Evaluation of Machine Learning Algorithms for Anomaly Detection in Industrial Networks

Giuseppe Bernieri, Mauro Conti, Federico Turrin
Department of Mathematics
University of Padua
Padua, Italy
{bernieri, conti}@math.unipd.it, federico.turrin@studenti.unipd.it

***Abstract*—The cyber-physical security of Industrial Control Systems (ICSs) represents an actual and worthwhile research topic. In this paper, we compare and evaluate different Machine Learning (ML) algorithms for anomaly detection in industrial control networks. We analyze supervised and unsupervised ML-based anomaly detection approaches using datasets extracted from the Secure Water Treatment (SWaT), a testbed developed to emulate a scaled-down real industrial plant. Our experiments show strengths and limitations of the two ML-based anomaly detection approaches for industrial networks.**

***Index Terms*—Machine Learning, Anomaly Detection, Industrial Control System, Cyber-Physical System, Security.**

I. Introduction

Modern Supervisory Control And Data Acquisition (SCADA) systems monitor physical devices (e.g., actuators, sensors), processes, and events characterizing the industrial plants. The increasing and pervasive interconnection capabilities of ICSs improve functioning but open new cyber-physical threat scenarios. Most communications are performed device-to-device, or device-to-computer, with relatively little human interaction. Nowadays, we can consider the Cyber-Physical Systems (CPSs) as the foundation of many applications in ICSs that need actuators and sensors to perform monitoring. Some ICS examples include refineries, power plants, nuclear plants, and water distribution systems.

During the last few years, numerous malware targeting ICSs have been documented. The most famous is Stuxnet [1], while more recent ones, such as Triton [2] and BlackEnergy [3], highlight the security vulnerabilities of ICSs. Problems to ICSs can cause devastating consequences, therefore it is very important to study and develop innovative cyber-physical security methods to protect these systems. Furthermore, since specific ICSs are categorized as critical infrastructures, damaging or destroying them can cause serious problem to the population and the environment.

Nowadays, the industrial infrastructures produce huge amounts of data. Due to this incredible load of traffic, it is not easy to filter, analyze, and make operational and security decisions for ICSs. To overcome these problems, ML and artificial intelligence have been introduced in the decision-making processes. In fact, these techniques allow to analyze very large amounts of data and provide accurate decisions that would otherwise be unaffordable for analysts. The research in this field is very active and in recent years it has made incredible progresses. The integration of anomaly detection systems with ML and artificial intelligence allows to obtain greater performances and accuracy compared to traditional techniques. Therefore, it is necessary to analyze and develop accurate and novel anomaly detection systems in order to prevent possible damages to ICSs.

We summarize our contributions as follows:

- we perform an analysis of supervised and unsupervised ML-based algorithms for anomaly detection.
- we exploit the SWaT testbed dataset to evaluate the ML-based anomaly detection methods identified.
- we discuss how ML-based techniques can represent an added value for the detection of threats affecting ICSs.

The rest of the paper is organized as follows: Section II presents the related work. Section III introduces the dataset considered for this work. Section IV briefly describes the ML techniques used. Section V discusses the results of our experiments with the ML-based techniques for anomaly detection. Section VI concludes the paper.

II. Related Work

Anomaly detection for ICSs is a largely studied problem in the literature. However, the scientific community is afflicted by the scarcity of available datasets on which perform evaluations. Lemay and Fernandez [4] provide a dataset of a SCADA system simulated with a sandbox. In this scenario, different data collections have been performed. An exhaustive description of the scenario simulated can be found in their work. This work introduces malicious activities exploited using penetration testing tools such as *metasploit*. The use of *metasploit* as a source of attack is one of the most prevalent drawbacks of the employed dataset: there are several Modbus-based attacks mentioned in [5], but none of these are introduced in the dataset. However, in-depth studies were performed on the Lemay dataset, for example in [6] the authors compared supervised and unsupervised algorithms, showing the superiority of supervised ones. Another approach to investigate ICS cyber-physical security is to use real ICS testbeds and develop ad hoc offensive and defensive techniques, such as the anomaly detection systems presented in [7], [8].

