2013 12th International Conference on Machine Learning and Applications

An Evaluation of Machine Learning Methods to Detect Malicious SCADA Communications

Justin M. Beaver, Raymond C. Borges-Hink, Mark. A. Buckner
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
{beaverjm, borgesrc, bucknerma}@ornl.gov

***Abstract*—Critical infrastructure Supervisory Control and Data Acquisition (SCADA) systems have been designed to operate on closed, proprietary networks where a malicious insider posed the greatest threat potential. The centralization of control and the movement towards open systems and standards has improved the efficiency of industrial control, but has also exposed legacy SCADA systems to security threats that they were not designed to mitigate. This work explores the viability of machine learning methods in detecting the new threat scenarios of command and data injection. Similar to network intrusion detection systems in the cyber security domain, the command and control communications in a critical infrastructure setting are monitored, and vetted against examples of benign and malicious command traffic, in order to identify potential attack events. Multiple learning methods are evaluated using a dataset of Remote Terminal Unit communications, which included both normal operations and instances of command and data injection attack scenarios.**

Keywords—SCADA; machine learning; intrusion detection; critical infrastructure protection; network

I. Introduction

SCADA systems used for the command and control of critical infrastructure have primarily been implemented independent of cyber security considerations. These systems, also called Critical Infrastructure Systems (CIS), have security with respect to trusted communications in transactions at the control system layer, but lack a focus on traditional cyber security implementations such as authentication and intrusion detection at the network layer. Historically, CISs were insulated from many security vulnerabilities due to their proprietary implementations and in many cases physical isolation from the Internet, and so the independence of these two different security missions reflects these different levels of vulnerability.

However, modern CISs are moving towards more open systems and standards, improving the compatibility of these systems with each other and with commercially available management software [3]. While this evolution increases the efficiency of operating multiple physical CIS locations, the lack of designed-in security at the SCADA level has exposed the potential for vulnerabilities that bridge both Computer Network Defense (CND) and CIS applications. The prolific nature of remotely controlled substations and wireless networks has created the potential for malicious and intentional deception. Adversaries can leverage the wireless networks to obtain unauthorized access and then manipulate CIS SCADA communications in order to adversely affect the delivery of resources to a population.

Information security for CISs must account for these new threat scenarios and provide effective critical infrastructure protection, despite the lack of inherent security mechanisms in commercially available SCADA software. Since mission resilience, or maintaining the delivery of a mission-critical service or resource, is the primary quality indicator of a CIS, incorporating a security layer a posteriori presents a significant challenge. Any implemented security must have a high reliability so that the critical system does not interrupt service based on an incorrect security layer assessment. Also, to accommodate the uniqueness of CIS hardware and software configurations, the security layer must be adaptive and provide the same quality regardless of a system's equipment or technologies.

In this work, the viability of machine learning applied to CIS communications is explored as an approach that can provide a reliable yet adaptive security layer. We focus specifically on the problem of intrusion detection in a CIS; where the control system transactions are monitored in real-time to detect, independent of the SCADA system, those transactions that have been manipulated to deceive operators or automated controls. The prevalence of wireless networks as a conduit for CIS communication makes this attack scenario very plausible for the interruption of a critical service or to damage mission-critical equipment. So, intrusion detection within a CIS, analogous to traffic monitoring in an enterprise network, is a necessary element in the broadened scope of security.

We evaluate a set of machine learning algorithms in terms of their ability to identify various attacks when analyzing remote terminal unit (RTU) serial communications in a gas pipeline system. The RTU data used in this set of experiments was developed by the Mississippi State University's Critical Infrastructure Protection Center [7], and includes examples of benign RTU transactions and variants of command and data injection attack transactions generated specifically for critical infrastructure protection research. We analyze the accuracy of each machine learning algorithm at correctly identifying malicious traffic using a set of features (key-value pairs) that are derived from the RTU telemetry.

