Machine Learning Models for Cyberattack Detection in Industrial Control Systems

David Arnold, John Ford, and Jafar Saniie

Embedded Computing and Signal Processing (ECASP) Research Laboratory (http://ecasp.ece.iit.edu)
Department of Electrical and Computer Engineering
Illinois Institute of Technology, Chicago IL, U.S.A.

Abstract – Industrial Control Systems (ICS) provide a network environment for operator command and control of cyber-physical systems and devices. As such, these systems are common throughout power plants, pipelines, manufactories, and other critical infrastructure. Due to the importance of critical infrastructure in everyday life, hackers of all stripes are taking an increased interest in compromising ICS for personal gain or nefarious purposes. Recent high-profile attacks also highlight the need to monitor system process and sensor data in detecting compromised user accounts or insider threats. Central to ICS data collection and monitoring, Data Historians hold the key to identifying malicious behavior and cyber breaches. Machine Learning techniques may be applied to detect data that is indicative of part failures or cyberattacks, allowing operators to take preventative measures prior to system failure. In this paper, several machine models for cyberattack detection within Industrial Control Systems will be introduced. These models will reside within the Data Historian and are implemented through the Apache Spark MLlib Libraries. The machine learning model were applied to an ICS Dataset to determine the system's accuracy at detecting cyberattacks. Naïve Bayes, Logistic Regression, Decision Tree Classifier, and Random Forest Classifier Machine Learning models were tested with the Tree Classifiers producing the most promising results.

I. Introduction

As the primary data storage unit for Industrial Control Systems (ICS), the Data Historian (DH) is responsible for logging events and system activity. These duties place the DH in a prime position to provide supplemental cyberattack detection by analyzing sensor data and network traffic. Detection capabilities can be implemented through machine learning models analyzing reported sensor data and network traffic. To handle the increased workload and increase system reliability, a distributed architecture increases data redundancies while also leveraging a larger computing cluster to host the machine learning models.

Modernization efforts for Industrial Control Systems have drastically increased the safety and reliability of plant operation, enabling the integration of advanced predictive models and remote operation. However, the shift from analog to digital systems increased the network footprint of ICS while also attracting attention from various global hacking organizations. Ranging from common criminals to nation-state actors, these hackers have targeted ICS networks for their importance to national infrastructure. Originally, ICS cyber-defense doctrine relied heavily on the use of proprietary software and network isolation from the wider Internet. High-profile cyberattacks highlight the deficiencies of traditional Industrial Control System security. During the Stuxnet attack, hackers deployed malware to manipulate the runtime code of centrifuge Programmable Logic Controllers (PLC), causing the centrifuges to operate at a range where they would become damaged and inoperable [1-3]. The attack circumvented the network isolation through the use of infected flash-drives while also displaying an intimate knowledge of ICS control surfaces and software. This served as a wakeup-call to the community as it was the first known attack to successfully damage ICS resources outside of an experimental setup. A cyberattack in 2015 plunged much of Ukraine's capitol of Kiev into darkness as multiple substations were rendered inoperable for several hours [4-5]. This complex attack successfully infiltrated the ICS network of several regional energy distributors after a social engineering attack compromised the credentials of facility engineers and staff. Local substations were taken offline by the hackers via legitimate network commands and remote operation became impossible after network interfaces were destroyed. During the Spring of 2021, a simple ransomware attack was launched against Colonial Pipeline, crippling gas supply lines across the Eastern United States [6-7]. This unsophisticated attack was the result of a social engineering attack, highlighting how inattentive employees can send operations to a sudden halt. These cyberattacks showcase a small subset of the challenges that face critical infrastructure networks.