In this work, we consider the SWaT testbed dataset [9] and, in particular, we will focus on the analysis of physical data.

The dataset from SWaT includes 36 different cyber attacks. Several studies have been done on this dataset in order to observe the performance of various types of ML detection algorithms. In [10], the authors used unsupervised algorithms to identify attacks on physical dataset sensors. In [11], the authors proposed a window-based anomaly detection method, where a neural network model predicts the future values of the data features based on previous values.

There are two ways to manage the data in an anomaly detection system. The first is through the analysis of the individual elements, while the second is done by grouping the elements in time series and analyzing the collapsed elements. While the former makes it possible to identify the individual malicious elements, the latter does not allow it, but on the other hand, it allows to exploit temporal features obtained from the union of several elements. In this work, we chose the first analysis approach in order to identify every single malicious element. Furthermore, unlike the previously mentioned works, we compare both supervised and unsupervised techniques in order to show the main differences.

III. ICS Testbed and Datasets

In this Section, we initially remark the availability importance of ICS datasets (Section III-A), useful to develop accurate strategies and methodologies to prevent malicious events. Then, we introduce the SWaT testbed (Section III-B) and the dataset used for the experiments (Section III-C).

A. Data availability problem for ICS security research

The security community requires cyber-attack datasets to discover, test, and evaluate new detection and prevention methodologies in order to understand patterns and impacts of cyber-physical attacks against ICSs. However, due to the lack of available datasets containing attacks in ICS networks, it is not easy to carry on innovative security research. The lack of available datasets is due to privacy concerns and also because it is not possible to test cyber-physical attacks on real-world critical systems. While some research groups are able to investigate novel security methods using professional testbeds [12] or low-cost ones [13], implementation and evaluation of cyber-physical security strategies and cyber-physical attacks represents an impediment in most of the cases. To overcome this problem, research groups provide useful datasets of their testbed communication. One example is the SWaT testbed with its dataset, described in the following Sections.

B. The Secure Water Treatment (SWaT) testbed

The SWaT testbed was built for the Singapore University of Technology and Design (SUTD) in order to provide a realistic ICS environment to safely test offensive and defensive cyber-physical strategies. The testbed operation processes are described in Figure 1. The testbed represents a real-world scaled down water treatment plant that produces purified water. The water goes through a six-stage filtration process from process P1 to process P6 and each stage is supplied with a precise number of sensors and actuators. For more information about the SWaT testbed, please refer to [14].

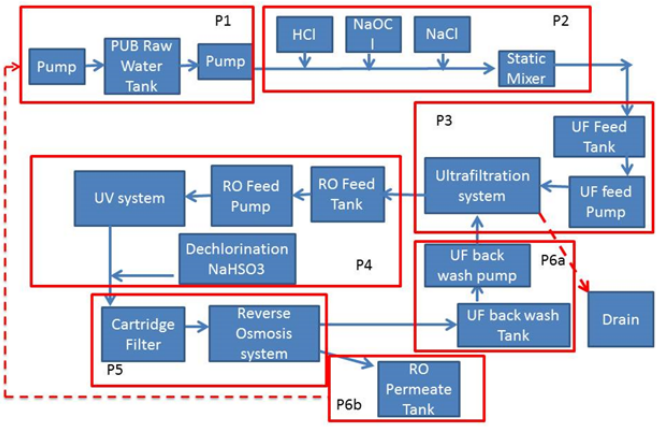

Fig. 1. SWaT testbed processes overview [9].

C. The SWaT dataset

The SWaT dataset contains seven days of data recording under normal conditions and others four days of recording where a total of 36 attacks were conducted. The dataset contains a total of 946722 elements and each of them is labeled as either attack or normal. Attacks include both targeting to a single stage of the process and targeting to different stages simultaneously. An exhaustive table containing all the attacks, the corresponding times, attack points outcomes is provided in the dataset documentation.

