SCADA Full State Network Intrusion and Malfunction Detection System

Mehdi Anisheh, Dale Lindskog, Pavol Zavarsky, Ron Ruhl
Concordia University College of Alberta
manisheh@student.concordia.ab.ca
{dale.lindskog, pavol.zavarsky, ron.ruhl}@concordia.ab.ca

Abstract - Industries are highly dependent on reliable, accurate and automated control systems to monitor equipment that are critical to their operation. Supervisory Control And Data Acquisition (SCADA) is the most advanced control system which is being widely used in industries and it is an attractive target for threat agents. Host based and network based intrusion prevention systems (IPS) and intrusion detection systems (IDS) are the best existing solution to improve SCADA security against cyber attack. This paper describes the evolution of network intrusion detection systems (NIDS) from signature based NIDS to a novel NIDS based on the general state of the SCADA control system. One of the most recent NIDS is Modbus/DNP3 state-based NIDS, which is a significant improvement toward detecting complicated attacks on SCADA systems. In this paper we investigate the pros and cons of Modbus/DNP3 state-based NIDS and introduce a new technique to address the limitations and weaknesses of this existing technology. We call our proposed enhancement the SCADA full-state Network Intrusion and Malfunction Detection System (NIMDS). It functions by monitoring SCADA's behavior and double checking the control process.

Keywords - SCADA; threat agent; cyber attack; IDS; IPS; NIDS; NIMDS; state; state-based; Full-state;

I. INTRODUCTION

SCADA systems are used to control remotely located equipment communicating via internal private networks or external networks such as the Internet. Unfortunately, a connection to Internet exposes the SCADA systems to cyber attacks [1]. The ability of SCADA systems to control remote equipment makes it a target for threat agents who may attempt to exploit its vulnerabilities and cause destruction to equipment or its operation in high tech and vital industries.

Regardless of how secure a system is, a system could still be vulnerable to structured attacks because attackers are constantly updating their methods of exploitation to circumvent security safeguards. Stuxnet is a recent example of a structured attack which has specifically been used to attack SCADA systems in nuclear power plants. It bypassed detection by packet filters, signature based intrusion detection and prevention systems (IDPS) and was capable of reaching the SCADA Master and taking control of it, while remaining concealed and undetectable [2][3].

By detecting an attack at an early stage, it is possible to mitigate the attack, by disabling the compromised system, or activating a planned response program. For minimal disruption, options such as switching to a redundant SCADA system or disabling the SCADA system and resorting to manual control of critical equipments are possible responses.

Host based and network based intrusion detection systems (HIDS and NIDS) are common tools to prevent or mitigate massive damage to operations, but they are too immature to be widely deployed in SCADA systems [4].

Moreover, a HIDS is a single point of failure (SPOF) as it fails to detect an intrusion when the host is compromised. Signature based NIDS are not an SPOF in this sense, and can potentially identify an attack by analyzing the traffic and looking for the attack signature. However, as the Stuxnet example demonstrates, and as is well known, signature based intrusion detection is easily evaded.

All traffic in a SCADA network is either a command from the SCADA master to control devices, or data from sensors to the master and there is no attack signature to be detected by this type of NIDS. The Modbus/DNP3 State-based Intrusion Detection System [5] is more advanced than a signature based NIDS. It monitors the states of a SCADA system. It can detect a complex attack by analyzing the traffic, keeping track of the states of the SCADA system and comparing it with critical state rules stored in NIDS's database. A complex attack is constructed from a set of commands from a compromised SCADA system to the control devices which would lead the SCADA to enter a critical state. So while each single command could be seen as a legitimate, and thus no attack would be detected by a signature based NIDS. Modbus/DNP3 NIDS updates the states in its internal representation of the SCADA to compare with critical state rules in the database. Critical state rules represent the combinational state of related control devices, and an attack is detected when this combinatorial state creates a critical state for the SCADA system. Consider the steam dryer depicted in figure 1. This is a good example to explain how differently these two types of NIDS's work. Paper machines in pulp and paper industries have several rolling dryers to press and dry the pulp to produce paper.

