The HMI periodically polls the RTU (every second) to read the water level for display on an operator's graphical user interface. The HMI also allows an operator to manually control the system to turn the pump on and off and set the HI and LO water level values. A water tank schematic is shown in Fig. 1.

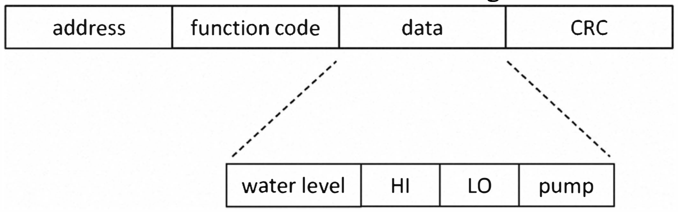

Fig. 2: MODBUS Link Layer Protocol Data Unit

The water tank control system uses the MODBUS protocol. The MODBUS protocol is a command response protocol. The control system HMI interfaces to the MODBUS application layer to send commands and receive responses to and from the control system respectively. Commands and responses are transmitted using a data link layer protocol data unit (LPDU). The physical layer (denoted as communication link in Figure 1) uses a RS-232 serial link configured for 9600 BPS communication with 8 data bits, 1 stop bit, no parity bit, and no flow control. No other Open System Interconnections (OSI) layers are used for the MODBUS communication. The MODBUS LPDU contains an address field, a function code, a data value, and a cyclic redundancy code (CRC) used for error detection. Fig. 2 shows a diagram of the MODBUS LPDU. The water level, HI and LO register values, and the pump status are encapsulated in the data field of the MODBUS LPDU.

III. Attacks on SCADA Control Systems

Many SCADA control systems are distributed over large distances and therefore use wireless networks to connect master terminals and remote terminals. Many industrial radios are vulnerable to illicit penetration from cyber intruders [9]. Once an intruder penetrates a SCADA control system, he or she may inject false responses into the control system to falsify physical process data or inject invalid control commands to RTU. Both scenarios can lead to harmful results. In 2000, a Maroochy, Queensland, Australia sewage control system was penetrated wirelessly by a cyber intruder who injected falsified commands to open flood gates [12]. Over 1 million liters of raw sewage leaked into 2 local fresh water streams over 3 month period while the intruder went undetected.

Three primary threats to process control systems are response injection, command injection, and denial of service.

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 from the physical process is critical. False response injection can be used by hackers to cause control algorithms to make misinformed decisions.

Many SCADA control system network standards do not include mechanisms to support response integrity.

Command injection attacks inject false control commands into a control system. Control injection can be classified into 2 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, with registers which hold key control parameters such as high and low limits gating process control actions. Hackers can use command injection attacks to overwrite RTU programming and remote terminal register settings.

Denial of Service (DOS) attacks disrupt the communication link between the remote terminal and master terminal or human machine interface. Breaking the communication link between master terminal or human machine interface and the remote terminal breaks the feedback control loop and makes process monitoring and control impossible. A common theme in DOS attacks is to cause hardware or software on one end of the network to become unresponsive responsive. Many varying mechanisms are used to cause the hardware or software to become unresponsive.

There are three modes of SCADA control system attack. First, an attack may be launched via external network connection. In this case, the attacker penetrates the SCADA control system network via network interface to gain access to the control system network. Such attacks include penetration via connections to the internet or penetration through dial-up connections. Second, an attacker may penetrate the SCADA network via wireless network connecting the MTU and RTU. In [9], Reaves and Morris discuss how to discover and penetrate a proprietary SCADA radio network and then inject false responses and denial of service attacks into the network. Finally, an insider with physical or electronic access to the SCADA control system may inject commands and responses over a network ordinarily isolated from outside connections, or an attacker may connect directly to control system equipment to initiate an attack.

A set of SCADA control system exploits have been developed for use in the MSU SCADA Security Laboratory. The exploits are grouped into six categories: illicit injection of false responses from RTU to MTU, illicit injection of false commands from MTU to RTU, man-in-the-middle attacks, replay attacks, and denial of service (DOS) attacks.