A shortcoming of the dataset is that, as the authors say in [9], the data was captured with a per-second interval, so there are instances of overlap where multiple rows reflect a different activity but carry the same timestamp. Similarly, based on the attack logs, the data was labeled based on the start and end times of the attacks without distinguishing between malicious and normal elements. Therefore, these problems affect data purity and may compromise the final results of the analysis.

Among all the 36 attacks performed, we choose the attack number 3 for the analysis in this work. The file considered contains both normal and malicious data together with the corresponding label. The dataset consists of scalar values recorded by sensors in constant time intervals. The attack performed during this time interval consists of increasing the water level by 1 mm every second. The result is a tank overflow that damages the P-101 sensor. These effects could be destructive in a real scenario. Since there are many sensors inside the dataset, we decided to divide the dataset as follows:

- **ds1**: this dataset contains the data of all the sensors of the testbed during the attack stage. This dataset is studied in order to observe if the alteration of a sensor (in this case LIT-101) influences the other sensors, and therefore provides additional information for the detection.
- **ds2**: this dataset contains only the data of the sensor under attack, that is the sensor LIT-101.

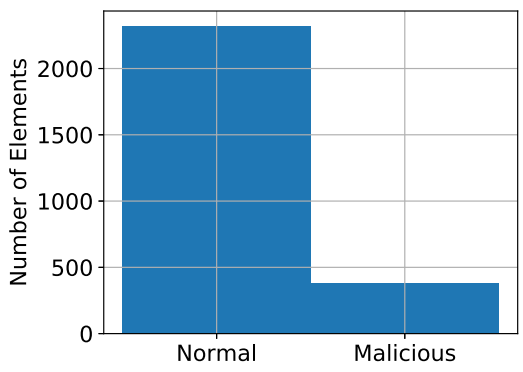

Fig. 2. Normal and malicious elements in ds1 and ds2 derived datasets.

Each of the derived datasets contains a total of 2701 elements, of which 383 are malicious, that is about 14.18%. Normal and malicious number of elements are sketched in Figure 2.

IV. ML-based Anomaly Detection Methods

In this Section, we present the ML-based algorithms used to perform our evaluation. We compare both supervised (Section IV-A) and unsupervised (Section IV-B) algorithms in order to observe the strengths and weaknesses of the two approaches.

A. Supervised Algorithms

a) Support Vector Machine: in 1992, *Boser et al.* introduced the Support Vector Machine (SVM) [15]. SVM is a supervised ML algorithm which can be used for both classification and regression. The main idea behind this framework is to create a divider, the so-called *large margin classifier*, between two groups of data such that each element has the greatest distance from the divider. The input of the training algorithm is a set of m examples x_i with labels y_i as follows:

$$(x_i, y_i) \quad i = 1, \ldots, m, \tag{1}$$

where each y_i is equal to:

$$y_i = \begin{cases} 1 & \text{if } x_i \text{ belongs to the first class} \\ -1 & \text{if } x_i \text{ belongs to the second class} \end{cases}, \tag{2}$$

Once the training phase is over, the decision function, namely *signum function*, is defined as follows:

$$z_i = sgn(w, x_i - b), \tag{3}$$

where w is the normal vector of the *large margin classifier*, b is the offset from the hyperplane, and z is the output vector which describes the attribution to one of the two classes. There are two possible cases for the application of SVMs: the instances can or cannot be divided by a linear function. If the instances are not linearly divisible, a transformation called *kernel trick* [16] is applied. This trick consists of mapping non-linearly the input space into a higher dimensional feature space, where the algorithm can create a linear divider.

b) Random Forest: the *Random Forest* (RF) is a supervised learning algorithm used for both classification and regression problems, composed of an union of *Decision Trees* [17] which are populated during the training phase. RF consists of a root node, internal nodes (the so-called split nodes), and leaf nodes. Each class predicted by the algorithm corresponds to a leaf node. The assignment of an element to a class is made by a majority voting. RF performs well under noise or overfitting conditions, which are very common problems in ML.

