Explainable AI for Process-Aware Attack Detection in Industrial Control Systems

Léa Astrid KENMOGNE, Stéphane Mocanu

Explainable AI for Process-Aware Attack Detection in Industrial Control Systems

Léa Astrid KENMOGNE
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
38000 Grenoble, France
lea.kenmogne-mekemte@univ-grenoble-alpes.fr

Stéphane MOCANU
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
38000 Grenoble, France
stephane.mocanu@grenoble-inp.fr

***Abstract*—Industrial Control System cybersecurity has become an important study area after the occurrence of several mediatic events in the 2010's (Stuxnet, BlackEnergy, Industroyer). Two common characteristics of these attacks are the fact that they were not violating the communication protocols being "stealth" for classical pattern-based detection methods and that they explicitly target the physical process. In this paper we study the performance and explainability of an artificial intelligence based detection system for the detection of such sophisticated attacks.**

***Index Terms*—Explainable Artificial Intelligence, Anomaly-based detection, Industrial Control Systems**

I. Introduction

A. ICS cybersecurity

Industrial control systems (ICS) are distributed computing architectures dedicated to the control of a physical plant. They are requested to comply with real-time response time constraints and are hardened for industrial environment. They are directly interacting with the physical process through sensors and actuators in order to accomplish a control objective (like trajectory control, for instance) while keeping the physical process and its environment safe. Traditionally, they are not connected to Internet and do not include cybersecurity controls. With the pervasive deployment of Internet technologies and remote interconnections, they become exposed to cyberthreats.

Due to their critical mission, successful attacks on ICS may have dramatic consequences for the safety of the physical plant, the environment or the operators as the seen in (already) classical ICS attacks like Maroochy Shire, Aurora attack, Stuxnet, BlackEnergy, Industroyer [1]. Therefore, cybersecurity of ICS become an important research domain especially in the last fifteen years.

There are several common characteristic of the aforementioned ICS cyberattacks. They have exploited the leak of confidentiality and authentication in the communication protocol and they were specifically targeting the physical process by manipulating the values of sensors or the set points of the actuators without violating the syntax or semantics of the communication protocols. Such attacks are called in the literacy *process-aware attacks* [2], and they are usually difficult to detect as a classical analysis of communication protocols headers is not sufficient.

This work has been partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22-PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

B. Process-aware attacks detection

Process-aware attacks will attempt to trigger an legitimate action in a an wrong process context, for instance, opening the filling valve of a tank while the tank is full and overflowing. They include manipulation of the sensors and actuators values, forcing the execution of programs or modifying the control program. A good overview of the various techniques is provided by the MITRE ATT&CK® for Industrial Control Systems matrix [3].

As the context (state of the physical process) is needed in order to decide is a frame is legitimate or malicious the pattern-based intrusion detection technique is not adequate for process aware attacks detection. Therefore, behavior-based anomaly detection techniques are studied. Several approaches were developed based on the regularity of the traffic [4], the analysis of the values of process variables [5] or the violation of the safety properties of the physical process [2], [6].

In this work, we are interested in the safety properties violation approach as it is explicitly relates to cybersecurity violation and the physical process safety jeopardizing. More precisely, we are interested into the application of explainable artificial intelligence (XAI) techniques for learning safety properties from the network traffic and detecting and explaining the attacks that violate these properties.

The rest of the paper is organized as follows: in Section II, we review the existing literature on intrusion detection in ICS. In Section III, we propose the detection model coupled with the AI-based explainable module. A practical test use case of our module is described in Section IV, and the results obtained are presented in Section V. Section VI concludes the work by outlining the various perspectives.

II. State of the Art

In this section, we review two distinct categories of results on intrusion detection systems (IDS) : the techniques based on security properties (not necessarily AI) and the XAI approaches for ICS (not necessarily security based). Both categories are needed to position our work.