First, in [9] Reaves and Morris describe a SCADA proprietary wireless network denial of service vulnerability. The vulnerability allows a rogue wireless device to connect to a network and then prevent RTU from sending responses to MTU commands by continuously transmitting. The device causing the DOS may transmit anything. Since other slaves connected to the same radio network use a carrier sense back off mechanism before transmitting, and there is no limit on transmission length, the attacking device may continue transmitting indefinitely. A false response injection attack was developed which exploits this vulnerability. In the attack, a rogue slave radio joins the network via mechanisms described in [9]. All MTU communications are transmitted to all connected slave radios, allowing the rogue slave to monitor all MTU commands. The rogue slave transmits random numbers continuously while eavesdropping on MTU transmissions. Other slaves also receive master transmission, but may not transmit because the rogue slave is transmitting. The rogue slave device deciphers each master command and chooses between two response options. First, if the rogue slave prefers to transmit a response, it continues transmitting random numbers until a response is prepared and then transmits the false response in place of random numbers. After the false response is sent the rogue slave resumes sending random numbers. Second, if the rogue slave prefers the addressed slave to respond to the issued command, it simply stops transmitting random numbers long enough for the slave to transmit. The rogue slave cannot listen to slave transmissions in this attack. Therefore, the rogue slave would back off for a prescribed time long enough for the addressed slave to process and respond to the command. In practice, it may be easier for the rogue slave to simply respond to all commands and never allow the real slave to transmit.

MODBUS and DNP3 are subject to false response injection from a rogue slave which inserts a response to a command sent to another slave before the addressed slave can respond. The MTU is required to discard responses which are not associated with a command. There is no command sequence number in the MODBUS or DNP3 LPDUs. The RTU associates the first response received after transmitting a command with that command. Also, there is no digital signature or authentication mechanism in the MODBUS or DNP3 LPDUs to allow the MTU to confirm the response is from the addressed RTU. This vulnerability allows rogue slaves to eavesdrop on MODBUS or DNP3 communications and attempt to respond to a command before the address slave responds. If the illicit response arrives at the MTU first, the MTU will assume it is the correct response. A false response injection attack was developed which leverages this vulnerability. As with the previous attack, a rogue slave radio first joins the control system network via mechanisms described in [9]. Next, the rogue slave monitors transmission from the MTU. The rogue slave device deciphers each master command and chooses between two response options. First, if the rogue slave prefers to transmit a response, it immediately transmits a response. If the illicit response arrives at the MTU before the valid response it will be accepted. If the illicit response arrives after the valid response it will be ignored. Second, if the rogue slave prefers the addressed slave to respond to the issued command, it remains silent. This attack is not successful all the time due to the response race. Slaves using the proprietary radio network, as previously described, use a carrier sense back off mechanism. Therefore, if the addressed begins its response before the rogue slave, the rogue slave will wait until the address slave completes its transmission. In laboratory experiments exploit works approximately 50% of the time. The rogue slave in laboratory experiments used a Windows laptop computer connected to a radio. The network stack ran in a VMWare Linux virtual machine. It is believed that OS serial port queuing limited the effectiveness of this attack, and that a similar attack run from an embedded platform would be more successful.

MODBUS over TCP/IP and DNP3 over Ethernet suffer from similar false command and false response vulnerabilities. Both standards lack digital signature or authentication mechanisms. The TCP layer does include a sequence number; however, TCP sequence numbers are unique from each communicating party and are not used to match a command to a response. DNP3 and MODBUS both will accept the first response received and discard additional responses and therefore all suffer from similar race condition vulnerabilities. As such, an attacker which penetrates the control system Ethernet subnet may transmit false responses. In many cases control system subnets are not isolated from other corporate subnets. As such any employee or contractor with access to the corporate network may transmit illicit responses from any type of connection to the corporate network; wired, wireless, or VPN. In better managed networks, control system subnets are isolated from other portions of the corporate network. In this case, false response injection is limited to employees and contractors with privileges on the control system subnet or external attackers which penetrate the control system network. EtherIP response injection attacks conducted in a laboratory setting have similar success rates to the MODBUS race condition based response injection attack.