c) k-Nearest Neighbour: k-Nearest Neighbour (KNN) algorithm is a non-parametric algorithm [18] used for both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. In fact, the classification is based on the characteristics of objects close to the one considered using specific distance metrics, in general, the Euclidean distance. The Euclidean distance is defined as follows:

$$D = \sqrt{\sum_{i=1}^{n} (x_i - p_i)^2}, \tag{4}$$

where x_i and p_i are the components of n-dimensional vectors. Once the distance between the considered point and all the others is calculated, the class of the considered point is assigned as the most frequent among the first k points with the smaller distance.

B. Unsupervised Algoritms

a) One-Class SVM: the One-Class Support Vector Machine (OCSVM) [19] is a special case of the conventional SVM. OCSVMs are trained using only one class representing the *normal* behavior. Specifically, the OCSVM is a One-Class Classification (OCC) method. OCCs try to identify objects of a specific class among all objects by creating a decision boundary. To do this, the OCSVM model is primarily trained with a set of elements containing only the objects of the specific class. Thus, having all the training elements with the same label is equivalent to having no labeled elements. For these reasons, the OCSVM is considered an unsupervised learning model [20], [21].

Once the OCSVM model is trained with the *normal* class, it is able to infer the properties of the *normal* elements and from these properties, it can predicts which elements do not belong to the *normal* class. The elements that do not belong to the *normal* class are called *outliers*.

b) Autoencoder: Autoencoders (AEs) [22] are fully-connected neural networks trained to reconstruct their input training sets. The AEs learn a "compressed representation" of the input (could be an image, text sequence, etc.) automatically, by first compressing the input (encoder) and decompressing it back (decoder) to match the original input. An AE network consists of three parts: one input, one output, and one (or more) hidden layer. While the input and output layers need to be of the same size, the size of the hidden

layer can be adapted to the use case. Over the last couple of years, this kind of neural network is getting popularity in real-world problems, including the anomaly detection. In fact, an AE trained on a test set X gains the capability to reconstruct unseen instances from the same data distribution as X. This kind of detection is called *Reconstruction-based detection* [23]. If an instance does not belong to the concepts learned from X, then we expect the reconstruction to have a high error. The Reconstruction Error (RE) of the instance $\bar{x}$ for a given AE can be computed by taking the Root Mean Squared Error (RMSE) between $\bar{x}$ and the reconstructed output $\bar{y}$. The RMSE between two vectors is defined as:

$$RMSE(\bar{x}, \bar{y}) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}, \tag{5}$$

where n is the dimension of the input vectors. Threshold selection is a crucial point in the development of AE. A too high threshold may increase the number of false negatives, while a too low threshold may increase the number of false positives. In our case, the AEs are trained with only normal class, so the threshold selection approach is based on the one used in [24]. A 20% validation set is extracted from the training set (which contains only normal elements), and after having trained the neural network, the maximum among the REs on the validation set is set as the threshold value.

V. Evaluation and Results

In this Section, we discuss the implementation details of the ML algorithms in Section V-A and the evaluation metrics used to compare the ML-based anomaly detection algorithms in Section V-B. Then, we describe the results obtained from the experiments in Section V-C.

A. Implementation Details

To perform the evaluation, we split each dataset in 80% for training and 20% for the detection test. In the case of the unsupervised algorithm, we exploit a semi-supervised learning method for the training. This means that the models are trained with the subset of normal elements extracted from the training set. In order to implement the SVM, RF, KNN, and OCSVM algorithms, we use the *Scikit-learn* library [25] for Python. Given a training set X_{train} and the corresponding labels y_{train}, all the estimators classes in the *Scikit-learn* library implement a `fit(`X_{train}, y_{train}`)` method to fit the model, which in the case of unsupervised algorithms, such as the OCSVM, take only the X_{train} value. To classify the unlabeled observations X_{test}, the *Scikit-learn* library implements the method `predict(`X_{test}`)`, which returns the predicted labels $\hat{y}_{test}$. To implement the AE neural network, we use the *Keras* [26] library. One of the main difficulties in the implementation of neural networks concerns the parameters tuning task. The AE model implemented is composed of an input layer and an output layer with the same dimension of the features number. The number of neurons in hidden layers is half of the input and output layers' neurons. In the case of **ds2**, where there is only one feature, the number of neurons is set to 1. We perform the evaluation on a laptop with the following specifications:

- CPU: Intel(R) Core(TM) i7-3537U 2.00GHz x 4.
- RAM: 10GB DDR3.
- Operative System: Ubuntu 18.10 64-bit.

B. Evaluation Metrics

For the evaluation of the ML-based algorithms for anomaly detection, we use the following metrics:

- **Accuracy**: represents the fraction of correct predictions of the model under consideration. In the binary classification case, the accuracy is defined in terms of positives and negatives as follows:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN}, \tag{6}$$

  where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.
- **F1-Score**: is a metric used to evaluate a classification by taking in consideration both *precision* and *recall* as follows:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \tag{7}$$

  where:

$$precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}. \tag{8}$$

C. Results

In Table I, we present the results obtained for the ML-based anomaly detection algorithms for the datasets under consideration. The dataset **ds1** contains the recording of all the sensor data, while **ds2** contains the data of only the sensor LIT-101, which is the one under attack. The results show that all the algorithms, except for the AE, perform better in the case of **ds1**, that represents the derived dataset characterized by all the sensors values. This means that the attack against the LIT-101 sensor also affects the state of the other sensors, which contributes to the identification of the malicious behavior. Furthermore, RF and KNN obtain a detection score of 1.0 on **ds1**, which means that their detection approaches are very efficient on this data. Also SVM performs well on this dataset, it fails to identify only one element, meaning that the data can therefore be separated from a hyperplane. OCSVM proves to be a discrete algorithm, achieving worse performance than supervised algorithms, but better than AE. Again, the result underlines the divisibility of the data by the hyperplane, which is able to better categorize the data compared to the RE-based approach. In fact, the F1-score of AE on **ds1** is null, this means that none of the malicious elements have been recognized. In the case of **ds2**, that is in the dataset in which only the sensor under attack is considered, the performances are generally worse. This is due to the fact that the information obtained from the sensors under attack is not sufficient to identify the malicious behavior. We show the graphical representation of the Accuracy for all the algorithms in Figure 3, while the representation of the F1-Score is in Figure 4.

TABLE I
THE RESULTS OF THE ML-BASED ANOMALY DETECTION ALGORITHMS ON THE DATASETS.

Dataset	ds1		ds2	
Algorithm\Metric	Accuracy	F1-Score	Accuracy	F1-Score
SVM	0.9963	0.9868	0.9242	0.6238
RF	1.0	1.0	0.9371	0.7069
KNN	1.0	1.0	0.9316	0.6782
OCSVM	0.8465	0.9024	0.8428	0.9050
AE	0.8539	0.0	0.9168	0.5710

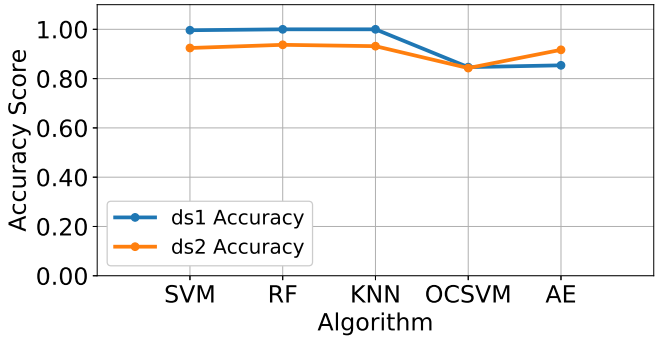

Fig. 3. Accuracy representation for ML-based anomaly detection algorithms.