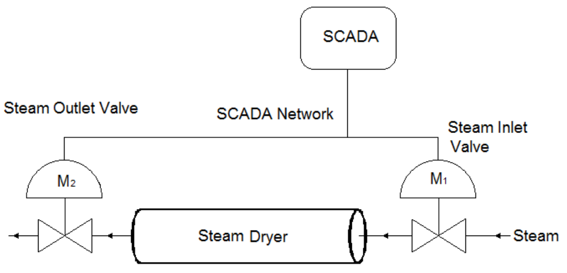

***Figure 1*: SCADA controls steam flow in a Steam Dryer**

In an attack to explode a steam dryer, an attacker might force SCADA to send two commands in sequence, first to open the dryer's steam inlet valve and second to close the dryer's steam outlet valve. The signature based NIDS will consider both packets legitimate and won't trigger any alert. On the other hand, for the Modbus/DNP3 NIDS, the first command updates the NIDS internal state, setting the inlet valve to "Open", while the second command sets the outlet valve to "Close". This is not consistent with the related valve's state, and therefore the NIDS alerts that SCADA is falling in to a critical state.

Note that detecting an attack using the Modbus/DNP3 NIDS is limited to critical state rules and the attack pattern. Moreover, Modbus/DNP3 depends on active monitoring (see below), and this is another limitation, since it imposes extra traffic on the SCADA network and, also makes it (the NIDS) itself vulnerable to cyber attacks.

The objective of this paper is to enhance NIDS capability to detect intrusion and malfunction in SCADA. The proposed technique also allows the NIDS to operate transparently; therefore it won't add any traffic to SCADA network and also will avoid being a new target for the attackers. Our proposed SCADA Full-State Network Intrusion and Malfunction Detection System (NIMDS) verifies the integrity of the control system by passively monitoring the control process, keeping track of transitions and the resulting state of the control system, in order to detect a malfunction. It passively collects data from sensors and computes the expected response to a state change, and then compares it with SCADA's actual response. Any difference between the expected response and the actual response is an indication of intrusion or malfunction. The proposed NIMDS is consists of two components: a data mining unit and a processing unit. The data mining unit employs a packet analyzer and pre-processor and eavesdrops on the network, passively collecting and processing the required data, which is then stored in a database which in turn will feed the processing unit. The processing unit simulates and analyzes the entire state of SCADA and stores simulation data and calculated states in two other databases for further comparison.

In sections II, III and IV of this paper, we will describe the limitations of the Modbus/DNP3 NIDS, and compare it to the proposed SCADA full-state NIMDS. In section V, a summary of our proposed SCADA full-state NIMDS will be given and the structure will be discussed further.

II. FALSE NEGATIVE & FALSE POSITIVE

Detecting an attack in Modbus/DNP3 NIDS is only possible when the attack pattern corresponds to a critical state rule stored in a database. For example, consider an attack to disrupt operation by forcing a SCADA to open both valves of the dryer shown in Figure 1, in order to lessen the steam pressure through the steam dryer or to increase the dryer's temperature. This won't be detected by Modbus/DNP3 NIDS because, first, there is no inconsistency between the valves' states and second, it doesn't know why both valves are open or closed. Critical state rules can be defined to raise an alert when two or more related control devices are in a specific position which is critical for operation. In our steam dryer example, the critical state occurs when the outlet valve is closed while the inlet valve is open, since this state will cause the dryer to explode. The other state can not be defined as a critical state because SCADA controls the dryer's temperature through these valves. This false negative would be detected in the SCADA full-state NIMDS, as it will be programmed to know the entire process and status of the control systems. In above mentioned example, the NIMDS acts as a perfect mirror of the SCADA and would raise an alert at the first step, since it is not an anticipated command from the SCADA master to open the dryer's steam inlet valve. For the same reason, it would also raise another alert if SCADA is forced to open the dryer's outlet valve. In summary, Modbus/DNP3 NIDS only examines the consistency in SCADA's commands to the field devices, while the NIMDS examines the logic between the states and SCADA's response to each state. Modbus/DNP3 IDS monitors that "portion" of the field devices for which knowledge of their state is required determine whether the SCADA is in a critical state [5]. Therefore, it could miss an attack, e.g. because it is not monitoring that particular device, or because it doesn't know the attack pattern. It might also generate false positives, by raising an alert when it discovers inconsistency between states, such as when the SCADA closes the inlet valve and opens the outlet valve in order to lessen the dryer's pressure. Figure 2 shows a moisture sensor installed on a paper production line to read the dryness and send data to SCADA. SCADA's decision to turn the state of each dryer on or off is based on the data collected from the specific sensors. If, at some point, it is not dry enough, it should bring the dryer in operation; therefore, the SCADA would first send a command to open the dryer's steam outlet valve, and then another command to open the dryer's steam inlet valve. These two commands issued from SCADA to the valves will be processed differently in each type of NIDS's. The Modbus/DNP3 NIDS looks for inconsistency in valves' states and it doesn't care why those states have changed. It checks to assure both valves are open or closed and, therefore, if an attacker forces SCADA to open or close both valves the intrusion will not be detected.