MODBUS TCP/IP, DNP3 Ethernet, and Allen Bradley EtherNET/IP all suffer from similar command injection vulnerabilities. Control system RTU contain registers which hold process set points. Process set points can be used to set minimum and maximum levels for process parameters or to control the state of mechanisms such as pumps and switches. The lack of digital signatures or authentication mechanisms allows a penetrating attacker to inject commands to control RTU. For example, an attacker may transmit a command to change the high and low water levels used to control the water tank control system in the MSU SCADA Security Laboratory. The illicit command will arrive from a IP address associated with the attacker and therefore the RTU response will be returned to the attackers IP address rather than to the MTU's IP address.

Ethernet based control systems can be subject to ARP poisoning. Attackers with local access to a control system subnet can use an ARP poisoning application such as Ettercap to setup and manage a man-in-the-middle attack. Ettercap supports active packet content analysis and filtering. Man-in-the-middle attacks can be used to inject false commands, false responses, or to create a replay attack which includes false commands and false responses. Commands may be passed from MTU to RTU unchanged, may be filtered to alter command contents, or may be discarded. Similarly, responses may be passed from RTU to MTU unchanged, may be filtered to alter response contents, or may discarded. Additionally, a man-in-the-middle node does not need to wait for an actual command to initiate an attack. False commands may be issued at anytime.

A second man-in-the-middle attack requires physical access to the control system MTU or RTU. In this case, a device is physically placed adjacent to the MTU or RTU which captures MODBUS, DNP3, or EtherNET/IP transactions at their source and alters or replaces those transactions. Laboratory experiments were conducted with 3 such man-in-the-middle scenarios. A MODBUS transaction data logger device [13] was upgraded to support command and response filtering and command injection. In the first scenario the man-in-the-middle node is physically connected to the MTU. The man-in-the-middle node captured MODBUS LPDU as they arrived from the MTU and as they left the RTU. The second scenario moves the man-in-the-middle node from the RTU edge to the MTU edge. This scenario is offered because it adds the ability to monitor and alter commands and responses to and from all control system slaves communicating with the MTU rather than just one. The third scenario uses a software data logger described in [13] for the man-in-the-middle node. The software man-in-the-middle node can be used in placed of a separate device to monitor communications at the MTU edge. The software man-in-the-middle node is run in a virtual machine on the human machine interface (HMI) node. The man-in-the-middle node is connected to the physical serial port and the HMI software is connected to a virtual serial port which routes communications through the software data logger. The software man-in-the-middle node was also altered to inject false responses in the same way as the stand alone man-in-the-middle node. The software man-in-the-middle node may be advantageous to an attacker since it may be possible to install it remotely after gaining control of the HMI node through the use of malware, or advanced persistent threat type attacks. In laboratory experiments the man-in-the-middle node was used to turn on and off a pump and to alter water level response values in the MSU SCADA Security Laboratory water tank control system.

There are multiple types of control system denial of service attacks. First, as previously mentioned, certain proprietary radio systems commonly used in SCADA networks suffer from a denial of service vulnerability which allow attackers to exploit the carier sense back off mechanism used for congestion control to stop other slaves from transmitting.

Many SCADA RTU will stop code execution when the denominator of a division function is set to zero. The typical response is for the processor in the RTU to fault which results in the RTU executable code ceasing to function and a complete stoppage of any process the RTU is controlling. The results of this can range from annoyance to catastrophe, depending on the manufacturing process. In a chemical process, a stoppage of this nature might result in an entire batch of product being scrapped and a tedious cleanup and setup process to take place. In an assembly process, machines and stations may require resetting. Regardless of the collateral consequences of the stoppage, the production will remain down until a maintenance person can return the processor to "run" mode. This could take an extended amount of time depending on the ability of the maintenance personnel to quickly diagnose the problem.

Control system MTU and RTU are network appliances which may have unintended side effects when malformed network transactions are sent to the device. For example, in laboratory experiments a SYN flood attack consisting of a few thousand packets cause an RTU cease responding for several minutes. The RTU did eventually recover. In separate laboratory experiments a Mu Dynamics network tester was used to send malformed network packets to multiple control system network appliances. System responses to the malformed packets varied. In one case a device reset itself after receiving a malformed packet.

