Proposing Guidelines and Approaches to Make Anomaly Detection More Effective for Industrial Control Systems

Clement Fung

June 20, 2024

Software and Societal Systems Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Prof. Lujo Bauer, Carnegie Mellon University (Chair)
Prof. Eunsuk Kang, Carnegie Mellon University
Prof. Vyas Sekar, Carnegie Mellon University
Prof. Michael Reiter, Duke University

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Abstract

Industrial control systems (ICS) govern critical infrastructure and processes, such as power generation, chemical processing, and water treatment. Given their widespread impact and their critical nature, there is a strong incentive for adversaries to attack ICS. An adversary that gains access to an ICS network can manipulate its process values to cause physical damage and harm. Machine-learning-based anomaly detection can be used to detect such manipulated data and is a common proposal for defending ICS. To make anomaly detection more effective for ICS, this thesis investigates and proposes solutions to several challenges when applying anomaly detection to an ICS. First, it is unclear what models and methods are best for detecting ICS anomalies; we comprehensively evaluate prior approaches and compare their performance, identifying what strategies were most effective. Second, it is unclear if and how anomaly-detection outputs can be used to diagnose ICS anomalies; we evaluate a variety of approaches for attributing ICS anomalies to the underlying components that were manipulated. Third, we identify fundamental issues with prior anomaly-detection approaches for ICS, and we are investigating how incorporating domain knowledge through graphs can improve current detection and attribution approaches. Finally, to better understand if current anomaly-detection approaches appropriately match the needs of ICS in practice, we are conducting an interview-based study to understand the workflows and perspectives of practitioners that monitor ICS.

Thesis Statement: To make anomaly detection more effective for industrial control systems (ICS), we design approaches for detecting and attributing ICS anomalies and propose guidelines for choosing and configuring anomaly-detection models. We design these approaches and guidelines to address needs across the ICS anomaly-detection workflow: (i) when detecting anomalies; (ii) when identifying the root cause of anomalies; and (iii) when deploying, using, and maintaining anomaly-detection systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Industrial control systems (ICS) govern critical infrastructure, such as water treatment, power generation, and chemical processing. Because of their criticality and wide-ranging impact, ICS are common targets for attacks [63]. An attacker that gains access to the network of an ICS and manipulates a subset of its process values can cause the ICS to become unstable, causing damage and harm [15, 51].

To prevent such attacks from causing attacker-intended damage, researchers have proposed anomaly-detection-based approaches that can be used to detect when ICS process values have been manipulated [10, 16, 32, 42, 57, 71, 73, 77]. By first training an anomaly-detection model to recognize or reconstruct the process values expected during ICS operation, the model can then be used to detect when an ICS deviates from its safe, expected operation. Sufficiently high deviations are used to indicate an anomaly. Anomaly-detection approaches for ICS, both based on statistical estimation [10, 32, 71, 73] and based on deep-learning models [42, 57, 77], have been widely studied [27, 54]. Although researchers report strong detection performance with such approaches, several challenges prevent their widespread effectiveness and adoption [7].

To improve the effectiveness and adoption of anomaly detection for ICS, I design approaches for detecting and explaining anomalies and propose guidelines for choosing, configuring, and deploying anomaly-detection models. These approaches and guidelines are used across the anomaly-detection workflow:

- **When detecting anomalies**: evaluating which model architectures work best for detection; identifying the best techniques for training and evaluating models; and tuning, designing, and developing models based on domain-specific needs.
- **When identifying the sources of anomalies**: designing and developing approaches to attribute detected anomalies to manipulated ICS components; and analyzing when and why certain approaches are more effective than others based on ICS attack and defense configurations.
- **When end-users and organizations interact with anomaly-detection systems**: examining how ICS are monitored in practice; and recommending guidelines to use, maintain, and deploy anomaly detection at organizations that work with ICS.

This thesis is comprised of the following chapters:

- In Chap. 2, I cover the background and related work for this thesis, composing multiple

areas of research: ICS security, anomaly-detection models, attribution methods for deep-learning-based models, and studies on challenges faced by security-relevant practitioners in contexts that partially overlap with the ICS-anomaly-detection context.

- In Chap. 3, I describe a study that determines what models and techniques are most effective for detecting ICS anomalies. In this work, my collaborators and I perform a common evaluation of previously proposed ICS anomaly-detection approaches: comparing deep-learning-based models, training and data processing techniques, and metrics used for ICS anomaly detection. In contrast to what is suggested in prior work, we find that the choice of model hyperparameters for deep-learning-based models play a small role in determining which approaches work best. We instead find that data processing, training methods, and the choice of metrics have a much larger impact on anomaly-detection effectiveness. This work is published in the *27th European Symposium on Research in Computer Security* (ESORICS 2022) [24].
- In Chap. 4, I describe a study that investigates if and how attributions can be used to find the components that were attacked when anomalies are detected, potentially aiding operators when responding to alarms. In this work, my collaborators and I evaluate how accurate attribution methods are at identifying which features were manipulated in an attack on ICS. We find that prior, off-the-shelf attribution methods are ineffective for ICS anomaly detection; furthermore, factors such as the timing of attribution input and the type of feature that was manipulated affect how well attribution methods perform and which methods were best. We ultimately propose an approach that uses an ensemble of attribution methods, and we show that it performs best. This work is published in the *31st Network and Distributed System Security Symposium* (NDSS 2024) [25].
- In Chap. 5, I describe a study that makes anomaly detection and attribution more effective for ICS by augmenting anomaly-detection models with domain-specific knowledge. In this work, my collaborators and I investigate how spatial and logical information from an ICS can augment prior anomaly-detection approaches. We represent this spatial and logical information as graphs, and we use these graphs to guide anomaly-detection-model training and prediction. I describe the initial design of our approach and propose a set of experiments to evaluate which approaches most effectively improve ICS anomaly detection and attribution. This work is in progress and will be completed for my thesis.
- Finally, in Chap. 6, I describe a study that makes ICS anomaly detection more useful in practice, by proposing guidelines based on the challenges and experiences of ICS practitioners when monitoring ICS and detecting anomalies. In this work, my collaborators and I are conducting semi-structured interviews with ICS practitioners, asking them (i) how they diagnose alerts, (ii) how they view the tools they use for detecting anomalous behavior, and (iii) how they view the potential use of machine-learning-based detection and explanation approaches for ICS anomalies. We will connect the findings from these interviews to closely related studies from prior work to identify distinct recommendations for anomaly-detection approaches designed for ICS. This work is in progress and will be completed for my thesis.

Chapter 2

Background and Related Work

In this chapter, I cover the relevant background and related work for the research areas encompassed by this thesis proposal. I first describe ICS (Sec. 2.1), attacks on ICS (Sec. 2.1), and public ICS datasets used for evaluation in prior work (Sec. 2.2). I then describe models that are used to detect ICS anomalies (Sec. 2.3) and methods that attribute model predictions to their input features (Sec. 2.4). Finally, I describe studies of practitioners that work in contexts that partially overlap with the ICS anomaly-detection context (Sec. 2.5).

