BLOOM FILTER BASED INTRUSION DETECTION FOR SMART GRID SCADA

Saranya Parthasarathy and Deepa Kundur
Department of Electrical & Computer Engineering
Texas A&M University
College Station, TX 77843 USA

ABSTRACT

In this paper, we propose a distributed, light-weight and fast intrusion detection approach suitable for implementation across multiple resource constrained SCADA field devices in the smart grid. The predictable and regular nature of the SCADA communication patterns is exploited to detect intrusions in the field devices. The novel approach is anomaly-based, uses the Bloom filter data structure for memory efficiency and incorporates the physical state of the power system for greater robustness. The proposed method is tested using MODBUS protocol used for communication between a SCADA server and field devices in a SCADA system.

Index Terms— Bloom filter, intrusion detection, smart grid SCADA IDS.

1. INTRODUCTION

Supervisory control and data acquisition (SCADA) systems are industrial control systems responsible for distributed monitoring, control, collection and analysis of real-time data. Vulnerabilities in these systems have heightened in the recent years and are envisioned to further escalate (notably when used in power systems) for several reasons. The SCADA communication networks are increasingly interconnected with corporate information technology (IT) networks for the collection and processing of data in real-time providing greater opportunity for intrusion. Most SCADA devices such as remote terminal units (RTUs), programmable logic controllers (PLCs) and intelligent electronic devices (IEDs) do not incorporate authentication or encryption mechanisms; moreover, remote access to these devices via technologies like Bluetooth and Wi-Fi has been made available increasing the risk of exploit. In addition, typical protocols used for SCADA communications are proprietary meant for dedicated serial communication and hence do not possess basic security services. These protocols have been recently modified for use over Ethernet, in turn increasing access points and risk of compromise.

Hence recently there has been growing interest in SCADA security. Recent attacks involving the Stuxnet worm have also increased activity. The main goal of Stuxnet is to sabotage industrial equipment controlled by a specific Siemens PLC by modifying the PLC code and then hiding the changes using root kits. Such worms can cause severe damage to the underlying physical system. Thus there has been an increasing need to incorporate security combined with intrusion detection and mitigation in smart grid SCADA components.

There are challenges to developing intrusion detection systems (IDSs) specific to SCADA that, in part, stem from the differences between SCADA and traditional IT systems:

- Availability and timeliness are critical; hence the CIA (confidentiality-integrity-availability) design mantra in IT systems transforms to AIC (availability-integrity-confidentiality) for SCADA systems.
- There exists limited memory and computing resources in SCADA field devices such as PLCs, RTUs.
- There is cyber-physical coupling between the SCADA system and underlying power grid. Moreover, attacks on SCADA can have devastating cascading effects on the underlying physical system.

The objective of this paper is to propose a new light-weight intrusion detection approach suitable for implementation in components like RTUs, PLCs with limited computational resources. The approach takes into consideration the aforementioned distinctions. We leverage the properties of a data structure called the Bloom filter, which represents a set of data in concise form suitable for characterizing the set of normal communication traffic patterns in power system SCADA. The normal behavior established using the Bloom filter is used to perform fast $O(1)$ lookups during online intrusion detection. The approach takes into account the physical state of the power system in relation to the SCADA traffic.

Section 2 reviews related work in the field of SCADA-specific intrusion detection; we also explain the MODBUS protocol used for SCADA communications. Section 3 presents the SCADA system architecture considered and our proposed methodology for Bloom filter-based intrusion detection. Section 4 details our implementation while Section 5 highlights insights gained and offers future research directions.

2. RELATED WORK

IDSs can be classified into two main categories: misuse-based and anomaly-based [1]. Misuse-based approaches

2012 25th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)
978-1-4673-1433-6/12/$31.00 ©2012 IEEE

employ known attack patterns to construct signatures that are represented as rule sets using languages such as Snort. Incoming traffic is compared against these rules to determine whether traffic is normal or corresponds to an attack. The improved performance for known attacks comes at the cost of lack of support for novel attacks. In contrast, anomaly-based methods employ system data to create "normal" behavioral profiles during a training phase and then flag deviant profiles during intrusion detection. The ability to detect new attacks trades-off with often-higher false alarm rates. Researchers have recently studied the suitability of these approaches for SCADA communications.