978-0-7695-5144-9 2013
U.S. Government Work Not Protected by U.S. Copyright
DOI 10.1109/ICMLA.2013.105

II. Related Work

This work is focused on an evaluation of machine learning methods as discriminators for malicious SCADA communications. This is an extension to previous efforts in applied machine learning for malicious network traffic detection. In the cyber security domain, prior art exists where machine learning methods have been used to discriminate network traffic. In many of these, as in [9], machine learning is used to classify traffic for the purpose of discovering the type of service being used. Most other work is focused on discriminating malicious network traffic from non-malicious traffic. A review of these approaches is described in [5], including both supervised and unsupervised methods such as support vector machines, naive Bayes, clustering, decision trees and random forests. In addition, recent work has focused on more complex approaches to malicious network flow detection by applying semi-supervised machine learning models [2], and in formal experimentation of scaled machine learning intrusion detection system (IDS) prototypes [1].

There are several advantages to using machine learning to discriminate malicious network communications [11], including minimizing reliance on human analysis and adaptation to local network environments. Machine learning provides an insight that would be difficult for a human to explicitly describe as a rule or signature because it can potentially evaluate thousands of interdependent metrics simultaneously. Unlike signature-based systems, zero-day (previously unseen) attack detection is possible because communications are classified based on their *similarity* to known types rather than the explicit patterns specified in signatures. These systems are also particularly effective in detecting variants of attacks [12], which are small changes in a known attack vector created to bypass signature-based sensors.

In the CIS domain, there has been some prior work in the analysis of SCADA communications for intrusion detection. Signature-based intrusion detection approaches have been implemented on Modbus networks as described in [13]. Machine learning methods have also been applied previously to critical infrastructure, albeit not with an intrusion detection focus. Fukuda and Shibata [6] describe an application of neural networks to supervised control where the neural network learns the associations between control measurements and user actions. Fuzzy learning [10] has been employed to improve the performance of machine drives. More recently, Won et al. [8] used decision trees to perform fault and failure diagnosis in industrial control networks. These prior works seek to optimize the efficiency of the control system, and the use of machine learning as a tool for control system security does not appear to have been explored.

The benefits and qualities of machine learners applied to network traffic are also applicable in SCADA systems, in order to give a CIS system a defense capability in recognizing when system operations are being manipulated. We believe this work to be original in terms of using machine learning in a CIS context for the purpose of control system security.

III. Evaluation Approach

The focus of the evaluation is to quantify the ability of a set of machine learning methods to detect malicious (attack) transactions within a CIS data stream. We evaluate 6 of the most widely used supervised learning algorithms as described in [16]. Each method is applied in isolation to the RTU communication data set and then compared with competing methods, using 10-fold cross validation for the training/test data balance. The features selected for classification were extracted from the RTU telemetry. In this section, we describe our entire evaluation approach, including the machine learning algorithms applied, the data used in the experiment, the collection of features used in discriminating traffic types, the nature of the malicious traffic, and the overall approach to experimentation.

A. Machine Learning Methods

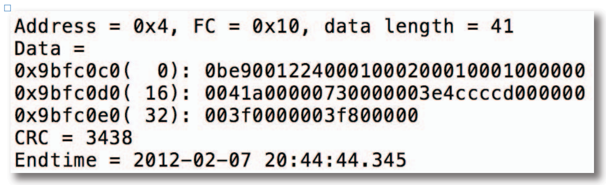
The methods selected are a mix of simple learners, industry-standard learners and traditionally good performers in terms of generalization. We intend for this work to establish a foundation for the application of machine learning to CIS, with the expectation that more complex algorithms will be necessary as the complexity of the CIS grows. The evaluations were performed using the Weka machine learning software [15]. The methods used were:

1. **Naïve Bayes** - is a probabilistic classifier based on Bayes' theorem [20], and adopted into the field of machine learning in 1992 [21].
2. **Random Forests** – consisting of a combination of tree predictors where each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest [17].
3. **OneR** - simply evaluates each feature's optimum ruleset and chooses the best one [18].
4. **J48** - is an implementation of the C4.5 decision tree algorithm [19].
5. **NNge** – Nearest-neighbor-like algorithm using non-nested generalized exemplars [22].
6. **SVM** – Support vector machines [4] trained using sequential minimal optimization [23].