2.1 Attacks on industrial control systems (ICS)

ICS monitor and control physical, safety-critical processes. ICS are interconnected systems that are separated into layers of access and function in the hierarchical Purdue model of ICS [37]. Figure 2.1 shows an example of the layers in an ICS. The Purdue model divides an ICS into levels: from the physical process (Level 0, the strictest level of access) to higher-level applications

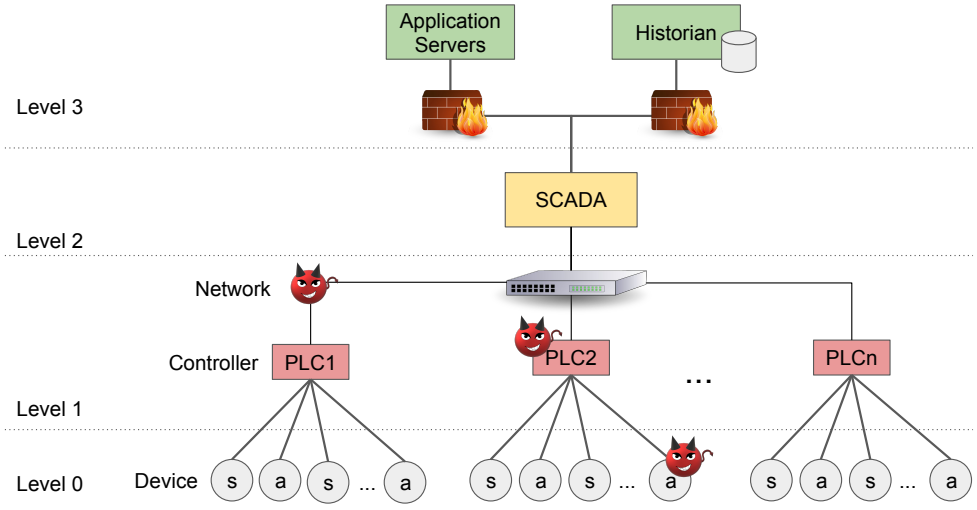


Figure 2.1: An overview of a typical, layered ICS architecture with examples of compromised endpoints and communication channels.

(Level 3, less strict). Sensors and actuators (Level 0) provide feedback from and input to the physical process. Programmable logic controllers (PLCs, Level 1) directly interface with sensors and actuators to control the ICS process. Supervisory control and data acquisition (SCADA, Level 2) governs multiple PLCs by collecting process data and providing an interface for operators to control and analyze the physical process [66]. With the exposure of ICS environments to the Internet and third parties [63], the potential of compromise has increased significantly. If an attacker compromises parts of an ICS in Levels 0–2, they can manipulate the data being sent over the network to cause process degradation or failure [51]. For example, the BlackEnergy (2015) and Industroyer (2016) attacks on the Ukrainian power grid [44] caused over 200,000 people to lose electric power for several hours; and the Triton malware attack (2017) [17] caused a chemical processing plant to maliciously shut down. To prevent these potentially harmful outcomes, it is critical to monitor ICS networks for signs of potential compromise and manipulation.

In this thesis, I focus on multiple aspects of ICS anomaly detection, spanning from the effectiveness of anomaly-detection models (Chap. 3, Chap. 5), techniques that help identify which feature was manipulated (Chap. 4, Chap. 5), and how users would interact with and use anomaly-detection systems (Chap. 6).

2.2 Datasets for ICS anomaly detection

The works presented in this thesis use a variety of data sources, ranging from public datasets to simulation environments.

Public datasets We use three public datasets for evaluation: BATADAL[1] [69], SWaT[2] [29], and WADI[3] [6]. All three datasets are provided by the Singapore University of Technology and Design iTrust Lab[4]. Each dataset contains one or more multi-day executions of an ICS, including benign executions (i.e., data from normal operation) and attacked executions (i.e., data from operations where pre-defined manipulations are performed). The attacked executions for each dataset include documentation that reports the start and end time of each attack, the component(s) that were manipulated, and the values used in each manipulation.

Simulation We use the Tennessee Eastman Process (TEP) [13, 18] as a source of simulated data. TEP is a simulation of a chemical process written in C and MATLAB. We execute TEP and record its time-series process values, using this data to train and evaluate anomaly-detection models. To generate attack data for TEP, we develop a module that executes pre-defined process-value manipulations during TEP's execution. We configure and perform several simulations to collect a variety of attack data. Our modified simulator is publicly available[5].

[1] "Battle of Attack Detection Algorithms" dataset
[2] "Secure Water Treatment" dataset
[3] "Water Distribution" dataset
[4] `https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/`
[5] `https://github.com/pwwl/tep-attack-simulator`

2.3 Models for process-level ICS anomaly detection

A variety of prior work has designed and applied models for ICS anomaly detection. These models are *unsupervised*; they are trained with only benign data that does not contain any anomalies. In contrast, *supervised* learning requires explicit data labels (attack or benign). As ICS attack data is rare and difficult to generalize, unsupervised learning is preferred.

Instead of anomaly-detection approaches that use network traffic [56] or system host information [33], this thesis focuses on process-level anomaly detection: models are trained with and predict ICS process values, such as sensor readings and actuator commands. We define the process values at a given time t as X_t. Given a d-by-h sequence of the previous h process values $(X_{t-h}, ..., X_{t-1})$ as input, the model predicts either an anomaly score a_t that indicates how anomalous this input is [10] or the next expected set of process values X'_t, which are then compared with the next true ICS values X_t to compute a mean squared error (MSE) [42, 77]. An anomaly is then declared when the anomaly score or MSE exceeds a predefined threshold. Such anomaly-detection models span statistical models [10, 32], models based on deep learning [42, 46, 77], and graph-based models [16].

Statistical anomaly-detection models Two statistical anomaly-detection models for ICS are AR [32, 71] and PASAD [10]. AR (auto-regressive modelling) is an approach that trains a reconstruction-based linear model for each feature; AR predicts each feature value from a linear combination of its historical values [32]. Prior work has extended AR by using a cumulative sum of prediction errors as a detection threshold [71]. PASAD trains an embedding into a lower-dimensional subspace for each feature; to detect anomalies, inputs are projected onto this subspace and the distance from the subspace centroid is used as an anomaly score. A threshold is then set on the anomaly score using validation data, and inputs that exceed this threshold are declared as anomalous.

Deep-learning-based anomaly-detection models A variety of deep-learning-based architectures are also used for ICS anomaly detection: these include autoencoders (AE) [68], convolutional neural networks (CNN) [42], recurrent neural networks (RNN) [46], and long-short-term memory networks (LSTM) [77]. AEs are trained to reconstruct inputs through a low-dimensional representation, supporting high quality reconstruction of states similar to those observed during training, and low quality reconstruction of inputs dissimilar to those observed during training. CNNs are commonly used for image-based tasks [47] but they can also be applied over sequences in time with one-dimensional convolutional kernels [42]. RNNs and LSTMs are similar to CNNs but do not required fixed-time-length convolutional kernels; RNN units are trained over sequences with the ability to update or reset parameters based on time sequences, while LSTM units further include the ability to maintain parameter values over time.