In [2] a misuse-based intrusion detection and event monitoring approach to address unauthorized access to SCADA devices is presented. Details about each SCADA device such as IP address, telnet port, and legal commands are expressed using XML. A Perl program parses this XML and creates Snort signatures that are used to detect attacks offline. The authors of [3] propose a specification-based multi-IDS algorithm based on a formal model of the underlying MODBUS/TCP, which also uses Snort rules to analyze violations in communication patterns. There also exist misuse-based commercial MODBUS filtering modules like Cisco's Net filter [4].

In [5] an anomaly detection method for SCADA systems based on features including network traffic, link utilization and CPU usage is proposed. The feature vector is fed as input to an auto associative kernel regression model that predicts the correct versions of the inputs. Residuals are formed by comparing the observed input values with the model predictions. A binary hypothesis technique called the sequential probability ratio test is applied to the residuals to determine whether the residuals correspond to a normal or abnormal distribution. In [6], a test bed is developed to simulate a SCADA client, server and power grid. The authors propose an autonomic software protection system that has rules to analyze MODBUS requests/responses. This system selectively drops packets to protect SCADA systems from flooding and denial-of -service (DOS) Attacks.

Many IDS systems for SCADA and the MODBUS protocol are centralized and focus on intrusion detection at either the SCADA master or server. In this paper, we propose a distributed approach that extends intrusion detection to field devices including RTUs and PLCs. Moreover, we address detection by leveraging MODBUS payload parameters embedded within the TCP packet. This enables our IDS to take on a more contextual flavor. This is achieved by taking into account the relationship amongst adjacent MODBUS packets (as opposed to considering isolated packets for detection) and incorporating the cyber-physical couplings in the power system by incorporating knowledge of the physical state of the power system.

2.1. MODBUS Protocol

The MODBUS protocol has been the industrial standard for communication among automation and control devices since 1979. Originally designed for serial communication, it has been modified to run over Ethernet residing at the application layer (level 7) of the OSI model. It enables client/server communications between devices connected on different types of networks by employing a master-slave method at the message level but peer-to-peer at the network.

The master (or client) initiates the transactions with the slave devices (or servers) by sending requests; the slaves respond to the requests by performing the requested action and sending a response message. The query from the master using the MODBUS protocol consists of the device address, a function code corresponding to the action requested, any data to be transmitted, and an error–checking field. The slave's response message constructed using MODBUS contains fields confirming the action taken, any data to be returned, and an error–checking field. The function code together with the data is called a protocol data unit (PDU) and is the entity used by our proposed intrusion detection mechanism.

3. PROPOSED APPROACH

The architecture we consider is a typical SCADA master-slave configuration as shown in Figure 1.

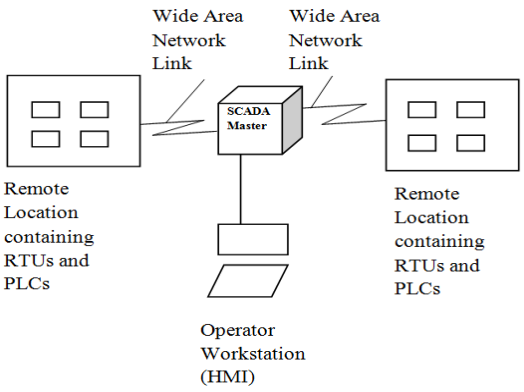

Figure 1. SCADA Master-Slave Configuration.

The main components are the human-machine interface (HMI) at the operator workstation, the SCADA master (also called SCADA server), field devices such as RTUs and PLCs and their associated communication infrastructure. The communications between the SCADA master and field devices are assumed to follow the MODBUS protocol.

3.1. Threat Model

We focus on developing an IDS for external targeted attacks. The reader should note that we do not consider insider attacks nor coordinated distributed attacks. The attack scenarios that fall under this threat model include:

1. HMI compromise, in which the attacker has compromised the HMI and can send commands to SCADA components to destabilize the grid. Common HMI commands in a power systems include read bus line status, read transformer status, read/change magnitude and phase angle of the bus, and open/close circuit breakers. While the attacker can manipulate these commands as per his attack objectives, he can also make sure that the display at the operator interface does not convey the actual state of the system to stakeholders.
2. Man-in-the-middle attack, which involves intercepting communications between the SCADA server and field devices, sending spoofed packets to them. The attacker can employ this to perform illegal writes to coils or registers and even to restart MODBUS servers.