B. Experiment Data

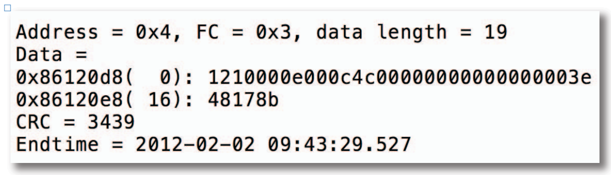
The data used for this experiment is a collection of labeled RTU telemetry streams from a gas pipeline system in Mississippi State University's Critical Infrastructure Protection Center [14]. The telemetry streams included examples of command injection attacks, data injection attacks, and those where no attacks occurred (normal). From these streams, we generated feature sets used as the basis for discrimination by the learning methods. The process for feature set generation was to ingest the RTU data frame-by-frame, break the frames into individual values of telemetry data, translate data items (such as floating point values) into variables that are programmatically represented, and grouping variables in collections based on timing. Once the feature sets were developed from the RTU telemetry, we evaluated various learners and their generalization performance on those labeled data sets.

Each telemetry stream was comprised of both commands and responses. A command is used to set the value of an item under test. In the case of the gas pipeline system, a programmable logic controller (PLC) is used to maintain a specific pipeline pressure value. The RTU commands communicate the desired gas pipeline pressure, called a setpoint value, to the PLC that contains the logic and physical relays to control the pipeline to that pressure value. In some cases, the RTU command is simply a request for the latest pipeline pressure value. A response is the PLC reporting the current value of the gas pipeline pressure. Nominally, commands and responses will occur in pairs, where the SCADA system commands the PLC to a specific setpoint or requests the pressure value, and the PLC reports the current value. Examples of an RTU command and response from the raw data are shown in Figure 1 and Figure 2, respectively.



```
Address = 0x4, FC = 0x10, data length = 41
Data =
0x9bfc0c0(  0): 0be90012240001000200010001000000
0x9bfc0d0( 16): 0041a00000730000003e4ccccd000000
0x9bfc0e0( 32): 003f0000003f800000
CRC = 3438
Endtime = 2012-02-07 20:44:44.345
```

Figure 1 RTU Command Example



```
Address = 0x4, FC = 0x3, data length = 19
Data =
0x86120d8(  0): 1210000e000c4c00000000000000003e
0x86120e8( 16): 48178b
CRC = 3439
Endtime = 2012-02-02 09:43:29.527
```

Figure 2 RTU Response Example

Each telemetry item has a PLC address and function code (FC) that defines the type of command or response it is. Raw floating-point values for the setpoint and pipeline pressure are embedded in the packet body in addition to bytes with flag values that define the state of the gas pipeline system in each command and response. Timestamps and Cyclic Redundancy Checks (CRCs) are additional components to the commands and responses.

C. *Feature Set Design*

Machine learning systems base both their learning and analysis on a data structure called a *feature set*, which is a collection of key-value pairs derived from the raw data that represent indicative elements of the data. Examples of different classes of data are stored as feature sets, and acquired raw data is converted into feature sets prior to classification. The feature sets in this application focus on the specific values associated with the RTU data and the results of simple tests that provide checks on the integrity of both the data and protocol of the transactions. The complete set of features used in this machine learning evaluation is described in Table 1.

Table 1 RTU Feature Descriptions

Feature Name	Description
Pipeline Pressure (PSI)	The gas pipeline pressure value, pounds per square inch (PSI), in the transaction response.
Invalid Function Code	A binary indicator of whether the function code is invalid.
Setpoint	The setpoint value included in the command.
Invalid Data Length	The transaction is comprised of a command or response with a data element of an invalid size.
Command	A binary indicator of whether the transaction contains command data.
Command Data Length	The length of the command data for this transaction.
Response	A binary indicator of whether the transaction contains response data.
Response Data Length	The length of the command data for this transaction.
Control Mode	An indicator of whether the RTU unit is in auto control mode or off.
Control Scheme	An indicator of whether control is accomplished through the solenoid.
Solenoid State	A binary indicator of whether the solenoid is on or off.
Pump State	A binary indicator of whether the pump is on or off.