IV. IDS FOR SCADA CONTROL SYSTEMS

Many of the attacks described in the previous section can be detected by a signature based IDS such as SNORT. For example, a previously described false response injection attack continuously sends random numbers to stop other control system slaves from transmitting before the attacker transmits a response. A signature based IDS can be programmed alarm if it detects continuous transmissions greater than a set length. Signature based IDS can be programmed alarm when an ARP poisoning attack is detected. A third attack described above used attempts to leverage a race condition to inject false responses. A signature IDS could be programmed if two responses to the same command were detected in close proximity to one another.

Signature based IDS require prior knowledge of threats to develop signatures. New attacks and variants on existing attacks can be missed by signature based IDS. Also, certain attacks are difficult for a signature IDS to detect. For instance, in Ethernet based systems a rogue may inject false commands to change RTU set points. These commands are very similar to valid commands. IDS signatures can be developed to detect invalid set commands, such as commands which move a set point above or below specific threshold values. However, it is much more difficult to create an IDS signature which alarms when a rogue changes a set point to a legal value. Replay attacks are equally difficult for a signature IDS to detect.

Statistical IDS can be used to classify network activity into normal and abnormal categories. To test the effectiveness of statistical IDS in SCADA control system networks a set of false response injection exploits were developed for the water tank control system in the MSU SCADA Security Laboratory. A neural network was used to classify network transactions as normal or abnormal.

A. IDS Dataset

An experimental dataset was developed to train and test the IDS. The dataset included data from normal operation and 6 types of false responses from the MSU SCADA Security Laboratory water tank control system described in section II.C.

The water tank control system uses HH, H, L, and LL set points in the RTU to manage the system. The HH set points are alarm levels, the water level exceeds HH or is lower than LL an alarm is triggered. The H and L set points control the water pump. If the water level exceeds H, the pump is turned off. If the water level is below L, the pump is turned on.

Each type of injected false response was a false water level. The water tank MTU periodically sends a command to the RTU to read the water level in the water tank. The water level is returned as a single-precision floating point number. The first type of false response was a negative water level. In normal practice the water level should always be greater than or equal to 0. The second type of false response was a water level greater than the HH set point. The water level should rarely approach or exceed the HH set point. The third type of false response was a water level which was greater than the H set point, but less than the HH set point. In normal operation, the water level often slightly exceeds the H set point value due to time lag between reading the water level and the pump being turned off. The fourth type of false response was a water level which was less than the L set point and greater than the LL set point. In normal operation, the water level often slightly under cuts the L set point value due to time lag between reading the water level and the pump being turned on. The fifth type of false response was a water level less than the LL set point. The water level should rarely approach or exceed the LL set point. The sixth type of false response is a random water level value.

The experimental dataset included commands and responses from the system running in normal mode for approximately 4 hours. The data set from normal operating conditions includes 12,000 captured network packets. A second experimental dataset includes commands and responses captured during a man-in-the-middle attack. The man-in-the-middle node injected false responses every randomly after every 3rd, 4th, or 5th command. This dataset includes 5,000 captured packets. A third experimental dataset includes commands and responses captured during a DOS enabled false response attack. In this dataset the rogue node was configured to always inject a false response, i.e. never allowing the addressed slave to respond. Finally, in a fourth dataset a replay attack was created which substituted previously captured water level response packets for valid water level response packets.