Graph-based anomaly-detection models Graphs can be used for ICS anomaly detection by training and making predictions with graph neural networks (GNN) [16]. GNNs perform convolutions defined by a graph structure, and therefore require a graph representation of the relationships between its input features. In the case of ICS anomaly detection which uses sensors and

actuator values as features, nodes represent sensors/actuators and edges between nodes represent causal relationships between a pair of sensors/actuators. Prior work has shown that the graph structure can also be dynamically learned as a parameter while training GNNs for ICS anomaly detection [16].

In this thesis, I evaluate how effective these models are for detecting ICS anomalies (in Chap. 3) and propose improvements to these models for the ICS anomaly detection use case (in Chap. 5).

2.4 Attribution methods for machine-learning models

Attribution methods compute the influence of a machine-learning model's input features on their predictions [5]. In this thesis, we focus on *instance-based* attributions, which compute a distinct attribution for a single input example. Instance-based attributions depend on the input example's feature values, the weights of the model, and the chosen output of interest. Such attribution methods are further divided into two categories: *white-box* attribution methods and *black-box* attribution methods.

White-box methods White-box attribution methods use gradients computed over model parameters. Saliency maps (SM) are computed by directly computing the gradient of the output of interest (e.g., the probability of a specific class) with respect to the input features [60]. Other white-box attribution methods build on SM by introducing approaches that improve the robustness of the attribution: SmoothGrad (SG) perturbs inputs with random noise [64], integrated gradients (IG) compute gradients with respect to a reference baseline [67], and expected gradients (EG) compute gradients over an expectation of randomly sampled reference baselines [21].

Black-box methods Black-box attribution methods do not use model parameters. Instead, they approximate the behavior of the model around the provided input. This approximation is built from several model queries drawn from a neighborhood surrounding the input of interest [31, 53, 58]. Several black-box attribution methods exist and the primary differences amongst them come from how the model approximation is constructed. For example, LIME uses the coefficients of a linear regression [58], SHAP uses Shapley values from game theory [53], and LEMNA uses the coefficients of a fused Lasso model and a Guassian mixture model [31].

In this thesis, I investigate how attribution methods can be applied to and designed for ICS anomaly detection. I propose that attribution methods can be used to help identify which component in an ICS was manipulated by an adversary, potentially aiding practitioners when responding to anomalies. I compare existing models (in Chap. 4) and propose improvements to them (in Chap. 5) when attributing ICS anomalies.

2.5 Challenges faced by practitioners in contexts similar to ICS anomaly detection

A variety of prior work has studied the challenges that practitioners face when (i) securing ICS, (ii) using and deploying machine-learning-based solutions for computer security, and (iii) responding to security alarms in real-time.

Security of ICS Prior work has studied the security-related perspectives of professionals that work with the electric power grid [26, 61]. Although the electric power grid is a prevalent example of an ICS, these works do not focus on real-time anomaly detection and remediation, instead focusing on how practitioners perceive the severity of attacks and vulnerabilities. Other prior works have studied usability and maintenance challenges in industrial control [14, 62]. These works identify relevant practical and organizational challenges within ICS but are not focused on their implications on security.

Machine learning for security Mink et al. study practitioner's perspectives of machine learning for security-relevant tasks (e.g., finding software vulnerabilities), suggesting ways in which machine learning can be used more effectively in practice [55]. Mink et al. compare rule-based approaches and machine-learning-based approaches, suggesting that they differ in terms of false positives, false negatives, interpretability, and required domain expertise. Ultimately, they suggest that a hybrid solution that leverages rules and machine learning is likely required for most tasks to balance tradeoffs.

Real-time security alarms Security operations centers (SOCs) are organizational units that monitor organizations for malicious and potentially harmful activity in real-time [76]. A variety of prior work has studied SOCs and the challenges they face when responding to alerts [8, 39, 72]: common themes include operator burnout, a high volume of false alarms, difficulty interpreting alarms, and difficulties when maintaining tools.

Although these prior works cover various parts of the ICS-anomaly-detection workflow, none of them are focused on our specific context, in which anomaly-detection systems raise alarms when ICS are attacked in real-time. In this thesis, I explore the alert-diagnosis workflow and perspectives specifically for the ICS anomaly-detection context (in Chap. 6).

Chapter 3

Comparing models and techniques used for detecting ICS anomalies

Several approaches based on machine learning (ML) have been proposed for ICS anomaly detection, including those based on autoencoders [20, 68], convolutional neural networks [40, 42], and LSTMs [22, 77]. These approaches share many common datasets, yet make differing conclusions about which architectures are best. Thus, when practitioners are determining which anomaly-detection approaches to adopt, it is unclear which models and techniques should be used. In this chapter, we systematize prior work in ICS anomaly detection to examine why these discrepancies in findings exist, and determine the models and techniques that are most effective for ICS anomaly detection. This chapter describes a summary of this work; a longer description of this work is published in the *27th European Symposium on Research in Computer Security* (ESORICS 2022) [24].

3.1 Introduction

When using a reconstruction-based anomaly-detection approach, practitioners must: (1) select an ML model architecture (e.g., convolutional neural networks), (2) select hyperparameters for the model (e.g., the number of hidden layers in the model), (3) collect a sufficient volume of benign ICS process data, (4) train an ML model to predict expected process values, and (5) tune detection hyperparameters (the threshold for an anomaly) to turn process-value predictions into alarms. Design decisions made in each of these steps play a role in the final performance of the anomaly-detection model.

Despite the variety of work in ICS anomaly detection, there is no consensus on what solutions are best. Proposed approaches use different ML model architectures (e.g., autoencoders [20, 68], CNNs [40, 42], LSTMs [22, 77]), use different datasets [6, 29, 69], and employ different data pre-processing and training techniques. As a result, when one approach is reported to outperform another, it is not clear what is responsible for the improved performance. In this work, we perform a comprehensive, empirical evaluation of techniques across datasets commonly used in reconstruction-based ICS anomaly detection. Perhaps surprisingly, we find that the best performance can be achieved by most models, including models that are far smaller (i.e., contain

fewer parameters) than the previously reported best models. We also identify training and data pre-processing techniques that strongly affect the results of reconstruction-based ICS anomaly detection, but are not used consistently across prior work.