D. *Attack Data Description*

The data/response injection attacks in this experiment focus on manipulating PLC responses to deceive a human operator or the automated control as to the actual state of the gas pipeline system. There were 7 variants of the data injection attack replicated in the RTU telemetry, each manipulating the PLC response in a different way.

1. Negative Values - Injecting negative values as the pipeline pressure, which are invalid as the pressure should always be a positive or zero value.
2. Burst Values - Sending multiple successive pipeline pressure values, faster than the data display rate for the operator interface.
3. Fast Change - Sending successive and variant pipeline pressure values to create a lack of confidence in the correct operation of the system.
4. Single Data Injection - Following an actual response with an artificial one where the gas pipeline value is doctored in order to deceive the PLC control loop.

5. Slow Change - Sending delayed and variant pipeline pressure values to create a lack of confidence in the correct operation of the system.

6. Value Wave Injection - Multiple oscillating pressure values in order to deceive the PLC control loop and undermine the operator's confidence in the system.

7. Setpoint Value Injection – The attacker sends false pipeline pressure values equal to the setpoint.

Command injection attacks manipulate outgoing commands to control the gas pipeline, or to acquire information about the pipeline system and PLC. The four types of command injection attacks are:

1. Address Scan – The attacker scans and map the whole system by sending packets with a range of addresses.

2. Function Scan – A scan to determine the range of possible function codes.

3. Illegal Setpoint – The attacker modifies the PSI value and setpoint to be too high.

4. Illegal PID Command – The PLC control loop parameters are changed to modify its performance.

The RTU telemetry data included examples for all seven and all four of the command/data injection attack types and examples of normal transactions. The distribution of these training data instances is shown in Table 2.

Table 2 Distribution of Training Data Instances

Command Injection		**Response Injection**	
Dataset	**Instances**	**Dataset**	**Instances**
Normal	28086	Normal	10000
Address Scan	2	Burst	217
Func Code Scan	9	Fast	225
Illegal Setpoint	197	Negative	119
Illegal PID	49	Setpoint	171
		Single	33
		Slow	306
		Wave	253

Of immediate note is the potential for class imbalance as there are a disproportionate number of normal instances compared to attack instances for each injection type. We calculate precision and recall for each class in order to insure that the relatively high number of normal RTU transaction instances does not skew our overall performance results.

E. Experiment Design

The aim of this work is to determine the feasibility of machine learning as a discriminator of malicious RTU transactions. Feasibility can be quantified in terms of a high degree of precision and recall. That is, an ability to correctly identify both malicious and normal transactions with minimized misclassifications. We performed a 10-fold cross-validation on each learner, and since the data is comprised of variants of each attack type, we evaluated the learners as both binary and multiclass classifiers.

IV. Results

This section contains the evaluation results for applying machine learning methods to SCADA system commands and responses. For each type of command/data injection attack we describe the base performance of each learner when using the features extracted from the RTU telemetry. We present the precision and recall for each class and each method, for both binary and multiclass classification problems.

A. Data/Response Injection Results

Figure 3 and Figure 4 show the recall and precision for each of the learning methods in discriminating the data injection attack types as well as normal RTU transactions.