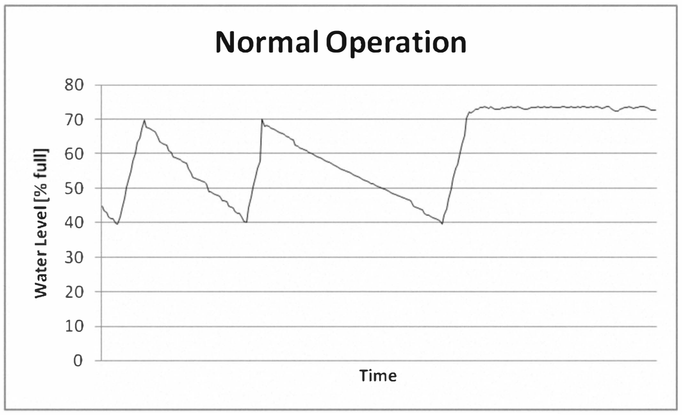

Fig. 3: Water Level Trend during Normal Operation

Fig. 3 shows the first 250 responses from the normal operation dataset. The water level can be seen rising and falling with L set point of 40% and H set point of 70%.

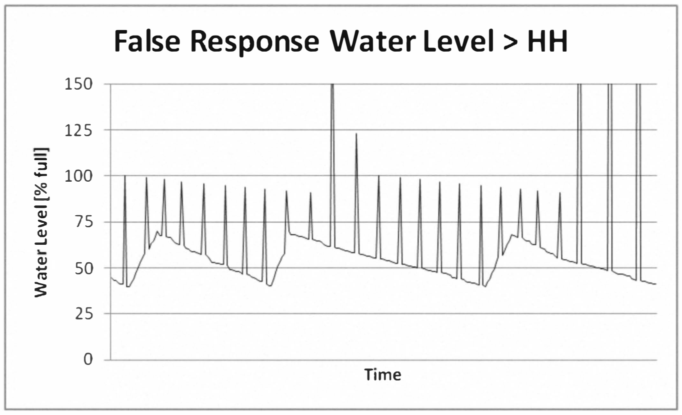

Fig. 4: Water Level Trend with False Responses > HH Set Point

Fig. 4 shows the first 250 responses from the dataset which includes a man-in-the-middle attack injecting responses greater than the HH set point which is set to 80%.

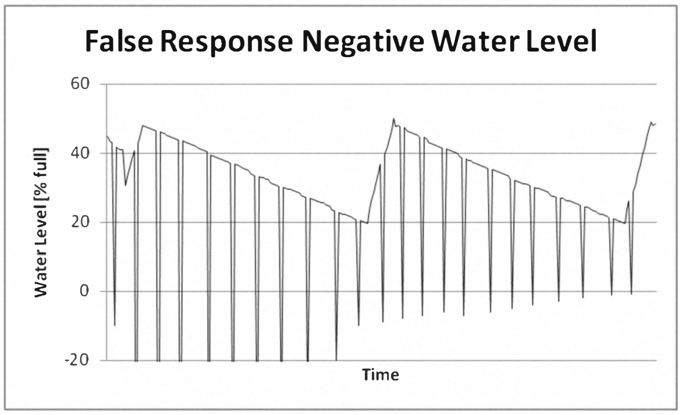

Fig. 5: Water Level Trend with Negative False Responses

Fig. 5 shows the first 250 responses from the dataset which includes a man-in-the-middle attack injecting negative water level responses.

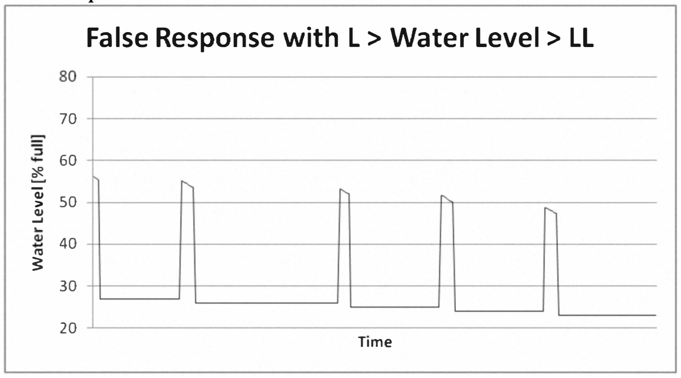

Fig. 6: Water Level Trend with False Responses, L > Water Level > LL

Fig. 6 shows the first 250 responses from the dataset which includes a DOS based attack injecting false water level responses where the injected water level is below the L set point and above the LL set point. The L set point is set at 40% and the H set point is 70%. Most of the data shown are false responses. Only the peaks of the spikes in the graph are from valid responses.