3.2. Methodology

Our proposed approach for detecting such attacks is based on the intuition that the aforementioned attacks manifest as anomalies in the MODBUS request/response traffic patterns. Employing an anomaly-based approach requires identification of "normal" behavior and comparing to it. To establish the normal profile, parameters such as the function code and data extracted from the MODBUS request/response patterns are used. This process can be represented in two steps:

I. Data Extraction, which involves extracting the MODBUS PDU information (function code and data) from the communication traces between the MODBUS server and the field devices using n-gram analysis. This requires extraction of contiguous sequences of 'n' items using a sliding window. The size of the window is dependent on the nature of traffic. For example, say, the MODBUS function codes in the five contiguous requests are 1, 5, 5, 1, and 5 respectively; n-gram analysis to extract the function codes using a sliding window of length five yields 15515.

II. Training and Detection, which is detailed next. The training data is obtained during phases of SCADA operation. The associated MODBUS function codes and data such as coil addresses from this training set are subject to n-gram analysis and then are represented in a compressed form via a Bloom filter.

3.2.1. The Bloom filter

The Bloom filter is a probabilistic memory-efficient alternative data structure to the traditional look up hash. It can be used to compactly represent a set of input values via two components: a set of k independent hash functions h_1, $h_2 \ldots h_k$ with range $\{1, 2 \ldots m\}$ and an m-bit vector **v**.

Let S represent the set of all valid n-grams during training, a representative sample of all n-grams during correct SCADA operation. Consider an n-gram $s=\{s_1, s_2, s_n\}$ of n elements given as input to the Bloom filter. Suppose h_1, $h_2 \ldots h_k$ and **v** are all initialized to zero. To "insert" an element $s \in S$ into the vector **v** during training, it is hashed k times with h_1, $h_2 \ldots h_k$ to produce a string of k values each ranging from 1 to m which we denote as $\{h_1(s), h_2(s) \ldots, h_k(s)\}$. Next the bits in **v** at the indices corresponding $\{h_1(s), h_2(s) \ldots, h_k(s)\}$ are set to 1. A particular bit may be set to 1 multiple times by different inputs; thus an element of **v** set to 1 always stays 1.

To test whether an element b exists in the set S, the elements of **v** at positions $\{h_1(b), h_2(b) \ldots, h_k(b)\}$ are checked to see if they are all marked one; if any one of these k bits is zero, b is assumed not to exist in S. Otherwise, it is possible that $b \in S$ with finite probability.

In our IDS approach, the Bloom filter is used to compactly characterize the sequence of function codes and data obtained during n-gram analysis of the MODBUS training data set. During detection, the n-grams obtained during SCADA communications are passed through the filter; if the outputs, obtained after hashing all point to elements in vector **v** that are marked 1, the tested pattern may be non-anomalous. Otherwise it is not in S.

3.2.2. Bloom filter parameters

There is a trade –off between memory requirements (m) of the Bloom filter and the false positive rate. The false positive rate can be tuned through choice of parameters m and k. According to [8], the probability of a false positive is minimized when the parameters are chosen according to the relation $k= \ln(2) \times (m/n)$ where n is the size of the input data set. The choice of hash functions employed affects the collision rate and hence the false positive rate. A collision occurs when two inputs in S hash onto the same index of vector **v**. The hash function used should produce minimal collisions for this training set. Ideally, the hashes should be independent and uniformly distributed so that the output is equally distributed over the hash space. Unlike some cryptographic hashes such as MD5 and SHA1 the hash functions chosen should be light-weight. After analyzing the suitability of various hash functions shown in the Table 1, the Murmur hash was chosen for our implementation.

Hash function	Pros and Cons
FNV	Pros: Fast Cons: Higher false positives for our data
SAX	Pros: Simple and fast Cons: High collision rate
Jenkins	Pros: Less collision rate Cons: Complicated and suitable for light weight
Murmur	Fast, light weight, produces minimal collisions for our data

Table 1 Comparison of hash functions.

3.2.3. Advantages

There are several advantages to the Bloom filter. Lookups and insertions are of order O(1). The data structure can be dynamically updated easily as more data becomes available as adding an element involves hashing and setting an element in an array. False positive rates can be adjusted by the choice of hash function used and other Bloom filter parameters. The Bloom filter is space-efficient compared to other data structures like arrays, linked lists and hash tables and is used in many network applications.

3.2.4. Anomaly-based detection via the Bloom filter

The Bloom filter is used during an initial offline training phase and when in online detection mode for anomaly-based detection. During training, the function code and data parameters are extracted from the offline data and stored in two different Bloom filters as shown in the Figure 2.