A. Security properties based IDS

Behavioral intrusion detection based on safety properties needs, firstly, a methodology to identify the needed properties then express them in an efficient manner in order to allow an effective IDS implementation. We focus mainly on approaches based on *runtime monitoring* as they are able to provide a fast response time. In such an approach, safety properties are expressed as *security patterns*. A security pattern expressing the fact that the filling valve of a tank shall never be open if the tank is full may be described as : $never(valve\_open)when(tank\_full)$. The use of such security patterns based on the temporal logic of signals is classic in computer security [7]–[9]. The main difficulty in ICS is the identification of patterns. Some approaches, like [10], will attempt to mine predefined patterns into the network traffic; others, as [6] will attempt to extract patterns from standard specifications of the control system. The inconvenient of the mining approach is that it may identify redundant patterns, and therefore, a supplementary optimization step is needed [10]. The standard-based approach relies on the availability and completeness of the standards and then fails to identify non-standard security properties.

Our approach does not use predefined patterns, and we do not rely on standards. We build a learning model containing process state variables features then we train the model on a normal traffic dataset. In the detection stage we investigate the impact of each feature.

B. XAI IDS for ICS

In existing works based on machine learning methods, some authors have started with a benchmark study in which they evaluate the model on several different algorithms and compare the different results. This is the case of [11], in which they train the model on supervised machine learning algorithms. The main limitation of this work is the use of supervised techniques, as they require the model to be provided with datasets containing a wide variety of attacks, which is not suitable for our purpose. Other authors have oriented their work toward unsupervised techniques, more specifically, autoencoders. It is in this context that the work carried out by Do Thu et al. [12] is set. They combine a Long Short Term Memory (LSTM) Autoencoder and a One Class Support Vector Machine (OCSVM) for detection. Firstly, the LSTM Autoencoder is used to train the model which is then used by the OCSVM to classify the samples (abnormal if it is out of the boundary and normal otherw). Additionally, they introduce a SHAP explainability module to help understand the model predictions.

Explainability in machine learning refers to the ability to understand and explain the decisions made by the model in an understandable and transparent way, in order to justify predictions. There are numerous methods for explaining the machine learning models, which can be classified according to three main criteria: i) depending on when the explanation takes place: it may occur directly during the training phase for those methods that are interpretable by their nature (intrinsic methods), or the explanation may take place when the model has already been trained (post-hoc methods), ii) depending on the sphere of the data it explains: the explanation of the model's decision can be given on a particular instance (local methods) or it can be given over the entire data distribution (global methods), iii) according to the generalization of the tools involved: the interpretation tools of some methods are limited to specific classes of models (model-specific methods), while for others, the tools can be used on any learning model (model-agnostic methods). The survey [13] classifies these different methods, emphasizing those that are used in cybersecurity and which have been used in the literature. Among them, SHAP is a well-used method in the context of anomaly detection ([12], [14], [15]) due to its robustness to interactions between features, while it makes possible to identify the most important characteristics that pushed the model to make a decision.

III. Explainable Anomaly Detection

In this section, we describe the autoencoder-based model for anomaly detection in ICS and the explainability module to interpret and better understand the model's predictions. In our work, we call an instance a message from our dataset.

A. Anomaly Detection based on Autoencoder

An autoencoder is a type of artificial neural network model used for unsupervd learning of efficient data representations that takes input data and transforms it into a reduced-dimension representation, and attempts to reconstruct it into a version close to the original data while minimizing the difference between the input data and the reconstructed output data. The goal of an autoencoder is to learn a compact and efficient representation of data by compressing (through the encoder) and reconstructing the inputs (through the decoder). We work with the undercomplete autoencoder (the latent space has a smaller dimension than the input data space), which forces the model to learn a compressed representation of the data while capturing only the most important features present in the data. The reconstructed output is then compared to the original input to evaluate the reconstruction error.

Since our aim is to detect anomalies, we train our model on only normal data, as we can easily translate the normal behavior of our system. Over time, the model learns to recognize what is normal and will therefore be able to identify any abnormal instances. Let us consider $N = \{X_1, X_2, \ldots, X_n\}$ the set of normal instances of our training dataset. For each $X_i \in N, X_i = \{x_1, x_2, \ldots, x_m\}$, where $x_j \in \mathbb{R}^m, j = 1, 2, \ldots, m$. n represents the number of instances in the training dataset and m the number of features. As shown in Fig 1, the autoencoder is trained on N. Let us denote $T = \{X'_1, X'_2, \ldots, X'_p\}$ the set of instances of our test dataset. Each $X'_i \in T$ has also m, features because the datasets were both generated by the same system.