As Industrial Control Systems began to become integrated into the larger organizational network, a new network diagram was proposed to account for these new connections. The Purdue Enterprise Reference Architecture (PERA), also known as the Purdue Model, segments the network into roughly 5 layers [8-9]. At the top of the model, layer 4 represents Enterprise and day-to-day business operations of the organization. This layer is the closest to the wider Internet and is the most likely to be breached by external actors. Next, Facility Command and Control functions are composed of layers 2 (Control Systems) and 3 (Management and Operations). These layers are responsible for housing engineering workstations and other network infrastructure for communicating with embedded systems and Programmable Logic Controllers within the process environment. Finally, layers 0 (Sensors and Actuators) and 1 (Intelligent Devices) constitute the direct process control

functions of the network. Figure 1 presents how the Purdue Model can be applied to Nuclear Facilities. The model includes important network resources such as Human Machine Interfaces (HMI), Data Historians (DH), Programmable Logic Controllers (PLCs), and Remote Terminal Units (RTU). As noted, the Data Historian's logging responsibilities are a great candidate for expanded cybersecurity capabilities in future Nuclear Reactor Facilities.

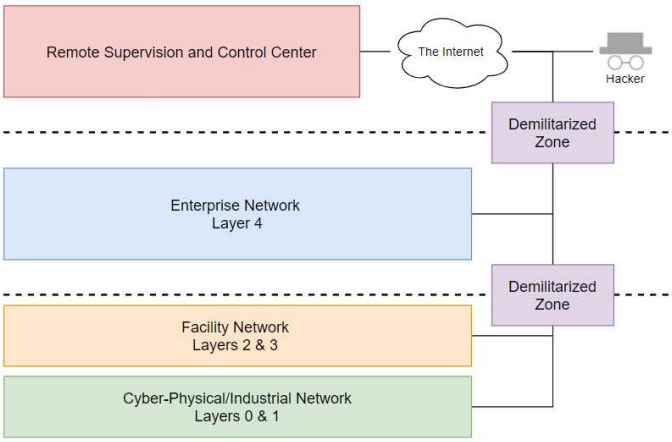

Fig. 1. Industrial Control System network. The facility's network is divided into five layers. The Enterprise Layer contains Layer 4, the Facility Network contains Layers 2 and 3, and the Reactor Network contains Layers 0 and 1.

Through the remainder of this paper, a Novel Data Historian Architecture for Cyberattack Detection using Machine Learning is presented. First, the utilization of the Apache Spark Framework for the Data Historian will be discussed. Next, the machine learning models for cyberattack detection will be presented. These models will be implemented in the Scala programming language and will utilize the Apache Spark MLlib libraries. After discussing the machine learning models, the Industrial Control System dataset for validating the proposed architecture will be discussed. Finally, the results of applying the models will be presented along with discussion of their accuracy at detecting cyberattacks.

II. Data Historian Architecture

Designed to monitor environment conditions, Data Historians (DH) are centralized databases for storing relevant Industrial Control System datasets. Historically, the DH was simply a time-series database provided for data auditing purposes and operations simulation [10-11]. Common implementations include InfluxDB, Graphite, TimescaleDB, Apache Cassandra, and Apache Druid and often have compatibility with relational databases, such as MySQL or Postgres. Data Analytics tools are integrated to enable state-estimation and can potentially provide real-time cybersecurity monitoring capabilities. In recent years, distributed architectures and enhancements have been proposed due to the importance of the device [12-17]. The proposed Data Historian will aggregate data from Human Machine Interface (HMI) commands and reported sensor data for collection by a new Data Analytics component, powered by Apache Spark. Upon collection, the sensor data and commands will be packaged by the Apache Spark Master Node and submitted to Apache Spark Worker Nodes for analysis via our machine learning models. Detected cyberattacks will be reported to the network security administrator and other important metrics will also be provided for external auditing and other expanded use of data. Figure 2 presents the proposed Data Historian Architecture.

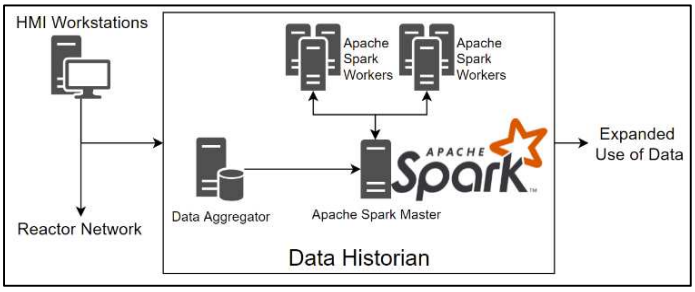

Fig. 2. The Proposed Data Historian Architecture. The architecture contains a data aggregator for collecting data from the Human Machine Interface (HMI) and the network sensors, an Apache Spark Master for assigning work, and Apache Spark Workers for completing assigned tasks. The architecture reports detected cyberattacks to the security team, but can also assist with expanded use of data purposes.