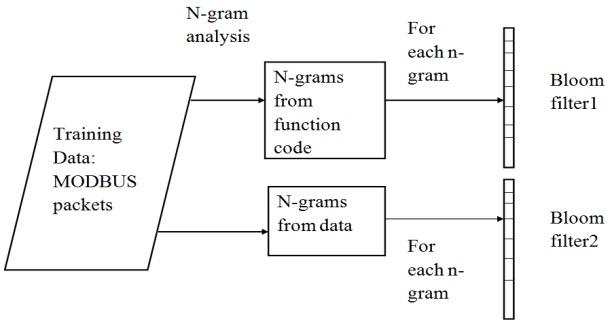

Figure 2 Training phase. The n-grams for function codes and associated data are separated and fed into two different Bloom filters. The output of each filter is a binary vector v.

During detection, the same parameters are extracted from the live MODBUS request/response patterns and then hashed. Lookup is performed against the Bloom filters constructed during training as shown in the Figure 3. Each bloom filter assigns a score to the pattern tested indicating the probability of anomalous behavior and the weighted sum from various Bloom filters is the final anomaly score.

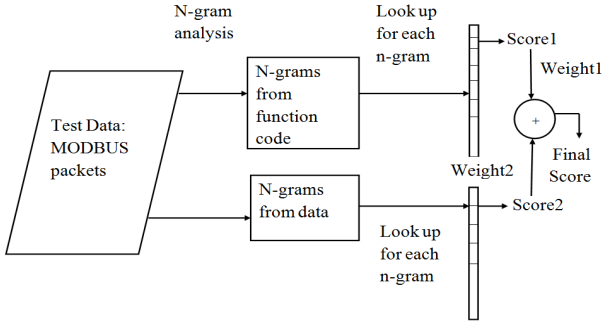

Figure 3 Detection phase. The hash outputs corresponding to the live traffic are compared to the Bloom filter vectors established during training.

In our current approach, the scores from both the filters are equally weighted at 0.5. However these weights can be varied when more parameters are added.

3.2.4. Including Physical Information

Our approach takes into account the communication patterns within the SCADA network. However as these patterns are strongly correlated with the physical information in the power system, we assert that an approach to incorporate the physical state would improve robustness.

The operating states of a power system shown in the Figure 4. In the normal state, existing generators can serve the total load without violation of operational constraints. Examples of some operational constraints include limits on transmission line flows and voltage magnitudes.

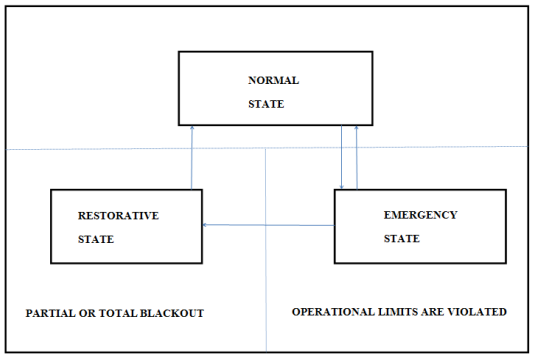

Figure 4 State Diagram for Power System Operation.

In the emergency operating state, though some operational constraints are violated, the power system still supplies power to all the loads. In this state, the system operator takes corrective measures to bring the system back to the normal state. The corrective measures can include disconnecting loads, lines, and opening and closing of breakers. If these measures rectify the operating limit violations and stabilize the system with a reconfigured topology, load versus generation balance has to be restored for the system to supply all loads once again. This state is known as the restorative state. Hence the set of SCADA actions performed by the system operator vary according to the operating states of a power system, which in turn affects the way in which normal and abnormal behavior is defined for the MODBUS under different physical states.

For example, the function code 0x01 in MODBUS corresponds to the "Read Coils" operation. This command also restarts the device and runs start-up tests. Seeing this function code frequently within a time window may be abnormal under normal system state. However, represents a valid operation under emergency state. If the intrusion detection system does not take this physical state into account while assigning an anomaly score to the tested communication pattern, normal traffic would be assigned a higher anomaly score. This results in higher false positives.

This necessitates the inclusion of power system state information for intrusion detection.