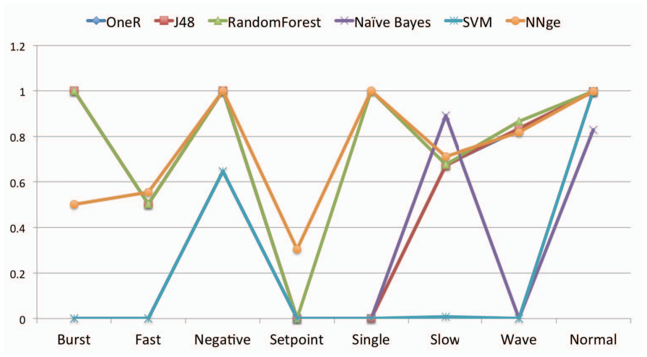


Figure 3 Data Injection Multiclass Classification Recall

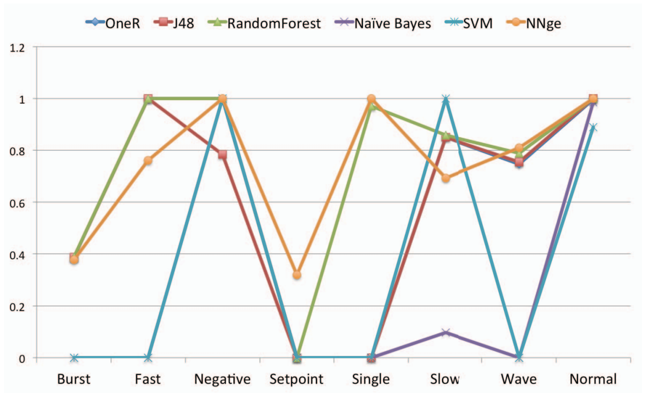


Figure 4 Data Injection Multiclass Classification Precision

Overall, the nearest neighbor (NNge) and random forest algorithms performed the strongest across all classes with perfect classification of the normal responses and recall/precision values of 0.75 or higher for five of the seven response injection classes. All learners had degraded performance in classifying the setpoint attack, which is not surprising since the reported pressure value in that attack is consistent with an expected setpoint value. Being able to discriminate this attack at all is a testament to the value of machine learning as considering complex combinations of features.

Figure 5 and Figure 6 show the results of binary data injection classification. In this approach, all data/response injection attack instances are combined into a single 'malicious' labeled class to be discriminated from the normal communications. The binary classification results are very compelling with several learning algorithms, including the simplistic One-R method, approaching perfect classification. Reducing the problem to a binary classification problem appears to leverage the generalization performance of machine learning more effectively, albeit at the cost of fidelity in understanding the specific type of levied attack.

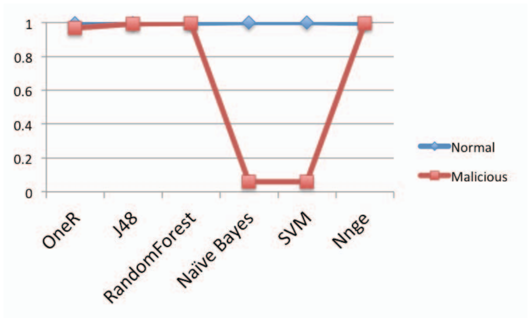


Figure 5 Data Injection Binary Classification Recall

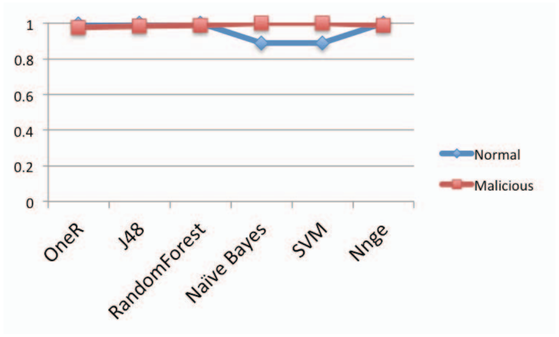


Figure 6 Data Injection Binary Classification Precision

B. *Command Injection Results*

Figure 7 and Figure 8 show the recall and precision for the each of the learning methods in discriminating the command injection attack types as well as normal instances. The multiclass results for command injection are stronger and more consistent overall than those of data/response injection.