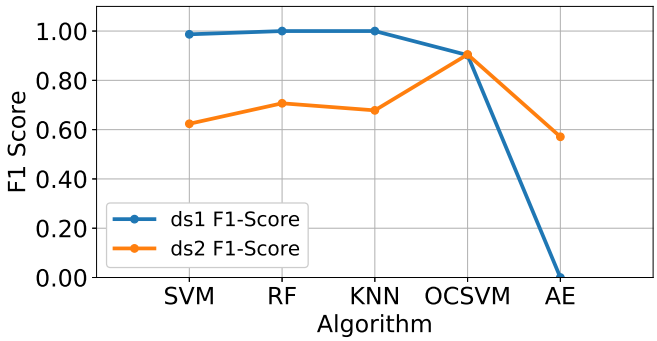

Fig. 4. F1-Score representation for ML-based anomaly detection algorithms.

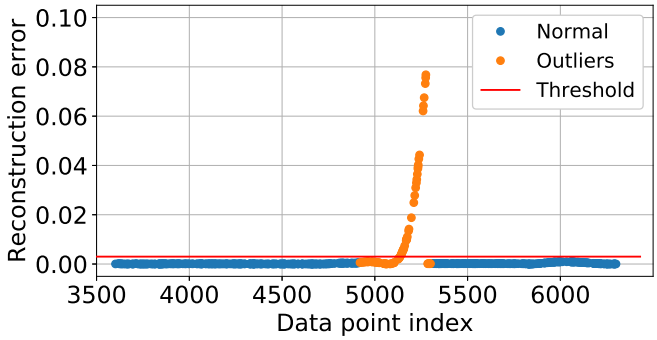

Fig. 5. Autoencoder Reconstruction Error on ds2.

In order to investigate the lower scores obtained by all the algorithms on **ds2** with respect to **ds1**, we analyze the RE of AE on this dataset. The analysis of the RE allows to easily visualize the distance of the elements of the test set, with respect to the compressed representation of the training set memorized during the AE training phase. The results are shown in Figure 5. Since AE is trained with only elements labeled as normal, when a prediction is performed on the test set, elements close to the normal ones will have a low RE value, while a high RE value will be obtained for abnormal elements.

Figure 5 shows that elements are labeled as malicious even if the physical state of the device LIT-101 has not yet begun to vary and therefore it is still considered normal. Consequently, the low accuracy score is caused by a labeling shortcoming of the dataset. This involves a low F1-score despite the correct use of the algorithm. This problem is caused by the fact that the starting point of the malicious labeling was set at the attack start time, instead of the instant when data started to vary. This problem, although displayed on AE, also appears for the other algorithms in the case of the **ds2** analysis.

D. Discussion on ML-based anomaly detection algorithms for ICS security

Considering the results obtained, we identified that the algorithms using supervised learning have generally better performance than the unsupervised ones. This is given by the fact that the supervised algorithms use a priori knowledge given by the labeling of the dataset. On the contrary, the unsupervised algorithms do not use any kind of information, but instead, they look for latent information in the data in order to find common characteristics. The supervised approach has a fundamental limitation: it necessarily requires labeling of the dataset to perform the model training. After a more accurate analysis of the AE results, we note that there is a labeling problem related to the dataset (the problem also appears in the other algorithms). In fact, data with values equal to normal are also labeled as anomalous, thus leading to a low F1-Score and then the presence of many FNs elements.

As shown in the experiments, machine learning techniques can be very powerful in anomalies identification. In the ICS environments, communications between different devices are characterized by a polling time, and therefore are constant and repeated. The use of intelligent and self-adaptive techniques allow to identify anomalies even where a human fails, especially when the amount of data to be managed is huge.

VI. CONCLUSIONS

In this paper, we evaluated ML-based algorithms for anomaly detection in industrial control networks. Specifically, we analyzed supervised and unsupervised approaches using datasets extracted from the SWaT testbed. The tests performed show the potentials and the limitations of supervised and unsupervised algorithms applied for anomaly detection on ICS networks datasets.