The question naturally arises as to how easily an attacker can compromise the state information in IDS. The physical state is typically estimated using a process called state estimation in SCADA. According to [9], if a power system has l measurement devices and r state variables, an adversary needs to compromise at least l-r+1 measurements to ensure that he can falsify state estimation. However l-r+1 is usually large for practical power systems. Even an IEEE 300-bus test system has l-r+1 =824 devices that must be compromised thus creating an impractically high degree of difficulty for the attacker especially in light of our threat model.

4. EXPERIMENTAL RESULTS

The software used for our implementation is Wire shark, a network protocol analyzer and C++. The data obtained from sources [7] and at www.pcapr.net were analyzed using Wire shark. The n-gram analysis, creation of Bloom filter during training and anomaly detection were implemented using C++. The data for training is based on the MODBUS register and command assignments in [7]. Preliminary data is taken from [7] and some similar patterns are created to be used as the training data. The MODBUS function codes and the data like coil addresses from this training set are subject to n-gram analysis.

In our simulations after performing n-gram analysis, parameters k and n of the Bloom filter were varied and the results obtained are shown in Tables 2 and 3. The reader should note that it is well known that false negatives are not possible in a Bloom filter whereas the false positive rate can be adjusted via parameter tuning. However, Tables 2 and 3 show the reverse due to a shift in interpretation. The false negative of the bloom filter corresponds to false positive of the overall IDS. This can be explained as a false negative in the Bloom filter is equivalent to saying "a particular sequence is not valid when it is actually valid". This is not possible because if the sequence is valid and is included in the training data set, it would be represented in a compressed form in the bloom filter. Hence any normal pattern represented by the filter during training will not be classified as anomalous during detection. In other words, the system cannot have false positives with proper training data set. The results are in accordance with this fact.

In a similar manner, the false positive rate of the bloom filter corresponds to the false negative rate of the overall IDS. We can also infer that there is a decrease in the false negative rate when k is increased from 2 to 4 after which there is no improvement. Hence it is sufficient to have 4 independent hash functions to achieve nominal detection for the chosen data. The parameter 'm' was calculated using the formula

k= (ln 2) *(m/n). For example, when k=4, m= (n*k)/ (ln 2) = (100*2)/ (ln 2) = 580 which is the size of the bloom filter.

Value of k	**2**	**4**	**6**
m (size of the Bloom filter)	289	580	870
No. of bits set in **v** after training	202	404	606
False positive rate (normal pattern classified as anomalous)	0	0	0
False negative rate (anomalous is classified as normal)	6%	2%	2%

Table 2: Performance by varying k.

From Table 3, it can be observed that there is a decrease in the false negative rate when n is increased from 100 to 200. But when it is increased beyond 200 there is no improvement in false negative rate.

Value of n	**100**	**200**	**500**
m (size of the Bloom filter)	580	1159	2899
No. of bits set in **v** after training	402	799	2000
False positive rate (normal pattern classified as anomalous)	0	0	0
False negative rate (anomalous is classified as normal)	2.4%	0%	0%

Table 3: Performance by varying n.

Hence a training data set of size n=200 with k=4 different hash functions were chosen and the intrusion detection algorithm was tested. A representative output of the IDS is shown in the trace below:

Testing function sequence: *53523*
The tested function sequence may exist in the training set
cumulative Anomaly Score=0

Testing function sequence: *55555* Data sequence: 0000000000
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5

Testing function sequence: *55555* Data sequence: 00FF00FF00
The tested function sequence does not exist in the training set
cumulative Anomaly Score=1.0

Testing function sequence: 15151
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5

It can be seen that the anomaly score can be different for the same sequence of function calls (55555) depending upon the data. This is because a function code of "5" corresponds to read coils and the pattern of alternating 00 and FF correspond to alternating coil 'ON' and 'OFF' which is more suspicious. Hence this pattern gets a higher anomaly score. The function code sequence 15151 corresponds to "Read Coil- Write Coil- Read Coil-Write Coil-Read Coil". As the command "read coil" restarts the device and runs some start-up tests, occurrence of this pattern quite frequently within a time window is classified as anomalous. The same pattern is not anomalous under "emergency" conditions of the power system as can be seen in the output trace below:

Testing function sequence: *53523*
The tested function sequence may exist in the training set
cumulative Anomaly Score=0

Testing function sequence: *55555* Data sequence: 0000000000
The tested function sequence does not exist in the training set
cumulative Anomaly Score=0.5

Testing function sequence: 15151
The tested function sequence may exist in the training set
cumulative Anomaly Score=0