The test data is then passed to the trained model, and for each $X'_i \in T$, the autoencoder produces an output $\hat{X}'_i$, which is the reconstructed data. We compute the difference and, in

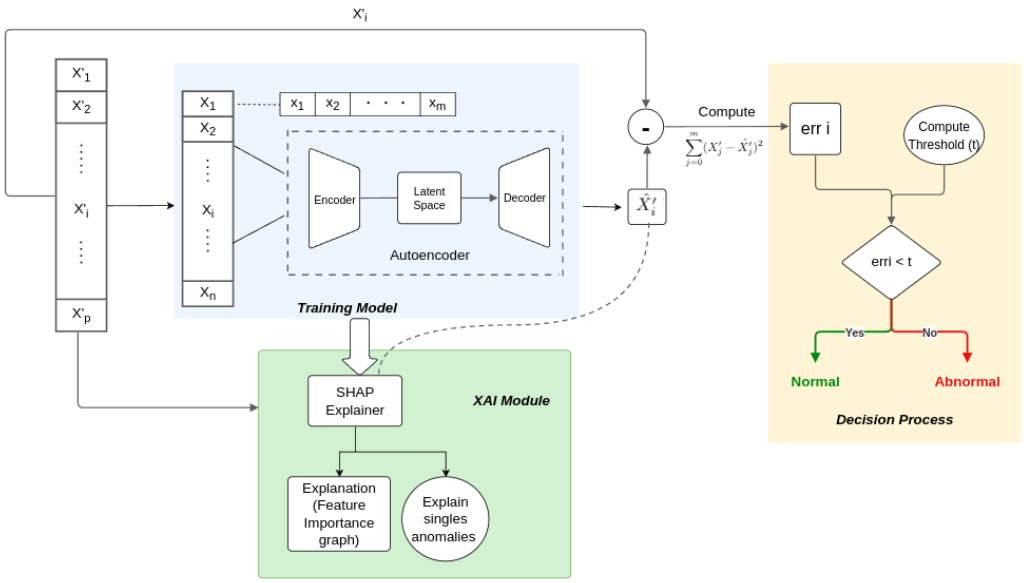

Fig. 1. XAI Detection Module

order to have the reconstruction error for each instance, we compute the Mean Square Error between X'_i and $\hat{X}'_i$.

$$err_i = \frac{1}{m}\sum_{j=0}^{m}(X'_j - \hat{X}'_j)^2 \tag{1}$$

The final step is to determine whether the instance X'_i is normal or not. To do this, we compare the calculated error (err_i) with a defined threshold: if the reconstruction error is smaller than the defined threshold, the model predicts the instance as normal, otherwise, it is abnormal. We can therefore remark that the definition of the threshold is very critical and greatly influences the performance of our model. To define an optimal threshold, we can use reconstruction score statistics. Indeed, each instance is reconstructed with an error (which is used to assign a reconstruction score to each instance). Together, these scores form a distribution that can be used to determine the optimal threshold. A well-known approach is to use descriptive statistics (mean, standard deviation) to categorize the distribution of scores. The threshold is then selected on the basis of the number of standard deviations above the mean. We use this method to determine the optimal threshold and we consider 2 standard deviations above the mean.