Apache Spark is a cluster computing and data analytics platform with three primary libraries for performing specialized functions [18]. First, Spark SQL libraries power the platform's relational processing and supports data queries similar to traditional SQL databases. Spark SQL will be used throughout the implementation for temporary storage and quick querying of sensor data. The Apache Spark MLlib libraries compromise the core Machine Learning utilities and will be used extensively to implement the models. Finally, the GraphX libraries support graph processing, but will not be used for the DH Architecture. Apache Spark is natively written in Scala, so it will be the primary programming language for development. In addition to Scala, alternative programming languages include Java, Python, and R. Within Spark, resources acquired by the cluster manager are known as Executors while Nodes are composed of entities that perform tasks for the cluster manager. Nodes are often divided across devices to allow for the execution of multiple subtasks simultaneously. Cluster managers can assign tasks and execute them on each node sequentially thanks to tasks queues that are within each executor. To assist with processing large batches of data, Spark utilizes Resilient Distributed Datasets throughout the cluster that are stored within the RAM of each node. When compared to Apache Hadoop, a sister platform, Apache Spark runs workloads at speeds 100 times faster making it an ideal solution for our application. The Apache Spark platform is also able to seamlessly integrate with other Apache platforms such as the Apache Kafka data streaming and Apache Druid time-series database platforms. In future iterations, Apache Kafka will take over data aggregation duties and Apache Druid will be responsible for storing raw sensor data and other alerts from Apache Spark.

The machine learning models will be implemented using Apache Spark's MLlib machine learning framework, which consists of machine learning algorithms, featurization, pipelines, persistence, and other utilities. The machine learning algorithms are composed of a mix of classification and regression models,

clustering algorithms, collaborative filtering, pattern mining, and other tools. Common classification models include Logistic Regression, Decision Tree Classifiers, Linear Support Vector Machines, and Naïve Bayes. Examples of clustering algorithms include K-means, Latent Dirichlet Allocation (LDA), and the Gaussian Mixture Model (GMM). The Apache Spark featurization tools provide all of the necessary preprocessing and post processing tools for data manipulation and feature extraction for the available machine learning models. MLlib's persistence mechanisms allow for the saving and loading of algorithms and pipelines, especially among pre-trained models used in supervised learning. The additional utilities support linear algebra, statistics, and data handling that are useful when working with machine learning models. In addition to these five primary components, MLlib also provides pipelines that define the overall workflow of any given program. The pipelines themselves are composed of DataFrames, Transformers, and Estimators. DataFrames hold machine learning datatypes while Transformers are responsible for transforming DataFrames into other types. The Estimators are any algorithms used on DataFrames that produce a Transformer. For example, a machine learning algorithm that creates a model from raw data would be considered an Estimator.

III. Machine learning models

For this project, four machine learning models were selected for testing within the proposed DH architecture. The first model is the Naïve Bayes classifier, which has a wide range of applications including document labelling. Applications of the classifier operate by treating each feature independently to consider each feature's contribution separately. This works well for classification of objects that have discrete features that may be unrelated. For detecting cyberattacks within Industrial Control Systems, the Naïve Bayes classifier is expected to perform well at identifying cyberattacks that have similar effects on the sensor data and Modbus packets. If the cyberattacks have a diverse set of impacts, performance when only identifying if a cyberattack occurred will suffer. Further, novel cyberattacks will also need to have similar impacts to existing known attacks.

Next, a Logistic Regression Model with elastic net regularization was applied to the dataset. This type of model utilizes the logistic function when estimating the probability of success or failure for the dataset. When making a final prediction, the model will select a threshold value (often 50%) to determine whether the data will be classified as a success or failure. Within the MLlib libraries, two types of regression algorithms can be applied, lasso and ridge. Elastic net regularization combines both of these algorithms to maximize their advantages and minimize their disadvantages. For this implementation, the default parameter of 0.8 was used, which favors lasso regression. The Logistic Regression Model was selected due to its common use in classification problems and expect it to do reasonably well with our ICS dataset.