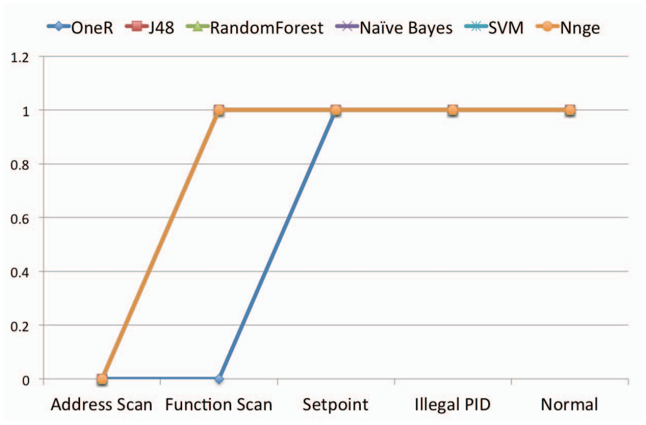


Figure 7 Command Injection Multiclass Classification Recall

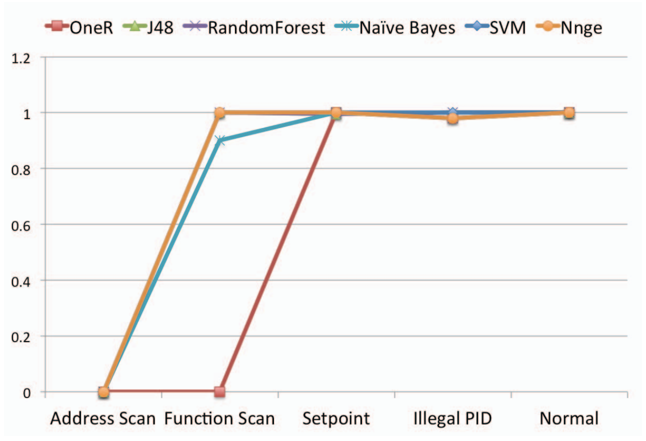


Figure 8 Command Injection Multiclass Classification Precision

For the normal RTU transactions, all learners demonstrated perfect classification, but most also did well for three of the four command injection attacks. We attribute this performance improvement in attack detection to the relative simplicity of the command injection attacks as compared to data/response injection. The address scan was not classified well, but we attribute this to the minimal amount of exemplar data for that attack type.

Similar to the data/response injection analysis, the learners performed more strongly when the training data was organized as a binary classification problem, as evidenced by the precision/recall results shown in Figure 9 and Figure 10. As with data/response injection, combining the malicious RTU instances more fully leverages the generalization power of the learners and produces a stronger classification.

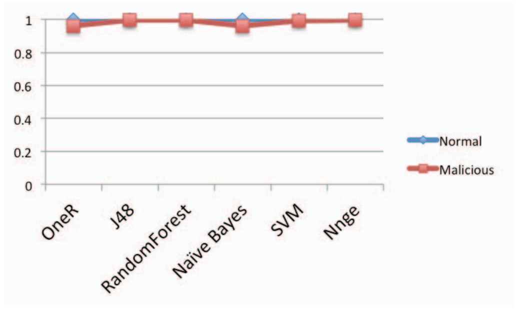


Figure 9 Command Injection Binary Classification Recall

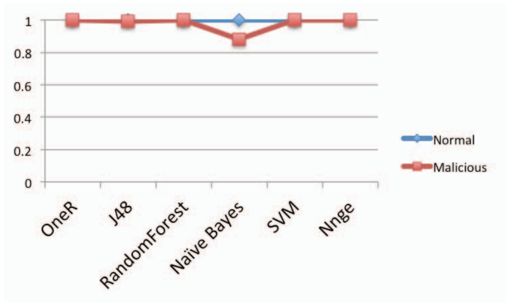


Figure 10 Command Injection Binary Classification Precision

V. Conclusion

The application of machine learning methods to SCADA data demonstrates their promise in addressing security concerns in the CIS domain. With a very basic set of features, and treating the detection of malicious RTU communications as a binary classification problem, this experiment was able to demonstrate the power of machine learning at detecting broad classes of attacks. The six selected learning methods generalized the RTU data well, and the precision/recall values for the binary classifiers were high enough to be operationally feasible as an a posteriori instrusion detection approach for CISs.

Despite the good performance of the various learners, this work identified several opportunities for improvement that center on considering additional features. As many of the injection attacks involve timing, format, and protocol violations, we intend to extend this work to explore features that consider these elements of the telemetry stream in making the intrusion detection decision. Improved performance will be necessary in order to maintain the same machine learning generalization performance with a more minimal training data set and in a more complex CIS comprised of multiple PLCs and multiple SCADA systems.

We applied multiple learning algorithms to RTU data in order to show their viability as an intrusion detection approach for CISs. We recognize that the learning methods employed in this work can be considered state-of-the-practice. The novelty is in the application of machine learning to solve the problem of the post-deployment application of security in the CIS domain. We see this work as a foundation, and encourage the future exploration of more complex SCADA systems, more difficult attack vectors, and more advanced machine learning methods to discriminate those attacks.