In this example, the Modbus/DNP3 NIDS starts its investigation when SCADA sends the first command to open the outlet valve. It then waits to see the second command to the inlet valve, to be able to compare the states of those valves with the critical state rules.

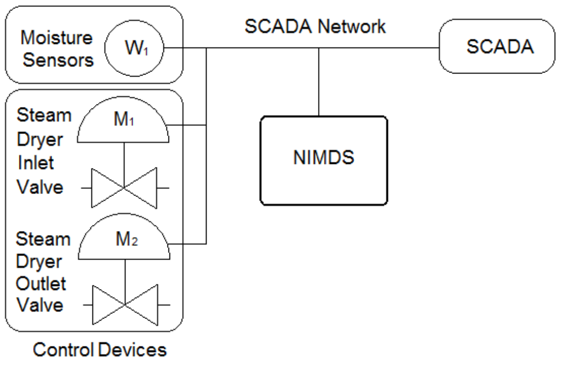

***Figure 2:* SCADA uses Moisture Sensor to check paper's dryness and two electric motor actuated valves to control steam flow in each dryer.**

By contrast, our proposed SCADA full-state NIMDS can figure out the process order to each valve. It determines the logical response to each state, and keeps track of the status of the all control devices. Since it is a live representation of SCADA, when, for example, it receives data from a sensor (shown in figure 2), it knows exactly which dryer should come into action; therefore, it knows the sequence of opening the respected valves (i. e. both valve's state should be changed to "Open" state). In the SCADA full-state NIMDS, the data sent by sensors is the starting point of the investigation. It expects SCADA to open the outlet valve first and inlet valve after, and if SCADA for any reason behaves differently, it will raise an alert.

III. ACTIVE MONITORING

The Modbus/DNP3 NIDS detection process requires knowledge of the state of related devices, in order to compare them with its critical state. If the attacker forces the SCADA to open only the dryer's inlet valve, Modbus/DNP3 NIDS will change its state to "Open" and, to compare the state with the critical state rules, it needs to know the state of dryer's outlet valve as it is related to dryer's inlet valve. In this case it has to query the outlet valve to retrieve and examine its updated state on critical state rules.

Active monitoring adds load to a SCADA network and any synchronization with SCADA's equipment from the Modbus/DNP3 NIDS could cause a delay or data loss on the SCADA side, which cannot be tolerated [6]. Furthermore, with active monitoring, the NIDS itself is vulnerable to attack: the attacker can find it while sniffing the network, and can initiate an attack against the NIDS, perhaps leveraging it to attack SCADA systems.

The SCADA Full-state NIMDS monitors the system passively and it doesn't need to query any of the control devices. If there is any need to query a device, SCADA is the one that sends the query because SCADA and NIMDS use exactly the same logic and if the NIMDS needs to know the state of a device, it is also needed by SCADA. If the SCADA should send the query and it doesn't, there would be an alert. Passive monitoring will allow the NIMDS to be invisible to attacker and it won't add any traffic in SCADA's private network.

If, for any reason, a feedback from NIMDS to SCADA or any other industrial control system is required, it could be done by connecting a discrete device to the NIMDS with an IP address to transfer the requested feedback to the control system.

IV. DEPENDENCY

A NIDS should operate independently to data in the SCADA master, as it is possible for an attacker to manipulate data on a compromised system. The Modbus/DNP3 NIDS is required to query the SCADA system to synchronize its virtual state with SCADA's physical state. In our previous example, it might query the steam dryer's outlet valve to update its state when SCADA sends a command to open the inlet valve. The attacker can respond to NIDS's query that the outlet valve is open while in fact it is closed. As the Modbus/DNP3 NIDS sees both valves are open, it would consider this a normal state and won't raise any alert, while those valves are really in critical state because of a malicious attack.