The final two classifiers are the Decision Tree Classifier and the Random Forest Classifier. Tree Classifiers operate by creating a set of if-else statements that consider different conditions among the features for classifying data. For the Decision Tree Classifier, splitting conditions are generated to maximize the information gain at the next tree node. Information gain is calculated by taking the node impurity and subtracting the corresponding left and right node impurities. When calculating the impurity, the probability of incorrectly classifying the randomly-chosen element in the dataset is measured. Within the Apache Spark framework, three methods are provided for calculating impurity, two are for classification and the third is for regression. For the proposed architecture, Gini impurity was chosen, which involves summing the product of the probability that a given element belongs to a certain class and the probability that it does not for each event. As an extension, the Random Forest Classifier combines multiple decision trees by calculating new trees based on subsamples of the dataset. Within the forest, trees are assigned weight depending on how much of the dataset they sample. In most cases, the trees are weighted equally, which we will do for our classifier. During operation, classification of each element is chosen by applying the weights to each of the trees and arriving at the final classification. The Tree Classifiers are expected to perform well within the proposed architecture as normal behavior of the ICS will fall within normal operating ranges.

IV. Industrial control system dataset

In order to test the machine learning models, a simulated gas pipeline dataset from Thomas H. Morris from the University of Alabama in Huntsville was selected [19]. This dataset was generated for educational and research purposes and simulated real-world ICS behavior of a gas pipeline and injected cyberattacks based on 35 of the most commonly-encountered cyberattacks. The simulated gas pipeline consisted of a pump and solenoid valve, along with devices capable of taking pressure measurements. Varying degrees of randomness were introduced to the device measurements to better model a real-world scenario. Their network included a simulated PLC device, an interactive Human Machine Interface (HMI), the virtual Supervisory Control and Data Acquisition (SCADA) network, and the simulated measurement devices. The simulation supported both manual system operation along with Proportional Integral Derivative (PID) control for automated reaction to system events.

For the resulting dataset, the CSV file contained twenty attributes and 214,580 rows of data observed by the system. Attributes for the environment corresponded to the Modbus communication protocol, PID control values, sensor controls and readings, and a timestamp. Three additional attributes signified whether the data for any given row was representative of a cyberattack. For cyberattacks, the author included a binary result, categorized result, and a specific result. A binary result indicated whether a cyberattack had occurred, the categorized result indicated that one of seven classifications of attacks occurred, and the specific result represented the specific cyberattack that was launched. The cyberattacks performed on the system are representative of typical real-world attacks and are divided into response injection, reconnaissance, denial of service, and common injection. The command injection class is then further divided into malicious state command injection

(MSCI), malicious parameter command injection (MCPI), and malicious function code command injection (MFCI) for a total of seven classes. In total, 214,580 Modbus network packets were logged. Through our testing, we only considered the binary result column but can expand to the categorized and specific categories in the future.

V. Results and Analysis

As a test of the machine learning models, they were applied to the ICS pipeline dataset to determine whether they were able to successfully differentiate between benign communications and cyberattacks. For the experiment the Data Historian Architecture was set to operate in a single-cluster mode with both the Master Node and Worker Node residing on the same machine. The overall dataset was split into a Training subset and a Testing subset with a split of 70%-30% (Training-Testing) to ensure that the models would retain their accuracy outside of the training dataset. Each model was run a total of ten times to achieve an average that would give a reasonable representation of accuracy. The accuracy metric was calculated by comparing the model's prediction against the dataset, with higher accuracy pertaining to more correct identification of cyberattacks or the lack thereof. Table 1 presents a breakdown of the evaluation metrics and overall accuracy of the machine learning models. The true positive, false positive, true negative, and false negative metrics are presented.