Acknowledgment

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, P.O. Box 2008, Oak Ridge, Tennessee 37831-6285; managed by UT Battelle, LLC, for the U.S. Department of Energy under contract DE-AC05-00OR2225. This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 for the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains non-exclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

We would like to thank Dr. Thomas Morris and his staff at the Mississippi State University's Critical Infrastructure Protection Center for providing the critical infrastructure data that made this study possible.

References

[1] J.M. Beaver, C.T. Symons, and R.E. Gillen, "A learning system for discriminating variants of malicious network traffic," Proc. 8th Annual Cyber Security and Information Intelligence Workshop, January 2013.

[2] C.T. Symons and J.M. Beaver, "Nonparametric semi-supervised learning for network intrusion detection: combining performance improvements with realistic in-situ training," Proc. 5th ACM Workshop on Security and Artificial Intelligence, pp. 49-58, October 2012.

[3] A. Cardenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems," Proc. 3rd Conf. on Hot Topics in Security (HOTSEC '08).

[4] C. Cortes and V. Vapnik, "Support-vector networks," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.

[5] S. Dua and X. Du, Data Mining and Machine Learning in Cybersecurity, Boca Raton, FL, Taylor and Francis Group, LLC, 2011.

[6] T. Fukuda and T. Shibata, "Theory and applications of neural networks for industrial control systems.," IEEE Trans. Industrial Electronics, vol. 39, no. 6, pp. 472–489, 1992

[7] Mississippi State University, Critical Infrastructure Protection Center, http://www.security.cse.msstate.edu/cipc/.

[8] Y. Won, M. Choi, B. Park, and J.W. Hong, "An approach for failure recognition in IP-based industrial control networks and systems," International Journal of Network Management, 2012.

[9] S. Zander, T.T.T. Nguyen, et al, "Automated traffic classification and application identification using machine learning," IEEE Conference on Local Computer Networks 30th Anniversary (LCN '05), Sydney, Australia, 2005.

[10] L. Zhen and L. Xu, "Fuzzy learning enhanced speed control of an indirect field-oriented induction machine drive," IEEE Transactions on Control Systems Technology, vol. 8, no. 2, pp. 270-278, 2000.

[11] R. Sommer and V. Paxson, "Outside the closed world: on using machine learning for network intrusion detection," 2010 IEEE Symposium on Security and Privacy, 2010.

[12] O. Sharma, M. Girolami, et al., "Detecting worm variants using machine learning," ACM Conference on emerging Network EXperiments and Technologies(CoNEXT), New York, NY, 2007.

[13] T. Morris, R. Vaughn, and Y. Dandass, "A Retrofit Network Intrusion Detection System for MODBUS RTU and ASCII Industrial Control Systems," 45th Hawaii Intl. Conf. on System Sciences (HICSS), 2012.

[14] T. Morris, R. Vaughn, and Y.S. Dandass, "A testbed for SCADA control system cybersecurity research and pedagogy," Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research, 2011.

[15] M. Hall, E. Frank, et al., "The WEKA Data Mining Software: An Update," SIGKDD Explorations, vol. 11, no. 1, 2009.

[16] R. Caruana and A. Niculescu-Mizil, "An empirical comparison of supervised learning algorithms," Proceedings of the 23rd Intl. Conf. on Machine Learning, pp. 161-168, 2006.

[17] L. Breiman, "Random forests," Machine Learning vol. 45, no. 1, pp. 5-32, 2001.

[18] R.C. Holte, "Very simple classification rules perform well on most commonly used datasets," Machine Learning, vol. 11, no. 1, pp. 63-90, 1993.

[19] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[20] T. Bayes. Phil. Trans. of the Royal Soc. of London, 1763.

[21] P. Langley, W. Iba, and K. Thompson, "An analysis of Bayesian classifiers," AAAI, vol. 90, 1992.

[22] B. Martin, Instance-Based Learning: Nearest Neighbor with Generalization, University of Waikato, 1995.

[23] J. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," Advances in Kernel Methods – Support Vector Learning, 1998.