Another important design consideration is the choice of metric used to tune and evaluate anomaly-detection models [74]. Typically, prior work equally penalizes false positives and false negatives with the point-F1 score computed on a per-timestep basis [65]. However, since ICS attacks take place over a sequence of timesteps [6, 29] and timely detection of attacks is important [35], ICS anomaly-detection models should be evaluated over temporal ranges. Unlike point-F1, *range-based* metrics score detection performance on temporal ranges and can express tradeoffs between increased detection rates, reduced false-alarm rates, and lowered detection latency [35, 45, 70]. When used to tune ICS anomaly-detection models, range-based metrics produce models that perform better on metric-specific tradeoffs. We also show that using range-based metrics to evaluate anomaly-detection models gives a better understanding of what models are optimal.

3.2 Comparing prior ICS anomaly-detection approaches

We first systematize ICS-anomaly-detection approaches from prior work by analyzing (i) what model is used, (ii) what datasets are used, and (iii) what tuning and evaluation metrics are used. Table 3.1 shows the results of this systematization. After analyzing the approaches described in prior work, we also observe that prior work varies in their training and data-processing techniques. Four key techniques were identified as most significant: (i) selecting features, (ii) shuffling benign data, (iii) cleaning attack start and end times, and (iv) early stopping while training. These techniques are necessary to fairly compare anomaly-detection approaches, as they improve the quality and consistency of anomaly-detection results. Table 3.2 shows, for each prior work, which key techniques are used.

Although some prior works evaluate multiple ML model architectures [3, 42, 77], no work covers the full selection of model architectures, datasets, and pre-processing techniques, making it is unclear what approaches are optimal across all settings. We therefore establish a standardized evaluation of three model architectures (AE, CNN, LSTMs) across three datasets (BATADAL, SWaT, and WADI) using all four key techniques. We perform a comprehensive comparison of models proposed in prior work to determine which models are most effective for ICS anomaly detection. We vary model hyperparameters across all three models; for autoencoders, we vary the number of hidden layers in the encoder/decoder from 1 to 5 (by 1) and the compression factor from 1.5 to 4.0 (by 0.5). For CNNs, we vary the number of layers from 1 to 5 (by 1), and vary the number of units per layer from 4 to 256 (by a factor of 2). The kernel size is fixed at 3 and we use history lengths of 50, 100, or 200 timesteps. For LSTMs, we vary the number of layers from 1 to 4 (by 1), the number of units per layer from 4 to 128 (by a factor of 2), and use history lengths of 50 or 100 timesteps. For comparison, we also implement three best-performing models from prior work with their suggested model hyperparameters [22, 40, 68].

Figure 3.1 shows the results of our comparison. We find that larger models (CNNs and LSTMs) perform poorly on the BATADAL dataset. We attribute the poor performance to the

Table 3.1: ML model architectures, datasets, and metrics from prior ICS anomaly-detection work. Range-based metrics are shown in **bold**. (CM = confusion matrix; TPR/FPR = true/false positive rate; TNR = true negative rate; Coverage = percentage of detection overlap; Norm-TPR = normalized true positive rate.)

Model Details	Datasets			Tuning Metric	Evaluation Metric(s)	Source
	B	S	W			
AE: 3-layers	●	●	●	FPR	Precision, Recall Point-F1	[42]
AE: 4-layers		●		None	Precision, Recall Point-F1, **Numenta**	[59]
AE: 5-layers	●			Point-F1	Precision, Recall Point-F1	[68]
AE: 5-layers	●		●	None	Precision, Recall Accuracy, Point-F1	[20]
CNN: 8-layers, 32 filters		●		**Range-F1**	**Range-F1**	[40]
CNN: 8-layers, 32 filters	●	●	●	FPR	Precision, Recall Point-F1	[42]
LSTM: 2-layers, 256 units		●	●	None	TPR, **Norm-TPR** FPR, **Atk TP**	[22]
LSTM: 3-layers, 100 units		●		Point-F1	Precision, Recall Point-F1	[36]
LSTM: 3-layers, 100 units		●		None	**Atk TP, Atk FP**	[28]
LSTM: 4-layers, 64 units		●		None	**Atk TP, Atk FP**	[38]
LSTM: 4-layers, 512 units		●		None	CM, Point-F1, **Atk TP**	[57]
LSTM: 4-layers, 512 units		●		Point-F1	Point-F1	[77]
1-class SVM		●		Point-F1	Point-F1	[36]
DNN: 3-layer	●			None	CM, TPR, TNR	[4]
Custom wide and deep CNN		●	●	None	Precision, Recall Point-F1, **Atk TP**	[3]
GAN		●	●	Point-F1	Precision, Recall Point-F1	[48]
Bayesian Network		●		None	**Atk FP, Atk TP FP length, Coverage**	[50]

relatively small size of the BATADAL dataset (only ∼48,000 datapoints, compared to ∼500,000 in SWaT and ∼1,000,000 in WADI); in prior work, only one study trains a CNN or LSTM on BATADAL [42]. For the SWaT and WADI datasets, we find that almost all model hyperparameter settings provide similarly strong performance: a 1-layer, 4-unit CNN or LSTM produces a similar point-F1 score to CNNs and LSTMs with more layers and units, including the best-performing models from prior work.

Although prior work performs model hyperparameter searches and claims to find the optimal models for ICS anomaly detection, our results show that equivalent performance can be achieved over a range of ML model architectures and hyperparameters when using the point-F1 score.

Table 3.2: Identifying key pre-processing and model training techniques from prior ICS anomaly-detection work. '●', '◐', and '○' indicate if the technique was used, partially used, or not used respectively. '**?**' indicates that we could not determine if the technique was used.

Feature Selection	Attack Cleaning	Benign Data Shuffling	Early Stopping	Source
○	○	?	○	[3]
○	○	●	○	[4]
○	○	?	●	[20]
○	○	○	○	[22]
◐	◐	?	●	[28]
○	●	○	●	[36]
○	●	○	○	[38]
○	◐	?	●	[41]
●	○	?	○	[42]
◐	○	○	○	[48]
○	●	○	○	[50]
●	●	○	○	[57]
○	○	○	○	[59]
○	○	?	●	[68]
●	◐	?	○	[77]

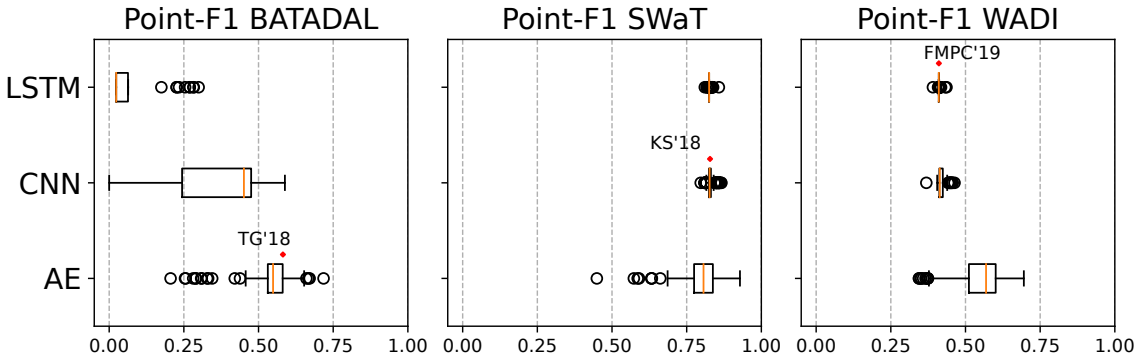


Figure 3.1: The final point-F1 scores of each model are shown when trained and tuned on three ICS datasets. For each dataset, a model hyperparameter setting from prior work [22, 40, 68] is included for comparison. When using the point-F1, the performance of AEs vary greatly, and most LSTM and CNN configurations perform similarly.