B. Explainable Artificial Intelligence Module

Recurrent neural networks are often regarded as black boxes, due to the opacity of the algorithms used for the various models. Hence, there is a need to interpret machine learning models to determine when and why the model is wrong, validate model consistency, limit the errors and improve performance. In the literature, there is a considerable variety of explainability methods. One such method is SHapley Additive exPlanations (SHAP), which was introduced in 2017 by Lundberg and Lee [16] and aims to provide explanations based on the contributions of each feature to the prediction of a model. It is a post-hoc method that is applied after the model training phase. SHAP is a model-agnostic method based on Shapley values, which are in turn based on game theory and represent the marginal contribution of each feature to model prediction. If we denote by g the explainability model, X' the vector representing the input values of the features, m the number of features and X'_j the value of feature j in instance X', the following formula allows us to describe the method:

$$g(X') = \phi_0 + \sum_{j=1}^{m}\phi_j X'_j$$

where ϕ_j, the SHAP value assigned to feature j is calculated as the sum of prediction differences between feature sets S including and excluding feature j, weighted by the number of possible permutations of these sets.

$$\phi_j = \sum_{S\subseteq\{1,2,\ldots,m\}\setminus\{j\}} \frac{|S|!(m-|S|-1)!}{m!}[g(S\cup\{j\}) - g(S)]$$

where S is a subset of features without the $j-th$ feature, $\mid S \mid$, the number of features in the subset, $g(S)$, the model prediction when the features in S are fixed at their respective values but excluding the $j-th$ feature.

In our context, since we are working with autoencoders, we are going to use Shapley values to evaluate the contribution of each variable to the reconstruction of the input data and not to the prediction directly, since the model makes a decision based on the reconstruction error, which is strongly linked to the reconstructed data.

SHAP[1] has several types of Explainer, each using a specific

[1]https://shap-lrjball.readthedocs.io/en/latest/index.html

method to calculate SHAP values and being designed for specific data. We use SHAP's Explainer, which is designed for any machine learning model and also takes the least amount of time. In this work, we use it to visualize the overall contribution of our features to the model's predictions, and to explain the model's predictions, for the anomaly instances detected (XAI Module on Fig 1). For better interpretability of local instances, we will plot the SHAP graph showing the contribution of features to the reconstruction of the instance with respect to a specific characteristic. For example, if we consider an instance X'_i of the test set T, the reconstruction error (1) is calculated as the average of the squared difference with respect to all features. In our case, instead of evaluating the SHAP graph of the instance in relation to this average (and therefore to all features), we will consider the feature with the most significant value in the squared difference $(X'_i - \hat{X}'_i)^2$.

IV. Test use case

We present now the application of the methodology described and the training of the model on a simple use case.

A. System description

We consider a remotely controlled valve. A controller remotely opens or closes a valve while periodically reading the state of the valve. The valve can be in only two states: open (1) or closed (0). The different actions of the automaton are **Write**(W) which controls the valve, **Read**(R) which reads the state of the valve and then returns the response through **Answer**(A). Fig 2 depicts the process thus described. We assume that the **W** does not have acknowledgements.

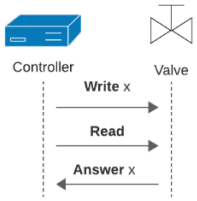

Fig. 2. Process Description

The normal behavior of the system follows the next rules:

1) After a write, the state changes accordingly.
2) The valve state may change only after a write.
3) A read request is always followed by an answer.
4) A write may occur only after the reading of a state (i.e. after receiving an answer message)
5) There are no "useless" writes (like sending an "open" command if the valve is open).

These different rules allow us to qualify all types of message sequences that are part of our system's normal behavior.

B. Dataset

Considering the predefined rules, we generate two datasets:

- A dataset containing only normal system behavior sequences of messages is used to train our model based on autoencoders (100,000 samples)
- A dataset with the normal system behavior sequences of messages but into which we inject some abnormal messages is used to evaluate our model (100,000 samples).

The dataset we are working on is made up of several messages (requests and responses). These include communication in our system. A message is classified as normal or abnormal according to the instance(s) that precede it. To ensure that we maintain the link between instances and enable our model to make decisions based on previous messages, we introduce the notion of **context** into our dataset. The context for a given instance will represent the state of the valve, the type and the content of the previous message. Table I shows an extract from the normal system behavior dataset containing 3 instances. An example of a normal message sequence from this extract is W 1 followed by R - and then by A 1 .