Plots are not provided for the other datasets.

B. *IDS Input Features*

We used a back propagation algorithm to build a 3 stage neural network which classifies SCADA network transactions as normal or abnormal based upon 4 input features.

The first input feature was the water level expressed as a percentage. Fig. 4 and Fig. 5 show water levels from the water storage tank control system under attack. The water level normally trends between the L and H set points. The false responses are shown as spikes in the data. The water level feature should easily identify the large spikes as violations of the water level trend and classify these packets as abnormal. The false response datasets which inject water levels above the H set point and below the L set point, but not beyond the HH and LL alarm levels, are less obvious and therefore more difficult for the neural network to correctly classify based on only the water level input feature.

The second input feature was command response frequency. During normal operating conditions the control system network includes commands and responses in pairs. In many of the attack scenarios a command will be followed by 2 responses; a false injected response, and a valid response from the addressed slave.

The third input feature is the mode of operation of the control system. The mode of operation can be set as either automatic or manual. In automatic mode, the RTU automatically maintains the water level between the L and H set points. In manual mode, the RTU waits for input from the MTU to control the water level. This input comes in the form of turning the water tank pump on and off. For all of the attacks used for this work the mode of operation is set to automatic. This feature is reserved for more complex data sets.

The fourth input feature is the state of the water tank pump. As previously mentioned the pump can be set to on or off. The pump state can be changed by a command received from the MTU (or an illicit command from an network penetrator) or can be changed automatically by the RTU when the control system is in automatic mode. The water level should increase when the pump is on and decrease when the pump is off.

C. *IDS Experimental Results*

Collected data from laboratory experiments were used to train and test the neural network IDS developed.

Table 1: Classification Results for Man-in-the-Middle Response Injection

Exploit Scenario	% False Positive	% False Negative	Accuracy
Negative Response	0.0%	0.0%	100.0%
Response > HH	4.5%	0.0%	95.5%
HH > Response > H	2.3%	3.0%	94.7%
L > Response > LL	2.4%	3.0%	94.6%
Response < LL	3.2%	0.0%	96.8%
Random Response	6.2%	8.9%	84.9%

For each dataset a Perl script was used to classify individual network packets as normal or abnormal before training the neural network. The neural network was trained with 80% of the available samples and 20% percent of the available samples were used to validate the neural network.

Table 1 shows the classification results from the man-in-the-middle response injection exploits. Results included 100% accuracy for negative water level values. This is expected as a negative value is always abnormal. For the *Response > HH, Response > HH, L > Response > LL, Response < LL* exploit scenarios accuracy ranged between 94.6% and 95.5%. The inaccuracies are primarily related to the inability of the neural network to classify false responses which are very close to the L, H, LL, HH set points depending upon the exploit scenario. The random water level response shows the worst accuracy with 84.9%. This is due to the fact that the some many of the random false responses are within L and H set points and therefore indistinguishable from normal data.

Table 2: Classification Results for DOS Based Response Injection

Exploit Scenario	% False Positive	% False Negative	Accuracy
Negative Response	0.0%	0.00%	100.0%
Response > HH	0.1%	1.2%	98.7%
HH > Response > H	1.2%	2.0%	96.9%
L > Response > LL	1.0%	1.3%	97.7%
Response < LL	0.2%	1.0%	98.9%
Random Response	8.2%	1.0%	90.9%

Table 2 shows the classification results from the DOS based response injection exploits. IDS accuracy is similar to the results from the man-in-the-middle response injection exploits. Results again included 100% accuracy for negative water level values. For the *Response > HH, Response > HH, L > Response > LL, Response < LL* exploit scenarios accuracy ranged between 96.9% and 98.9%. These results represent an improvement over the man-in-the-middle accuracy results. The DOS dataset included continuous false data injection rather than random false data injection. Results indicate the neural network performs better when classifying the continuous injection attacks.