SCADA full-state NIMDS is designed to compute the entire states of control devices. It passively collects unprocessed data directly from the sensors when they communicate with the master and can process the data to calculate the state of each control device and update its internal simulated states. The simulated states in SCADA full-state NIMDS is exactly the same as the actual states in SCADA. Therefore it does not need to read data from SCADA and the simulated data will be used to measure SCADA's functionality.

V. SUMMARY OF SCADA FULL-STATE NIMDS

Installing a dedicated IDS on a SCADA's private network can help us to monitor the SCADA's functionality. Our proposed SCADA full-State NIMDS passively collects data sent from field sensors and/or other SCADA systems and computes the expected response (which is an expected command from SCADA to these control devices) and compares it with SCADA's actual response. If there is any discrepancy between expected and actual response, it would be an indication of a malfunction or intrusion. If SCADA is compromised by an attacker, the data and the state of control devices being displayed on HMI cannot be trusted as the attacker could have manipulated to mislead the operators to believe that the operation is normal. In this case, it is almost impossible to detect the attack by conventional NIDS or even Modbus/DNP3 NIDS.

The SCADA full-state NIMDS will be programmed to operate similarly to the SCADA it is monitoring, with the same algorithm used to compute the expected response to each state. Since it is located on the same network, it has access to original data coming directly from sensors, and has the capability to detect any intrusion or malfunction in control system. As the SCADA full-state NIMDS passively monitors the control system, it won't add any extra load or traffic, and therefore won't slow down or interrupt SCADA's private network, and also will avoid being a new target for the attacker.

VI. SCADA FULL-STATE NIMDS STRUCTURE

SCADA full-state NIMDS performs two major tasks in order to accomplish its mission. The first task is extracting the required data from the SCADA network which will be accomplished by a packet analyzer and pre-processor. The second task is processing the data, which includes simulating and analyzing the state of both SCADA and the simulator. The following diagram (Figure 3) represents NIMDS's structure in detail and each unit is described in the following section.

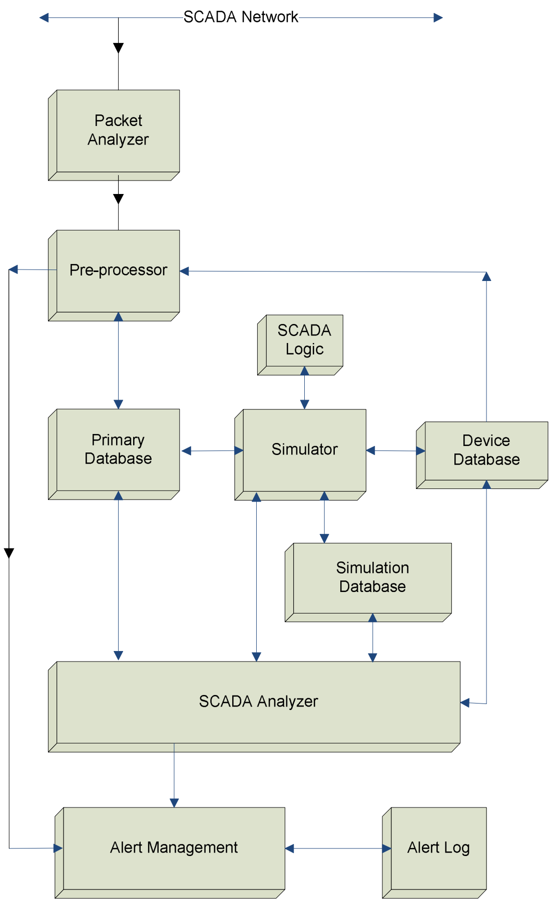

Figure 3**: NIMDS structure**

VII. PACKET ANALYZER

The packet analyzer eavesdrops on the SCADA network to collect the required data, and stores it in a database. SCADA network traffic is straightforward to analyze. Most of the IP enabled SCADA components are using UDP to communicate. Each packet would have a source and destination IP address and a value. A packet analyzer might even utilize snort rules to extract data from packets. It would be easy to determine which device is the sender by looking at the source address and which the receiver device by checking the destination address. If the sender is the SCADA master, the packet is carrying a command to a control device, and if the receiver is SCADA, then the packet is data from a sensor to SCADA.