TABLE I
AN EXTRACT FROM THE DATASET WITH CONTEXT

Index	Action	Data	Last State	Last Action	Last Data
1	R	-	1	W	1
2	A	1	1	R	-
3	W	0	1	A	1

C. Attacks description

Our model classifies only messages, not the sequences of messages. An abnormal message is one in which the effect of the attack is visible. In Table II, the three rows represent a sequence of messages, and each row is an instance of the sequence. The model qualifies only the last row as abnormal, as it is this instance that highlights abnormal behavior occurring throughout the sequence.

TABLE II
EXAMPLES OF NORMAL AND ABNORMAL MESSAGES

1st Rule		2nd Rule		5th Rule	
Normal	Abnormal	Normal	Abnormal	Normal	Abnormal
W 1	W 1	A 1	A 1	R -	R -
R -	R -	R -	R -	A 1	A 1
A 1	A 0	A 1	A 0	W 0	W 1

Considering our behavior rules, we defined four anomalies that we want to detect (although more can be defined) : **i) Answer 0 on 1** (Answer 0 when the last state is 1), **ii) Answer 1 on 0** (Answer 1 when the last state is 0), **iii) Write 0 on 0** (Write 0 when the last state is 0) and **iv) Write 1 on 1** (Write 1 when the last state is 1). Note that these types of attacks correspond to security patterns of the form absence *never(Write S)while(state S)* respectively chain precedence *(write(S)precedes(Read,Answer(S))*.

We are interested in detection of weak signals, so we introduce a few abnormal messages (exactly 14, chosen arbitrarily). Then only 0.014% of lines in the test dataset represent anomalies (100,000 instances in total) and 99.986% normal behavior instances, which is representative for stealth process aware attacks. We are generating ourselves the attacks, then we insert the following abnormal messages: 6 instances of

type i), 4 instances of type ii), 2 instances of type iii) and 2 instances of type iv).

V. Results and Discussion

The trained model detects all 14 anomalies into the test dataset. However, for our study, we are interested to know how the model has learn the security patterns. Therefore, we first identify the most important features of the model that contributed to its overall predictions. We recall that the outputs of our model are the reconstructed data: we therefore have a multi-dimensional output corresponding to the number of features (in our case 9 and not just 5 as in Table I due to the categorical encoding of the variables "action" and "last action"). The variables that had the greatest impact on our predictions are "data", "last state", and "last data" with "data" clearly dominating the two others.

TABLE III
Most important features contributions in reconstruction error of abnormal instances

Attack Type	Data	Last State	Last Data
Answer 0 on 1	8.43×10^{-5}	6.57×10^{-12}	0.00
Answer 1 on 0	2.50×10^{-1}	1.55×10^{-13}	3.55×10^{-15}
Write 0 on 0	8.84×10^{-5}	8.57×10^{-12}	9.46×10^{-11}
Write 1 on 1	2.50×10^{-1}	4.30×10^{-13}	8.48×10^{-2}

A first important remark while analyzing Table I is that, into the same class of anomalies (i.e. wrong Answer or redundant Write) if message data is 1 the anomaly will present a higher reconstruction error and is more easily detected. Indeed, if one inspects the model predictions for the four anomalies, one can find that predicted "data" values are small (of order 10^{-2} when message data is 1 and 10^{-4} when message data is 0). This is likely due to the "data" encoding (0 for "closed", 0.5 for "open" and 1 to fill in the absence of data in Read requests).

We can justify the importance of the feature "last state" in the fact that the model is based on the previous state of the valve to identify an anomaly; the feature "last data", which represents the last written value, also plays a significant role because it is strongly linked to the state of the valve, and the impact of the feature "data" is clear because its content intrinsically gives the state of the valve.

While examining the values in Table III , data is the most significant feature. We therefore show in figures 3, 4, 5 and 6 the SHAP contribution of features for an instance of type i), ii), iii) and iv) respectively, in relation to the feature "data". For the attack of type Write 1 on 1, the contribution of "last data" in the error is also significant, and therefore, for an instance of this type, we also draw a graph representing the impact of the features in relation to "last data" (Fig 7). The blue bars represent a negative contribution to data reconstruction (the feature contributed to the detection of the anomaly), and the red bars a positive contribution (the feature had an opposite effect and instead pushed the model towards a characterization of the instance as normal). The length of the bars is proportional to the impact (SHAP values), but for representational purposes, the SHAP values shown next to them are approximations of the exact values.