3.3 Tuning and evaluating with range-based metrics

In this section, we evaluate if using range-based metrics for tuning and evaluating anomaly-detection models results in better outcomes for ICS anomaly detection. Table 3.1 shows that point-based metrics (e.g., point-F1) are commonly used in prior work for tuning and evaluating ICS anomaly detection models; these metrics consider each timestep as an individual input when scoring anomaly-detection models. However, attacks in this dataset are executed *over time*—a contiguous range of inputs are attacked in sequence. We find that point-based metrics commonly used in prior work do not capture objectives relevant to ICS; Fig. 3.2 shows two examples with

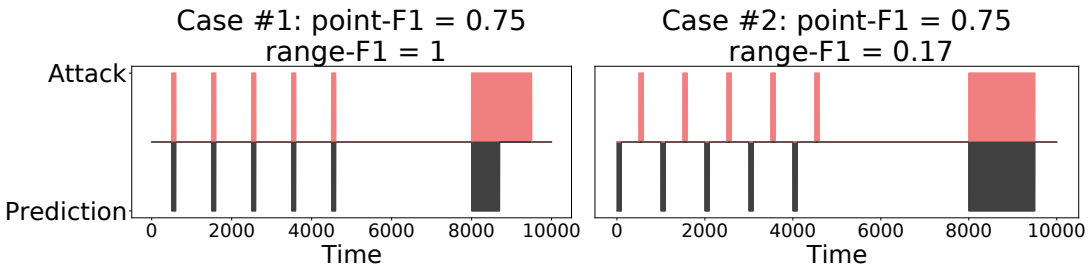


Figure 3.2: Two detection examples are shown: in each case, the x-axis represents time and the y-axis shows attacks (top, red) and attack predictions (bottom, grey). In the example on the left (case 1), all attacks are detected with no false positives, while in the example on the right (case 2) only one attack is detected, with five false positives; yet, the point-F1 scores are the same.

Table 3.3: For each optimal model proposed in prior work, we use a different tuning metric to select the optimal detection hyperparameters and show the resulting number of false alarms, detected attacks, and TP:FP ratio. Using range-F1 always outperforms its point-F1 counterpart in TP:FP ratio.

Dataset and Architecture	**Tuning Metric**	**False Alarms**	**Detected Attacks**	TP:FP **Ratio**
BATADAL AE	Point-F1	11	4/4	0.36
	Range-F1	1	4/4	4.00
WADI LSTM	Point-F1	143	10/13	0.07
	Range-F1	63	7/13	0.11
SWaT CNN	Point-F1	32	6/18	0.19
	Range-F1	4	4/18	1.00
	range-F$\beta_{3:1}$	0	3/18	∞
	range-F$\beta_{1:3}$	47	7/18	0.15
	NA-early	89	11/18 (7 early)	0.12

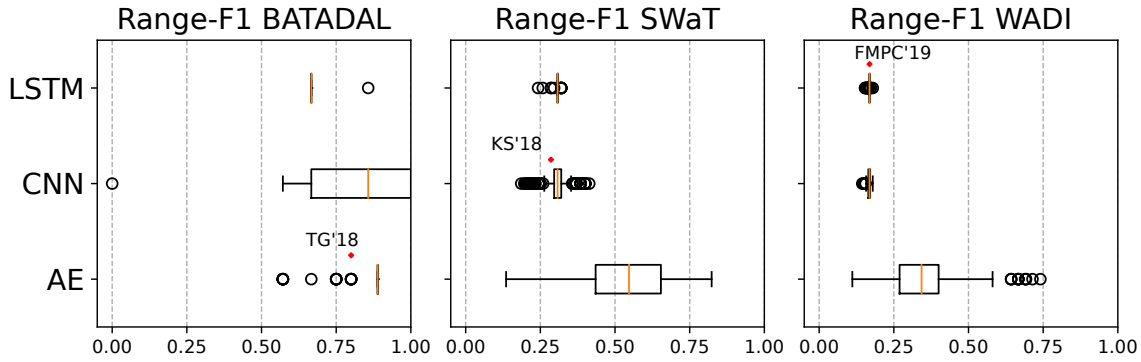


Figure 3.3: The final range-F1 scores of each model when trained and tuned on three ICS datasets. For each dataset, a model hyperparameter setting from prior work [22, 40, 68] is included for comparison. Compared to evaluating with the point-F1, evaluating models with the range-F1 provides a different view of which models are optimal.

the same resulting point-F1, but drastic differences in the number of attacks detected.

Prior work has proposed range-based metrics as a better alternative for tuning and evaluating

time-series anomaly detection [45, 70]. When scoring anomaly detection, these metrics capture objectives such as the precision and recall defined over attacks with the range-F1 [70], and the detection latency with the Numenta metric [45].

We explore the impact of range-based metrics on tuning by configuring the range-F1 and Numenta (NA) metrics from prior work for varying objectives: high precision, high recall, or early detection. Table 3.3 shows the result of this experiment: when metrics are defined for a particular objective, tuning a model with this metric achieves better performance on this metric. These results are achieved without any additional training; the baseline model is the same and only the detection threshold is tuned with the provided metric.

Finally, we report the performance of our models when the range-F1 is used to tune and evaluate ICS anomaly-detection models, as shown in Fig. 3.3. Compared to evaluating with the point-F1, evaluating with the range-F1 provides a different view of what models are optimal; for example, despite very low point-F1 scores (shown in Fig. 3.1), over 25% of CNNs produce a range-F1 of 1 on the BATADAL dataset, detecting all attacks without a single false alarm.

3.4 Summary

In this chapter, we determine what factors (e.g., the choice of model, training techniques, metrics, etc.) are most important when training ICS anomaly-detection models. Contrary to what is suggested by prior work, we find that the choice of model does not have a strong effect on anomaly-detection performance; the best performance can be achieved over a range of model architectures and hyperparameters. Instead, we used range-based metrics to optimize ICS anomaly detection and found that they lead to different and potentially more useful outcomes than the common approach of relying on the point-F1 score. Ultimately, we found that effective anomaly detection extends beyond optimizing for the point-F1, and better success measures are needed to practically tune and evaluate ICS anomaly-detection models.