When training was done using normal data, the sequence 15151 was given an anomaly score of 0.5 and when trained using emergency data, the same sequence was given an anomaly score of 0. Hence without the knowledge of physical state of the system, it is not possible to ascertain the correct anomaly score. When information regarding the physical state of the power system is available, the results change as shown below:

If the current state of the power system is "normal", then the results become

Testing function sequence: *15151*
The tested function sequence does not exist in the training set
state=normal
cumulative Anomaly Score=1

If the current state of the power system is "emergency", then the results become

Testing function sequence: *15151*
The tested function sequence does not exist in the training set
state=emergency
cumulative Anomaly Score=0

When the knowledge of the physical state is available, the anomaly score can be picked according to the state information as shown in the output. Hence knowledge of physical state of the power system improves the accuracy of the intrusion detection system.

During execution of this algorithm, input data sets of size 100, 1000 or greater than 1000 had an execution time of time in the order of a few milliseconds. This demonstrates that the time complexity of the algorithm is almost constant for various training input sizes demonstrating an advantage of Bloom filter approach.

5. CONCLUSION AND FUTURE WORK

A Bloom filter-based intrusion detection approach for resource constrained SCADA field devices is proposed and tested on SCADA traffic using the MODBUS protocol. The novel components of the system include the use of Bloom filter-based n-gram analysis as well as the physical state of the information. We have observed that the Bloom filter has the following advantages over other data structures for incorporation in a SCADA distributed IDS:

- low memory requirements;
- zero IDS false positive rates ;
- anonymity as the Bloom filter does not store information direction unlike other data structures.

The proposed approach has been tested with and without the knowledge of the power system state. We can infer from our results that knowledge of the physical state helps to reduce false positives. Moreover, the approach is distributed across field devices such as PLCs and RTUs, hence the IDS can be partially operational even if few field devices are compromised.

Future work involves identifying effective measures to aggregate individual alerts raised by local device IDSs. In addition, testing the approach for greater attack scenarios via implementation in a test bed with power system components will be considered as part of our future work.

6. REFERENCES

[1] P. Ning and S. Jajodia, "Intrusion Detection Techniques," in *The Internet Encyclopedia*, H. Bidgoli, ed., John Wiley & Sons, December 2003.

[2] B. Zhu, and S. Sastry, "SCADA-specific Intrusion Detection/Prevention Systems: A Survey and Taxonomy," in *Proceedings of the 1st Workshop on Secure Control Systems (SCS), CPSWeek 2010*, Stockholm, Sweden, April 2010.

[3] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner and A. Valdes, "Using Model-based Intrusion Detection for SCADA Networks," in *Proceedings of the SCADA Security Scientific Symposium*, Miami Beach, FL, January 2007.

[4] V. Pothamsetty and M. Franz, Transparent Modbus/TCP Filtering with Linux. Available online: http://modbusfw.sourceforge.net/. Last access 12/30/2011.

[5] D. Yang, A. Usynin and J.W. Hines, "Anomaly-Based Intrusion Detection for SCADA Systems," in *Proceedings of the 5th International Topical Meeting on Nuclear Plant Instrumentation, Control and Human Machine Interface Technologies (MPIC&HMIT 05)*, Albuquerque, NM November 2006.

[6] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga and S. Hariri, "A Testbed for Analyzing Security of SCADA Control Systems (TASSCS)," in *Proceedings of the 2011 IEEE PES Innovative Smart Grid Technologies (ISGT)*, Anaheim, CA, January 2011.

[7] S. Cohen, W. Sherman, and M. Sherman, "MODBUS/TCP Controller for the Power Supplies in ALS BTS Beam Line," in *Proceedings of the IEEE Particle Accelerator Conference (PAC)*, Albuquerque, NM, June 2007.

[8] Y. Al-Nashif, A.A. Kumar, S. Hariri, Y. Luo, F. Szidarovsky and G. Qu, "Multi-Level Intrusion Detection System (ML-IDS)," in *Proceedings of the International Conference on Autonomic Computing (ICAC)*, Chicago, IL, June 2008.

[9] R.B. Bobba, K.M. Rogers, Q. Wang, H. Khurana, K. Nahrstedt and T.J. Overbye, "Detecting False Data Injection Attacks on DC State Estimation," in *Proceedings of the First Workshop on Secure Control Systems (SCS), CPSWeek 2010*, Stockholm Sweden, April, 2010.