For all four types of attacks, we obtain a positive contribution of data to anomaly detection. This is due not only to the inconsistency with the previous state observed, but also to the action taken: for Answer instances, it is normal to answer 0 when the last state is 0, which is considered an anomaly for Write instances. We also note that in all 4 cases, Action_R has a negative contribution (and therefore a positive impact in predicting the anomaly). In fact, the action R is the only action whose data value does not vary and which cannot take on the values corresponding to the valve's open and closed states. Because of this, the model gives it a high weight in computing the reconstruction error of feature "data". While considering Answer and Write messages, the contribution of Action_R may be safely ignored. A similar reasoning may be applied to the other contributing feature related to actions whose values are 0 (Action_W, Last_Action_A, etc.).

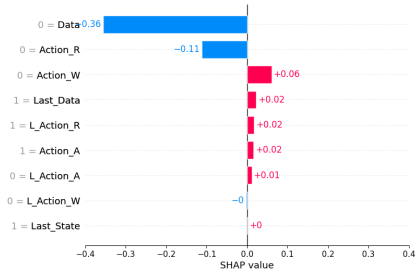

Fig. 3. Explaining an instance of Answer 0 on 1

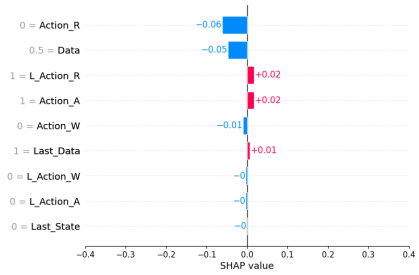

Fig. 4. Explaining an instance of Answer 1 on 0

In both cases of instances of type Answer, we have Action_A with the same contribution to a normal prediction, which is correct because this variable simply describe the action taken. Last_Action_R has also a contribution in the same direction because it enables the model to verify the rule 3 (part of normal system behavior).

For Write instances, we note a positive contribution towards prediction for the Action_W variable, which describes the action taken. Comparing figures 5 and 6, we can see that the contribution of the last data changes direction, and this may be due to the way in which the values are encoded in the data: although both are anomalies, the model tends to give more weight to one value than the other (Fig 7). And so, in relation

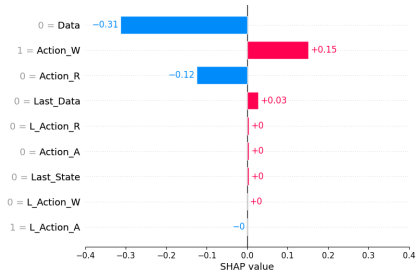

Fig. 5. Explaining an instance of Write 0 on 0

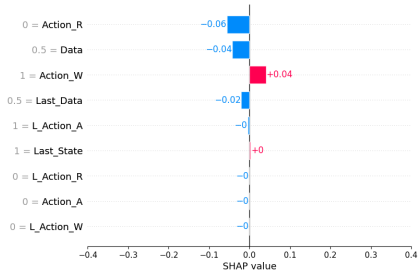

Fig. 6. Explaining an instance of Write 1 on 1

to the specific context under study, it is necessary to choose a data encoding that corresponds to the model's expectations.

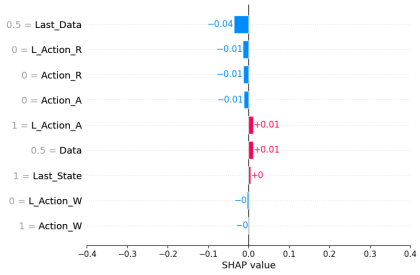

Fig. 7. Explaining an instance of Write 1 on 1 (Last Data)

VI. Conclusion and Perspectives

In this first approach to detect anomalies that violate security pattern (process-aware attacks) we tested an unsupervised autoencoder model trained on a normal behavior dataset against two types of attack patterns, a simple one (write) and a complex one (violation of a chain precedence pattern).