Chapter 4

Evaluating attributions for ICS anomaly detection

A detected ICS anomaly requires follow-up diagnosis and remediation. In this chapter, we investigate if and how outputs from the anomaly-detection model can be used to assist with alarm remediation. We evaluate if attributions of anomalies detected by anomaly-detection models can be used to identify the manipulated component in an ICS attack. We first evaluate if prior approaches are sufficient for attribution, before performing a broader evaluation across anomaly-detection models, attribution methods, and ICS attack properties. We identify factors that affect attribution performance and make recommendations that make attributions more effective for ICS. This chapter describes a summary of this work; a longer description of this work is published in the *31st Network and Distributed System Security Symposium* (NDSS 2024) [25].

4.1 Introduction

In Chap. 3, we evaluate a variety of anomaly-detection models for ICS. Each of these models is reconstruction-based: they predict a set of ICS values and detect anomalies by comparing these predictions to their observed values. In this chapter, we investigate if outputs from anomaly-detection models can be used to attribute the causes of anomalies by identifying the component (i.e., the sensor or actuator) that was manipulated in an ICS attack.

We first investigate if, as suggested in prior work by evaluating on a few ICS attacks [34, 42], per-feature error ranking can be used for attribution. We expand on prior evaluations by evaluating with over 150 diverse attacks; this work is the first to systematically evaluate attributions for ICS anomalies. After finding that raw-error ranking fails to generalize across a more diverse set of attacks, we investigate if machine-learning-based *attribution methods* proposed in prior work [21, 31, 53, 58, 60, 64, 67] can be used to effectively attribute ICS anomalies. Attribution methods are predominately designed for and evaluated on image classification, so we must first adapt these methods for the ICS anomaly-detection domain.

Attacks in ICS datasets vary in how they are performed, such as the manipulation magnitude and the type of component that is attacked. We perform a statistical analysis of attribution accuracy across these factors to identify which attribution methods perform best for specific types

of attacks. Since we find that different methods perform best for different types of attacks, we ultimately design an ensemble of attribution methods and show that it outperforms all other individual attribution methods.

4.2 Methodology

In this section, we describe the methodology required to evaluate and compare anomaly attribution across various configurations. We first describe how we collected and categorized ICS attacks from public sources. We next describe our implementation of ML-based anomaly-detection models and attribution methods from prior work. Finally, we define AvgRank, an evaluation metric that is appropriate for evaluating attributions.

Collecting our manipulation dataset We evaluate against two groups of ICS anomalies: real attacks found in the SWAT and WADI datasets [6, 29] and synthetic anomalies created with the TEP simulator [13]. For SWAT and WADI, each attack is documented by the dataset provider: 67 manipulations (43 in SWaT, 24 in WADI) are included. To further increase anomaly diversity when evaluating attributions, we create an additional dataset of anomalies with the TEP simulator. We implement a MATLAB module that interfaces with the TEP simulation and manipulates process values, based on an attacker model used in prior work [15, 43]: benign values for a chosen feature are replaced with a chosen value over a chosen time period. Using the modified simulator, we systematically perform generate a set of *synthetic anomalies*; for each anomaly, we manipulate a sensor/actuator and record the resulting process values, creating 89 anomalies.[1] Although our synthetic anomalies do not necessarily correspond to intentional ICS attack outcomes, they are genuine statistical anomalies (95th percentile or higher values) executed with similar strategies as those used in the real attack dataset.

Categorizing manipulations used for ICS anomalies We define all anomalies along two dimensions. First, we define anomalies by the manipulation magnitude: the difference in standard deviations between a feature's benign distribution and the replaced value. Second, we define anomalies by the type of feature manipulated. We group feature types into three categories: *actuators*, which directly control the physical process; *sensors*, which are read by controllers to compute future actuator values; and *out-of-loop features*, which do not impact the physical process.

Implementing anomaly-detection models We evaluate attributions with three model architectures: convolutional neural networks (CNNs) [42], gated-recurrent-unit networks (GRUs) [23], and long-short-term-memory units (LSTMs) [57, 77]. A 2-layer, 64-unit, 50-length-history model is trained for each combination of dataset (SWaT, WADI, TEP) and architecture (CNN,

[1] 11 out of 100 attempted manipulations triggered a shutdown sequence, causing the MATLAB simulator to exit and preventing data collection.

GRU, LSTM), using the default Adam optimizer. We test each model against the manipulations from SWaT, WADI, and TEP, using each model's 99.95-th percentile validation MSE as a detection threshold, using the same strategy from prior work [24, 42, 77].

Implementing ML-based attribution methods We next describe how ML-based attribution methods are adapted for ICS anomaly detection. Each attribution method uses a trained anomaly-detection model and an input of interest to produce a score for each input feature.

SHAP [53], and LEMNA [31] are prominent, black-box attribution methods. These techniques use perturbed samples to train a local, linear approximation around an input, and use the approximation model's coefficients as attributions. We use the public SHAP [2] code, and we implement LEMNA based on its published description [31] and public code examples [1]. Each of the described implementations assumes a single-output classification or regression task. To adapt these implementations for ICS anomaly detection, we use a technique from prior work [9, 34]: for a given input, we identify the feature with the highest prediction error and compute its SHAP or LEMNA attributions. Thus, we adapt the anomaly-detection task as a single-output regression task to comply with the expected inputs for SHAP and LEMNA.

Saliency map (SM) attributions are computed from gradients on model weights [60]. Given a trained model, an input of interest, and a quantity of interest, the internal gradient of the quantity of interest with respect to the input is computed. The saliency map $A_{SM}(X)$ is the product of X_e and the gradient of the quantity of interest with respect to the input window, computed at X_e. We compute the gradient with respect to the MSE:

$$A_{SM}(X_e) = X_e \times \frac{\partial MSE(X_e)}{\partial X}$$

AvgRank: an evaluation metric for attributions To quantitatively compare attribution methods over full datasets, we propose the metric *AvgRank*. Across a set of ICS attacks, AvgRank represents the average number of features that need to be searched when using attributions to find the manipulated feature. For a given attribution s, we sort each feature's attribution s_j in descending order and identify the ranking of the manipulated feature j'. Since we use three datasets of varying dimension, we normalize rankings by dividing them by the number of features in their corresponding dataset. We then report the average across all anomalies to compute AvgRank: a score $\in [0, 1]$ (lower is better) that represents how accurate an attribution method is at identifying manipulated features. For example, an AvgRank of 0.2 indicates that an attribution method will, on average, rank the attacked feature in the top 20% of features.

4.3 Results

In this section, we summarize the results of our evaluation. First, we evaluate prior attribution strategies and determine that they perform worse for attribution than previously reported. To build a better understanding of attribution performance, we explore how factors such as the timing of detection and type of feature attacked affect attribution accuracy. After finding that different attribution methods perform best under varying conditions, we propose an ensemble of attribution methods and show that it outperforms all individual attribution methods in general.

Table 4.1: Top-k feature attribution accuracy and AvgRank for the baseline attribution strategy from prior work (ranking features by raw error at detection time). We find that this strategy performs worse than previously reported; for all methods, less than 40% of all attacks are identified by the highest score.