For future work, we will implement the ML-based approaches identified on ad hoc ICS simulation networks and

real ICS testbeds to deepen the analysis and develop effective anomaly detection systems.

VII. Acknowledgments

This work was supported by the European Commission under the Horizon 2020 Programme (H2020), as part of the LOCARD project (Grant Agreement no. 832735). This work is also supported by a grant of the Italian Presidency of the Council of Ministers. The authors would like to thank the *iTrust, Centre for Research in Cyber Security, Singapore University of Technology and Design* for the SWaT testbed datasets.

References

[1] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[2] R. M. Lee, *TRISIS: Analyzing Safety System Targeting Malware*. DRAGOS, 2017.

[3] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *SANS Industrial Control Systems*, vol. 23, 2016.

[4] A. Lemay and J. M. Fernandez, "Providing {SCADA} network data sets for intrusion detection research," in *9th Workshop on Cyber Security Experimentation and Test ({CSET} 16)*, 2016.

[5] T. Morris and W. Gao, "Industrial control system traffic data sets for intrusion detection research," in *International Conference on Critical Infrastructure Protection*. Springer, 2014, pp. 65–78.

[6] S. D. Anton, S. Kanoor, D. Fraunholz, and H. D. Schotten, "Evaluation of machine learning-based anomaly detection algorithms on an industrial modbus/tcp data set," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*. ACM, 2018, p. 41.

[7] H. R. Ghaeini and N. O. Tippenhauer, "Hamids: Hierarchical monitoring intrusion detection system for industrial control systems," in *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2016, pp. 103–111.

[8] E. E. Miciolino, R. Setola, G. Bernieri, S. Panzieri, F. Pascucci, and M. M. Polycarpou, "Fault diagnosis and network anomaly detection in water infrastructures," *IEEE Design & Test*, vol. 34, no. 4, pp. 44–51, 2017.

[9] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *International Conference on Critical Information Infrastructures Security*. Springer, 2016, pp. 88–99.

[10] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun, "Anomaly detection for a water treatment system using unsupervised machine learning," in *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2017, pp. 1058–1065.

[11] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*. ACM, 2018, pp. 72–83.

[12] G. Bernieri, E. E. Miciolino, F. Pascucci, and R. Setola, "Monitoring system reaction in cyber-physical testbed under cyber-attacks," *Computers & Electrical Engineering*, vol. 59, pp. 86–98, 2017.

[13] G. Bernieri, F. Del Moro, L. Faramondi, and F. Pascucci, "A testbed for integrated fault diagnosis and cyber security investigation," in *2016 International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2016, pp. 454–459.

[14] A. P. Mathur and N. O. Tippenhauer, "Swat: a water treatment testbed for research and training on ics security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. IEEE, 2016, pp. 31–36.

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*. ACM, 1992, pp. 144–152.

[16] V. Vapnik, "Universal learning technology: Support vector machines," *NEC Journal of Advanced Technology*, vol. 2, no. 2, pp. 137–144, 2005.

[17] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[18] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[19] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[20] L. A. Maglaras and J. Jiang, "Intrusion detection in scada systems using machine learning techniques," in *2014 Science and Information Conference*. IEEE, 2014, pp. 626–631.

[21] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A geometric framework for unsupervised anomaly detection," in *Applications of data mining in computer security*. Springer, 2002, pp. 77–101.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[23] D. Zimmerer, S. A. Kohl, J. Petersen, F. Isensee, and K. H. Maier-Hein, "Context-encoding variational autoencoder for unsupervised anomaly detection," *arXiv preprint arXiv:1812.05941*, 2018.

[24] H. A. Dau, V. Ciesielski, and A. Song, "Anomaly detection using replicator neural networks trained on examples of one class," in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2014, pp. 311–322.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[26] F. Chollet *et al.*, "Keras: Deep learning library for theano and tensorflow," *URL: https://keras.io*, vol. 7, no. 8, p. T1, 2015.