On the positive side, the two types of attacks were correctly detected and no false positives were registered. Moreover, the model correctly identifies that the anomaly is related to the "data" field of the requests. Shapley values analysis allowed us to clearly identify the contribution of the various features to the computing of the reconstruction loss of the "data" feature. For the moment it is not possible to deduce the security pattern from the AI explanation as our model is unsupervised.

A less positive point is that the quality of the detection seems to be dependent on the encoding, and this point will be more deeply investigated in the future.

Future research will be continued on two main directions: extend the study to other classification models, consider more types of attack patterns and test the trained models against them into the explainable framework.

References

[1] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A.-R. Sadeghi, M. Maniatakos, and R. Karri, "The cybersecurity landscape in industrial control systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, 2016.

[2] O. Koucham, S. Mocanu, G. Hiet, J.-M. Thiriet, and F. Majorczyk, "Detecting Process-Aware Attacks in Sequential Control Systems," in *NordSec 2016 - 21st Nordic Conference on Secure IT Systems (NordSec 2016)*, Oulu, Finland, Nov. 2016, pp. p.20–36. [Online]. Available: https://inria.hal.science/hal-01361081

[3] MITRE — ATT&CK® for Industrial Control Systems, 2021, [Online] https://attack.mitre.org/matrices/ics/, last accessed Apr. 2024.

[4] R. R. R. Barbosa, R. Sadre, and A. Pras, "Exploiting Traffic Periodicity in Industrial Control Networks," *International journal of critical infrastructure protection*, vol. 13, pp. 52–62, 2016.

[5] A. Carcano, I. N. Fovino, M. Masera, and A. Trombetta, "State-based network intrusion detection systems for SCADA protocols: A proof of concept," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6027 LNCS, pp. 138–150, 2010.

[6] E. Hotellier, F. Sicard, J. Francq, and S. Mocanu, "Standard Specification-based Intrusion Detection for Hierarchical Industrial Control Systems," *Information Sciences*, 2024, article 120102.

[7] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21st international conference on Software engineering*, 1999, pp. 411–420.

[8] S. Konrad and B. Cheng, "Real-time specification patterns," in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 372–381.

[9] O. Maler and D. Ničković, "Monitoring Temporal Properties of Continuous Signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[10] O. Koucham, S. Mocanu, G. Hiet, J.-M. Thiriet, and F. Majorczyk, "Efficient Mining of Temporal Safety Properties for Intrusion Detection in Industrial Control Systems," in *SAFEPROCESS*, 2018, pp. 1–8.

[11] R. C. Borges Hink, J. M. Beaver, M. A. Buckner, T. Morris, U. Adhikari, and S. Pan, "Machine learning for power system disturbance and cyber-attack discrimination," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*, 2014, pp. 1–8.

[12] D. T. Ha, N. X. Hoang, N. V. Hoang, N. H. Du, T. T. Huong, and K. P. Tran, "Explainable anomaly detection for industrial control system cybersecurity," *IFAC-PapersOnLine*, vol. 55, no. 10, pp. 1183–1188, 2022, 10th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322018572

[13] F. Charmet, H. C. Tanuwidjaja, S. Ayoubi, P.-F. Gimenez, Y. Han, H. Jmila, G. Blanc, T. Takahashi, and Z. Zhang, "Explainable artificial intelligence for cybersecurity: a literature survey," *Annals of Telecommunications - annales des télécommunications*, vol. 77, no. 11-12, pp. 789–812, Dec. 2022. [Online]. Available: https://hal.science/hal-03965590

[14] K. Roshan and A. Zafar, "Utilizing xai technique to improve autoencoder based model for computer network anomaly detection with shapley additive explanation(shap)," *International journal of Computer Networks & Communications*, vol. 13, no. 6, p. 109–128, Sep. 2021. [Online]. Available: http://dx.doi.org/10.5121/ijcnc.2021.13607

[15] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using shapley additive explanations," *Expert Systems with Applications*, vol. 186, p. 115736, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421011155

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4768–4777.