	Total	Top-1	Top-5	Top-10	AvgRank
CNNs	103	40 (39%)	59 (57%)	70 (68%)	0.187
GRUs	86	26 (30%)	54 (63%)	62 (72%)	0.141
LSTMs	113	27 (24%)	62 (55%)	75 (66%)	0.171

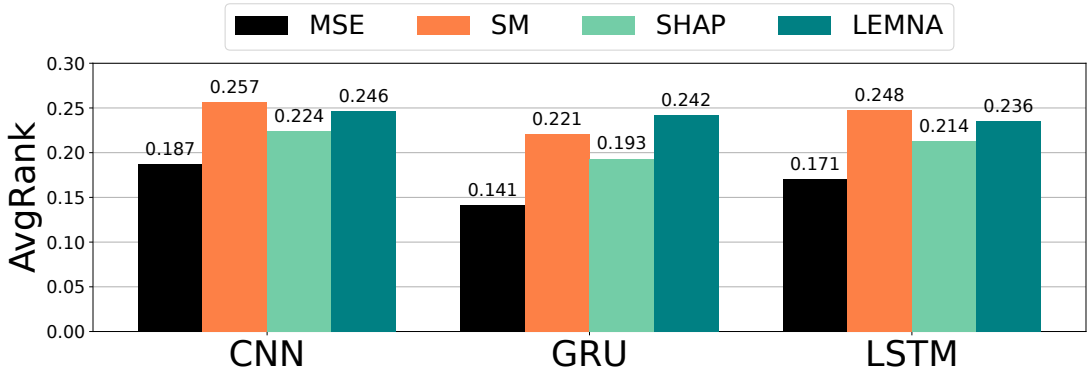


Figure 4.1: Across all detected anomalies, raw-error ranking (MSE) produces a lower (better) AvgRank than all best-performing ML-based attribution methods: the saliency map (SM), SHAP, and LEMNA.

4.3.1 Evaluating prior attribution strategies

We first use raw errors for attribution: we find the first correctly detected input within each attack and use its per-feature reconstruction errors as attribution. For each model, we evaluate over all detected attacks: 103 attacks for the CNN, 83 attacks for the GRU, and 113 attacks for the LSTM. Table 4.1 shows the accuracy when raw-error ranking is used for attribution. Although prior work has reported high attribution accuracies (e.g., 80% accuracy within the top few features [42]), we find that raw-error rankings are not as effective as reported; for all models, over 60% of attacks are not correctly identified by the highest-error feature, and over 25% of attacks are not identified within the top 10 features. Across models, the AvgRank ranges from 0.14 to 0.19 (i.e., at least 14–19% of features are searched on average before finding the attacked feature), which is far more than suggested in prior work.

Next, we compare attributions from raw-error ranking with attributions from ML-based attribution methods: the saliency map (SM), SHAP, and LEMNA, which were identified as the best-performing methods [25]. To select inputs for ML-based attribution methods, we find the first correctly detected timestep for each attack using the same methodology as with raw-error rankings. We then find the corresponding model input—the values from the 50 timesteps prior to this timestep—and use these values as the attribution method input for this attack. Fig. 4.1 shows the AvgRank for all attribution methods and for all models. ML-based attribution methods perform poorly; for each model, raw-error ranking (MSE) outperforms all ML-based attribution methods (0.14–0.19 for MSE, compared to over 0.19 for all ML-based attribution methods).

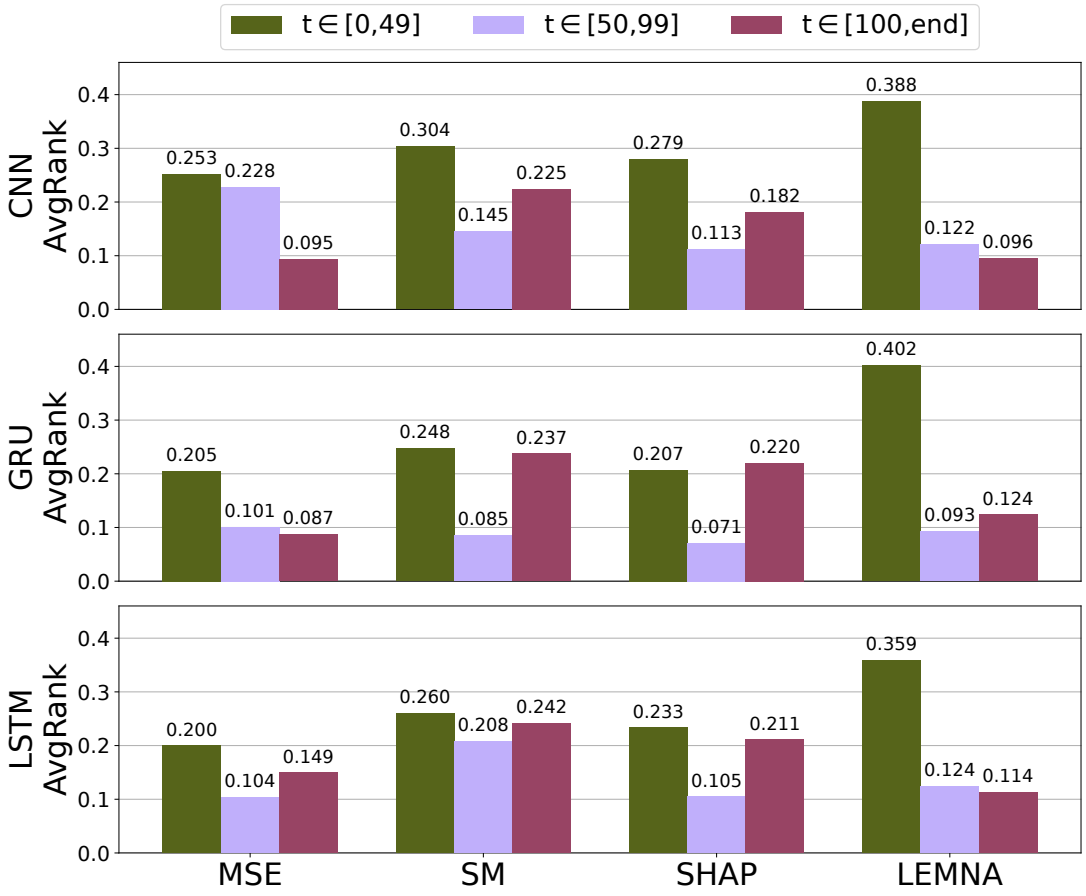


Figure 4.2: Across all detected anomalies, we compare AvgRank (lower is better) across three cases, based on the detection time t relative to the start of the anomaly ($t = 0$). Given that our models use 50 timesteps as input, we compare if $t \in [0, 49]$, $t \in [50, 99]$, or $t \in [100, \text{end}]$. For most cases, the AvgRank is lowest when $t \in [50, 99]$.