TABLE 1
Accuracy and Evaluation Metrics of the Machine Learning Models

Algorithm	True Positive	False Positive	True Negative	False Negative	Accuracy
Naïve Bayes	34.2%	5.7%	15.8%	44.3%	50%
Logistic Regression	76.7%	0%	19.1%	4.2%	95.8%
Decision Tree Classifier	80.2%	0%	19.3%	0.5%	99.5%
Random Forest Classifier	80%%	0%	19.2%	0.8%	99.2%

Based on the results the Naïve Bayes classifier performed poorly as a cyberattack detector with an accuracy of only 50%. While the Naïve Bayes classifier generally performs well at classification, it was no better than a coin flip in terms of its capabilities. This low accuracy was most likely reflective of the high relation between characteristics of the dataset. Within the proposed model, the dataset contained characteristics of both the Modbus packets and the reported sensor/actuator data. For many of the cyberattacks, malicious requests will have a direct impact on the data that is reported to the DH. Since the Naïve Bayes model assumes each variable is independent, it struggled to classify this dataset. In contrast, the Logistic Regression Classifier fared much better than the Naïve Bayes classifier with an average accuracy of 95.8%. The Binomial Logistic Regression Classifier performed well by taking the cumulative distribution function of the logistic distributions for each variable from the dataset. Unsophisticated cyberattacks within ICS result in drastic differences between normal and abnormal readings. By taking a cumulative distribution, the large disturbances caused by the cyberattacks will be captured by the model. The model then selected a probability threshold based on the training dataset for classifying between normal behavior and cyberattacks.

As shown, the Decision Tree Classifier and Random Forest Classifier performed the best of all models with an accuracy of 99.5% and 99.2%, respectively. Both of these classifiers were fairly similar, with the Random Forest Classifier being composed of ten decision trees, each given equal weight when making a new classification. Since the decision trees created boundaries for the dataset attributes, the trees would attempt to conform to the normal sensor readings that are seen during regular operation of the system. As such, any attack that relied on false data injection or resulted in a drastic deviation from normal operation would be identified by these models. Since both models relied on these trees, they performed the best throughout our tests. Initially, the Random Forest Classifier was expected to provide superior performance but ended up having a negligible effect on the results, having a slightly lower accuracy.

In terms of the evaluation metrics, all four machine learning models favored false negatives over false positives. For cyberattack detection, false positives refer to normal traffic labeled as a cyberattack whereas a false negative refers to a cyberattack labeled as normal traffic. In a production environment it is preferred to favor false positives as the mislabel will have a minor impact on the performance on the system. If false negatives are preferred, more cyberattacks will go undetected, which may have devastating consequences on the network. Of the machine learning models, the Decision Tree Classifier had the lowest false negative percentage at 0.5% of the dataset.

VI. Conclusion

Altogether, the proposed machine learning models were successful at detecting cyberattacks. A Data Historian Architecture for deploying these models was composed of an Apache Spark cluster and utilized the Apache Spark MLlib machine learning libraries to implement our models. The Naïve Bayes classifier was shown to be inadequate for detecting cyberattacks within this dataset with an average accuracy of 50% while the Logistic Regression and Random Forest Classifier were viable alternatives with 95% and 99.2% accuracy, respectively. The Random Forest Classifier had the highest accuracy with 99.5% accuracy when detecting cyberattacks. Additionally, all models favored false negatives, which is not desired. In the future, the Data Historian Architecture can be expanded by introducing the Apache Kafka data streaming platform along with the Apache Druid time-series database application. These additions would allow the current system to collect data from the system's Programmable Logic Controllers via Apache Kafka and store data and analysis within the Apache Druid databases. Additionally, the machine learning models can be extended to specify the type of cyberattack that is being launched against the network.

ACKNOWLEDGEMENTS

This material is based upon work supported under an Integrated University Program Graduate Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the Department of Energy Office of Nuclear Energy.

REFERENCES

[1] M. Baezner and P. Robin, "Stuxnet," ETH Zurich , Zurich, 2017.

[2] A. Matrosov, E. Rodionov, D. Harley and J. Malcho, "Stuxnet Under the Microscope," ESET LLC, 2010.

[3] K. Zetter, "An Unprecedented Look at Stuxnet, the World's First Digital Weapon," Wired, 3 November 2014. [Online]. Available: https://www.wired.com/2014/11/countdown-to-zero-day-stuxnet/. [Accessed 15 July 2021].