IIX. PRE-PROCESSOR

The pre-processor is responsible of validating all the data collected by the packet analyzer. It has access to a device database to verify whether the traffic is coming from the right source, and going to the right destination. The pre-processor can trigger an alert when it identifies invalid network traffic. It could also detect spoofed traffic by checking the time stamp for every piece of data captured by packet analyzer. It will set certain fields in the corresponding record in the database for suspicious traffic to help SCADA analyzer to identify any spoofed packet. Traffic will be identified as suspicious, when it comes from a source in an unacceptably short time interval.

IX. PRIMARY DATABASE

We will call the database, referred to in previous sections, the 'primary database', in order to distinguish it from another database, which we will call the 'simulator database'. The purpose of the simulator database is to store the calculated expected response from SCADA to the control devices. The pre-processor records data collected (by the packet analyzer) from the network in this primary database, and there would be a corresponding record for each transaction between sensors, control devices and SCADA which will be refreshed by the pre-processor. Each record consists of the following fields:

Sender	Receiver	Value	Time Stamp	Suspicious Bit	Suspicious Value	Process Bit

- *Sender and Receiver:* the packet analyzer will be programmed to find the sender and receiver address based on the source and destination address. The pre-processor checks the legitimacy of the packet by checking the source and destination addresses. The Simulator and SCADA Analyzer can determine if the packet is carrying data from a sensor (which might change system’s state) or it is a command from SCADA to a control device. If it is data from a sensor, the simulator will process it to compute the expected response from SCADA, and if it is a command from SCADA, the SCADA analyzer will compare it with its expected response to check for deviation.

- *Value:* the current state of a device can be determined by the value. For example, if it is a relay, the corresponding value will be one or zero to indicate “On” or “Off”, and if the record is for a pressure sensor, the value is showing current pressure. The value will be used for simulation when the packet is data from a sensor, and it will be used for comparison if the packet is a command from SCADA.

- *Time stamp:* the packet analyzer sets the time stamp for each record in database when it captures a packet. This field will be used in both pre-processing and in the SCADA analyzer for the investigation process. Also, the simulator calculates the estimated response time in the simulation database, based on the time stamp in primary database. The pre-processor uses this time stamp to detect suspicious traffic, and SCADA analyzer checks the time stamp to determine if SCADA has had enough time to respond to the last request. If the difference between current time and time stamp is longer than its expected time, the SCADA analyzer will perform a comparison, otherwise it simply ignores it.

- *Suspicious bit:* When the packet analyzer captures new data, the pre-processor compares the current time with the time stamp in the corresponding record to verify that the captured data has been sent within an acceptable time interval. If the packet analyzer captures two packets from a device in a short period of time, the pre-processor sets the suspicious bit and writes the last captured value in the “suspicious value” field (see below), which will be processed later by SCADA analyzer.

- *Suspicious Value:* this field contains the value from the latest captured packet which, when set, has been identified as suspicious by the pre-processor. The SCADA analyzer can identify a spoofed packet when the suspicious bit is set. It calculates the expected response by using data from the “suspicious value” field, and compares it with calculated response using data from “value” field. It will trigger an alert if the responses are not identical.

- *Process bit:* process bit will be set when packet analyzer writes to each record and simulator resets it when the data has been used for simulation and skips it in next trace when it is reset. SCADA analyzer also uses this bit for comparing a captured response from SCADA to a simulated expected response in Simulation database and resets it when comparison is completed.

X. SIMULATOR

The simulator follows the same logic as the SCADA master. It is stored in the SCADA Logic unit and uses the primary database as input. The simulator will use loop execution on the primary database, which is similar to SCADA’s ladder logic: it traverses and traces the entire database from left to right (or top to bottom) and calculates the expected response for each state and records it in Simulation and Device databases which will then be compared with SCADA’s response at the right time (“time stamp” will be used to determine the right time: see Section IX).

The simulator keeps the simulation states in a simulation database. It also keeps the state of the entire devices (controllers and sensors) in a device database, which will allow the NIMDS to determine the latest state of each device passively, i.e., without querying and therefore creating extra traffic on the SCADA network.

XI. SCADA LOGIC

The simulator works directly with SCADA logic to compute and simulate SCADA’s state. It contains an exact copy of the SCADA program, which could be a translated version or an identical SCADA ladder logic program. NIMDS’s capability to accept the identical program will make it powerful, flawless, easy to use and flexible. It can be equipped with this ability by adding a dedicated complier to SCADA logic to translate the program for the simulator.