Table 3: Classification Results for Replay Based Response Injection

Exploit Scenario	% False Positive	% False Negative	Accuracy
Replay Attack	45.1%	42.7%	12.1%

Table 3 shows the classification results from a replay based response injection exploits. Accuracy for this dataset was very low. This indicates a neural network with the aforementioned input features is unable to detect a replay attack. This is expected since the replay data looks exactly like real data.

V. FUTURE WORK AND CONCLUSION

A. Future work

This paper applied a back propagation algorithm to build the neural network for the intrusion detection system. Because the back propagation algorithm is supervised learning, training requires target output values. In future work, an unsupervised learning algorithm will be used to build the neural network using to support training only using input vectors.

Also, more work is required to develop input features for the neural network. This paper presents preliminary work. SCADA control systems control physical processes which must obey the laws of physics. For the water tank control system this means that water level may only change at rates based upon tank capacity and pipe diameters. Methods which train the neural network based upon physical properties will likely lead to more accurate classification results. Additionally, neural network inputs can be extended to include the n-most recent water level values to allow the neural network to recognize trends in the water level or other features.

Replay attacks prove particularly difficult to detect. Input Research to indentify features which improve replay detection accuracy to acceptable levels is required. Improvements may require upgrading SCADA control system networking protocols to include sequence numbers or time stamps or other mechanisms to assist in identification of replayed packets.

Finally, a Bayesian network will be used in future work to compare the accuracy to the neural network results.

B. Conclusion

In this paper we presented a set of command and response injection attacks and denial of service (DOS) attacks on SCADA control systems. Attacks were targeted at a commercial SCADA control system and a SCADA network transaction data logger was used to capture network traffic associated with the attacks. Finally, captured exploit network traffic was used in combination with captured traffic from the SCADA control systems running normally to train and validate a neural network based intrusion detection system which leverages knowledge of the physical properties of the controlled system to detect false response injection attacks. IDS results show that a neural network is a promising mechanism for detecting false responses however future work is required to developed input features which improve IDS accuracy and to find features capable of detecting replay attacks.

REFERENCES

[1] Santorelli, S. Who is looking for your SCADA infrastructure? March 2009. Published online. Sample June 30, 2010.
http://www.team-cymru.org/ReadingRoom/Whitepapers/2009/scada.pdf

[2] http://www.nerc.com/page.php?cid=2|20

[3] Ronald L. Krutz, Securing SCADA Systems. Wiled Publishing, Inc. Indianapolis, Indiana, November 28, 2005

[4] Oman, P., Phillips, M. Intrusion Detection and Event Monitoring in SCADA Networks. In Critical Infrastructure Protection. Spring Boston 2007.

[5] Monticelli, A. State Estimation in Electric Power Systems: A Generalized Approach. Springer. 1999

[6] Wang, Y., Statistical Techniques for Network Security, Modern Statistically-Based Intrusion Detection and Protection. IGI Global. October 2008

[7] Amit Kumar Choudhary, Akhilesh Swarup, Neural network approach for intrusion detection. Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Seoul, Korea, 2009

[8] Morris T., Srivastava A. Reaves B., Pavurapu K., Abdelwahed S., Vaughn R., McGrew W., Dandass Y. Engineering Future Cyber-

Physical Energy Systems: Challenges, Research Needs, and Roadmap . 2009 IEEE North American Power Symposium. October 4-6, 2009, Starkville, MS

[9] 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

[10] Martin T. Hagan, Howard B. Demuth, Mark H. Beale, Neural network design, PWS Pub, 1996

[11] Ke Wang., Salvateore J. Stolfo. Anomalous Payload-based Network Intrusion Detection. Lecture Notes in Compuer Science, Volume 3224/2004

[12] Slay, J., Miller, M. Lessons Learned From the Maroochy Water Breach. In IFIP International Federation for Information Processing. Volume 253. Critical Infrastructure Protection, eds. E. Goetz, S. Shenoi. Boston Springer. Pages 73-82. 2008.

[13] Morris, T., Pavurapu K. A Retrofit Network Transaction Data Logger for SCADA Control Systems. Submitted for publication 2010 IEEE SmartGridComm.

[14] Falliere, N., Exploring Stuxnet's PLC Infection Process, http://tinyurl.com/3ykx85p