[4] R. Lee, M. Assante and T. Conway, "Analysis of the Cyber Attack on the Ukrainian Power Grid," E-ISAC, Washington DC, 2016.

[5] K. Zetter, "Inside the Cunning, Unprecedented Hack of Ukraine's Power Grid," Wired, 3 March 2016. [Online]. Available: https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/. [Accessed 15 July 2021].

[6] U.S. Government Accountability Office , "Colonial Pipeline Cyberattack Highlights Need for Better Federal and Private-Sector Preparedness (infographic)," U.S. Government Accountability Office, 18 May 2021. [Online]. Available: https://www.gao.gov/blog/colonial-pipeline-cyberattack-highlights-need-better-federal-and-private-sector-preparedness-infographic. [Accessed 15 July 2021].

[7] The White House, "FACT SHEET: The Biden-Harris Administration Has Launched an All-of-Government Effort to Address Colonial Pipeline Incident," The White House, 11 May 2021. [Online]. Available: https://www.whitehouse.gov/briefing-room/statements-releases/2021/05/11/fact-sheet-the-biden-harris-administration-has-launched-an-all-of-government-effort-to-address-colonial-pipeline-incident/. [Accessed 15 July 2021].

[8] T. Williams, "The Purdue Enterprise Reference Architecture," *Computers in Industry,* vol. 24, no. 2, pp. 559-564, 1994.

[9] H. Li and T. Williams, "The interconnected chain of enterprises as presented by the Purdue Enterprise Reference Architecture," *Computers in Industry,* vol. 42, pp. 265-274, 2000.

[10] B. Chardin, J.-M. Lacombe and J.-M. Petit, "Data Historians in the Data Management Landscape," in *Technology Conference on Performance Evaluation and Benchmarking*, Berlin, 2012.

[11] S. K. Jensen, T. B. Pedersen and C. Thomsen, "Time Series Management Systems: A Survey," *Transactions on Knowledge and Data Engineering,* vol. 29, no. 11, pp. 2581-2600, 2017.

[12] M. Edwards, A. Rambani, Y. Zhu and M. Musavi, "Design of Hadoop-based Framework for Analytics of Large Synchrophasor Datasets," *Procedia Computer Science,* vol. 12, pp. 254-258, 2012.

[13] D. Zhou, J. Guo, Y. Zhang, J. Chia, H. Liu, Y. Liu, C. Huang, X. Gui and Y. Liu, "Distributed Data Analytics Platform for Wide-Area Synchrophasor Measurement Systems," *IEEE Transactions on Smart Grid,* vol. 7, no. 5, pp. 2397-2405, 2016.

[14] D. Mazur, R. Entzminger and J. Kay, "Enhancing Traditional Process SCADA and Historians for Industrial & Commercial Power Systems with Energy (Via IEC 61850)," in *2014 IEEE/IAS 50th Industrial & Commercial Power Systems Technical Conference*, 2014.

[15] A. Ayodeji, Y.-k. Liu, N. Chao and L.-q. Yang, "A new perspective towards the development of robust data-driven intrusion detection for industrial control systems," *Nuclear engineering and technology,* vol. 52, no. 12, pp. 2687-2698, 2020.

[16] R. Khan, K. McLaughlin, B. Kang, D. Laverty and S. Sezer, "A Seamless Cloud Migration Approach to Secure Distributed Legacy Industrial SCADA Systems," in *2020 IEEE Power & Energy Society Innovative Smart Grid Technologies Conference*, 2020.

[17] A. Nicolae, A. Korodi and I. Silea, "Identifying Data Dependencies as First Step to Obtain a Proactive Historian: Test Scenairo in the Water Industry 4.0," *Water,* vol. 11, no. 6, pp. 1144-1167, 2019.

[18] Apache Software Foundation, "MLib is Apache Spark's Scalable Machine Learning Library," Apache Software Foundation, [Online]. Available: https://spark.apache.org/mllib/. [Accessed 15 July 2021].

[19] T. Morris, Z. Thornton and I. Turnipseed, "Industrial Control System Simulation and Data Logging for Intrusion Detection System Research," in *7th Annual Southeastern Cyber Security Summit*, 2015.