We can use the steam dryer example to present a better clarification of SCADA logic. As shown in figure 4, when SCADA reads the moisture sensor, in the first step it compares the data with its internal database to determine whether the value has changed. In case of any changes, SCADA utilizes some rules to sequentially open or close electric motorized actuator valves (M1 and M2) to control the steam flow in each dryer. It sends a command to each valve and also records the state of valves in its internal database for future use.

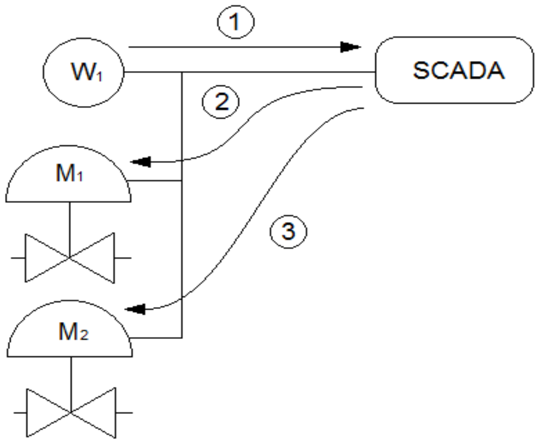

***Figure 4*: Sequence of a process**

Table 1 shows a sample rule set which, let us imagine, is being used by both SCADA and NIMDS. For example, if SCADA reads 45% from W1, it keeps both inlet and outlet valves half open.

***Table 1*: A sample rule set**

Moisture (W1)	Inlet valve position (M1)	Outlet valve position (M2)
Less than 20%	Closed	Closed
20% - 40%	25% Open	25% Open
41% - 60%	50% Open	50% Open
61% - 80%	75% Open	75% Open
Over 80%	Full Open	Full Open

The simulator uses the same rules from SCADA logic to calculate the expected response; therefore it sets two records (in the simulation database) with the calculated value for each valve (which is 50% in this example) and this data will be used for comparison.

XII. DEVICE DATABASE

The device database keeps the states of the devices (such as sensors and control devices) to enable the simulator to calculate the expected response without querying any devices in the network. The simulator updates the state of each device based on device's current state (which is stored in device database) and data stored in primary database. The device database allows the simulator to have access to the latest state of a device when it requires that for computing its simulation of the process. The following is the structure of each record in Device database.

Device Record

Device Address	Device Type	Time Stamp	Actual Value	Calculated Value	Process Bit

- *Device Address:* Each device has a unique IP address which will be used to identify that device. The pre-processor uses the address to validate the extracted data. If it finds a packet for which neither the source nor destination address can be found in Device database, it will raise an alert.

- *Device Type:* Device type records whether a device is a sensor or a control device. If the device is a sensor, the simulator doesn't need to compute the calculated value, and the SCADA analyzer also doesn't need to do the comparison.

- *Time stamp:* This field defines the time which SCADA is expected to send a command to the respected control device.

- *Actual Value:* When SCADA sets the state of a device or a device state is changed in a process the actual value will be set.

- *Calculated Value:* This field is set by simulator in response to state changes in a process.

- *Process bit:* the simulator sets the process bit when it changes the calculated value, and the SCADA analyzer resets the bit when it compares the calculated value with the actual value.

XIII. SIMULATION DATABASE

The simulation database is where all the expected and actual responses from SCADA are stored. The simulator computes the anticipated response and stores it in this database, to be compared later, by the SCADA analyzer, with the real response. The database has the following fields:

Simulation Record

Device Address	Time Stamp	Expected Value	SCADA Value	Process Bit	Compare Bit	Error Bit

- *Device Address:* In the controlling system only, SCADA sends commands to control devices, and therefore we do not need to keep the source address. Device address is the address of a device which is receiving commands from SCADA.

- *Time Stamp:* this field holds the maximum time interval that SCADA requires to fire a control signal to the respected device.

- *Expected Value:* the simulator saves the calculated response in this field. This value is a set point in a control device, which SCADA is expected to set in response to a state change in the control process.

- *SCADA Value:* the SCADA analyzer records SCADA's actual response in this field and compares it with expected value; a difference indicates malfunction or intrusion.

- *Process bit:* When the simulator calculates the expected response, it also sets this field to highlight that this record needs to be processed by the SCADA analyzer, and that the SCADA response is pending.

- *Compare bit:* This bit will be set by the SCADA analyzer when the comparison is performed.

- *Error bit:* This bit will be set by the SCADA analyzer when the expected value is not equal to SCADA value, which to repeat, is an identification of intrusion or malfunction.

XIV. SCADA ANALYZER

The SCADA analyzer monitors SCADA's functionality by comparing the processed data with actual data collected from the network. It reads the actual SCADA response from the primary database and compares it with the simulated data in the simulation database. It also analyzes the data with the suspicious flag set, to reveal spoofed packets, by calling simulator to calculate the expected response for both "Value" and "Suspicious Value". The SCADA analyzer will raise an alert if the responses are not unique. The SCADA analyzer checks the Simulation database when it finds a command from SCADA to a control device stored in primary database. It also looks for the corresponding data in the primary database when the time stamp in a record in the simulation database is about to expire. There would be an alert if SCADA has not set a control device within the expected time interval. The SCADA analyzer resets all the related flags, such as the comparison bit and the process bit, when it finishes its task on each record.

XV. ALERT MANAGEMENT

The alert management unit is responsible for presenting the alerts and their criticality to the process. It receives information from the SCADA analyzer and pre-processor to show the failure and its consequences on the control system.

XVI. ALERT LOG

Alert log unit keeps the alerts in a database, with each alert describing the event in detail. It records the reason for triggering an alert, including the device name, time, what was supposed to happen and what actually has happened. The log can help the process engineer to track down the failure and make proper corrections to the system.

XVII. SYNCHRONIZATION

One of the main advantages of our proposed SCADA full-state NIMDS is its passive operation, which allows it to collect the required data without querying any control device for synchronization. If both the NIMDS and SCADA start their operation at the same time, there is no need for any synchronization and NIMDS can start its normal operation without concerns about any false positives or false negatives. But what if the NIMDS starts to operate after the SCADA system has already started? The easy solution is to send a query to each device to find the state of each device, but then our proposed NIMDS is no longer passive. Moreover, this approach may not work, as the devices may change state (perhaps even several times) during the synchronization process. Another option is to use a starter program to passively set the state of each device while communicating with SCADA. The best solution is to allow the NIMDS to continue its normal operation from the zero point. It will raise lots of alerts in the beginning, but after a few seconds (or in the worst case a few minutes), as the SCADA master talks to all control devices and sensors, it will have the entire device's state, and synchronization will be completed without any query.

XIIX. CONCLUSION AND FUTURE WORK

We have introduced a new technique for SCADA system intrusion and malfunction detection. The proposed technique can enhance detection: it promises to be accurate, flexible, and independent; it does not rely on attack patterns or signatures. It only needs to understand the SCADA ladder logic software to monitor SCADA's functionality. SCADA full-state NIMDS is a detection system, and is not intended to replace other protection and prevention methods, which of course should also be implemented in a SCADA network.

REFERENCES

[1] Athar Mahboob and Junaid Zubairi: Intrusion Avoidance for SCADA Security in Industrial Plants, IEEE, 2010

[2] Gregg Keizer: Stuxnet code hints at possible Israeli origin, Computerword, Sep 2010 http://www.computerworld.com/s/article/9188982/Stuxnet_code_hints_at_possible_Israeli_origin_researchers_say?taxonomyId=82

[3] Gregg Keizer: Is Stuxnet the best malware ever?, Computerworld, Sep 2010 http://www.networkworld.com/news/2010/091610-is-stuxnet-the-best-malware.html

[4] Jared Verba & Michael Milvich : Idaho National Laboratory Supervisory Control and Data Acquisition Intrusion Detection System (SDACA IDS), IEEE International Conference on Technologies for Homland Security, May 2008

[5] Igor Nai Fovino, Andrea Carcano, Thibault De Lacheze Murel, Alberto Trombetta and Marcelo Masera: Modbus/DNP3 State-based Intrusion Detection System, 24th IEEE International Conference on Advanced Information Networking and Applications, 2010

[6] Ning Cai, Jidong Wang and Xinghuo Yu: SCADA System Security: Complexity, History and New Developments, School of Electrical and Computer Engineering, RMIT University, Melbourne, Australia