4.3.2 Analyzing factors that affect attribution accuracy

To better understand reasons for poor attribution performance, we analyze how (i) the timing of detection, (ii) manipulation magnitude, and (iii) the type of feature attacked affect attribution accuracy.

Detection timing Even when anomaly-detection models correctly detect an attack, the detection timestep can vary relative to the start of the anomaly. Given a model's input-window length (50 timesteps), the detection can occur before the window length has passed, far after multiple window lengths have passed, or at a time between these two cases—within one and two window lengths. We analyze AvgRank across these three cases and show the results in Fig. 4.2. In most settings, attributions computed within one window length (<50s) perform worst, and attributions computed within one and two window lengths (50–100s) perform best.

Based on these results, we select an input that starts at the same time as the anomaly (e.g., given a 50-timestep input window length, we use the first 50 timesteps of the anomaly as input

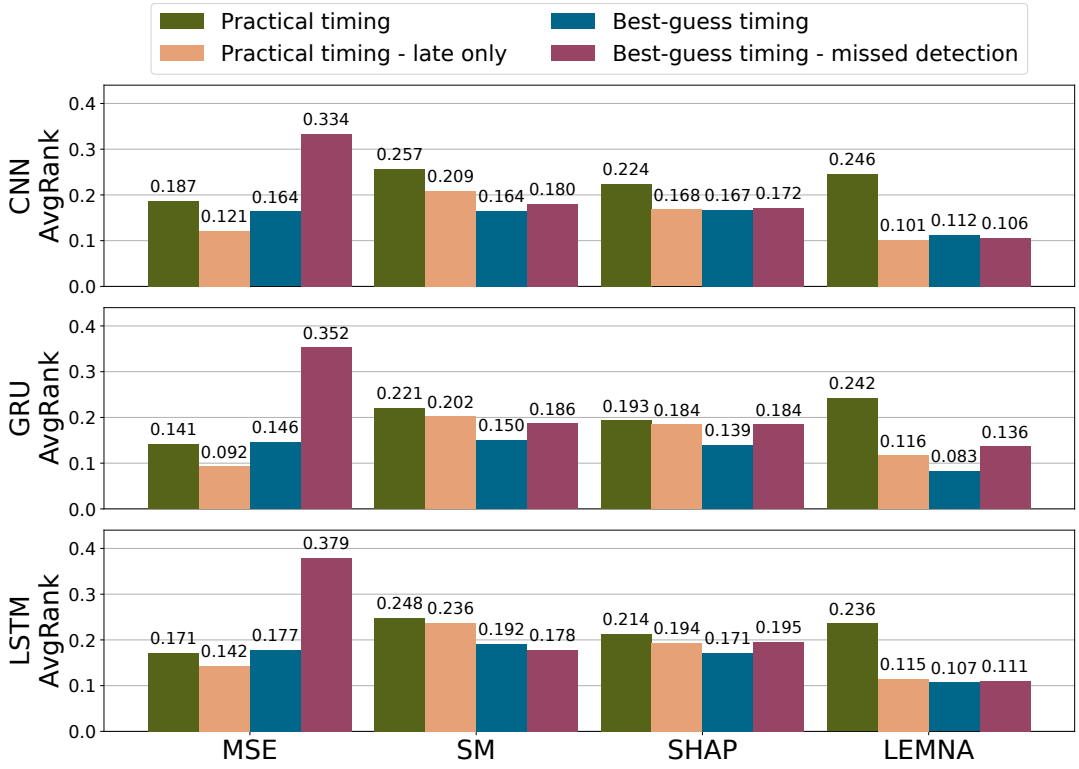


Figure 4.3: We report AvgRank (lower is better) for different timing strategies: "practical timing", which computes attribution immediately when anomalies are detected, and "best-guess timing", which computes attribution after 50 timesteps. Two additional variants are reported: practical timing with early detections removed, and best-guess timing for attacks that are not detected by the underlying model. In all cases, choosing an alternate timing strategy from the practical timing improves attribution accuracy.

to predict at the 51st timestep), and we call this strategy the "best-guess" timing.[2] To evaluate how detection timing affects attribution, we evaluate attribution methods across two timing strategies: when the input immediately precedes the detection time ("practical"), and when the input immediately precedes the 51st timestep ("best-guess").

Fig. 4.3 shows the AvgRank for different timing strategies. We first compare the "best-guess" and "practical" timings, and we find that best-guess timing outperforms practical timing in 10 out of 12 cases (including all cases with ML-based attribution methods). For example, when using best-guess timing with LEMNA, the AvgRank drops from 0.246 to 0.112 for CNNs, from 0.242 to 0.083 for GRUs, and from 0.236 vs 0.107 for LSTMS.

To analyze the impact of early detections on practical timing, we compare the AvgRank after removing attacks that are detected within the first 50 attacked timesteps ("practical timing" vs "practical timing—late only"). Although early detection of anomalies is beneficial in practice, it results in worse attributions. For example, when only considering detections after the first 50 timesteps, the MSE AvgRank on CNNs improves from 0.187 to 0.121. In all 12 cases, AvgRank improves when only "late" detections are considered.

[2]"Best-guess" since in practice the start time is not known.

Table 4.2: We perform statistical tests across our attribution results under "best-guess" timing: we compare the effect on AvgRank from manipulation magnitude and the type of feature attacked. All tested factors are found to be significant (p-value below 0.016, using Bonferroni correction). We find that all attribution methods perform better as attack magnitude increases. We also find that raw-error rankings (MSE) perform better on sensor-based attacks, whereas SM, SHAP and LEMNA perform better on actuator-based attacks (in each row, the AvgRank of the best performing subset is bolded).

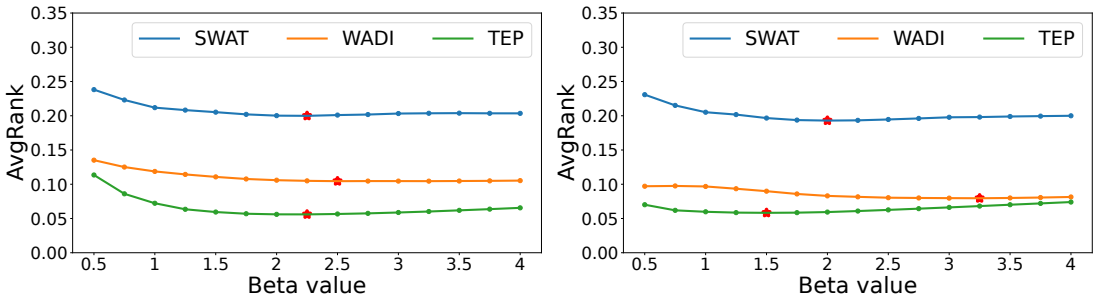
Attribution Method	Model	Magnitude (Pearson)		Sensor vs actuator (continuous) vs actuator (categorical) (ANOVA)		
		Corr.	p-value	AvgRank	F(2, 153)	p-value
MSE	CNN	-0.336	$p < 0.001$	**0.162** vs 0.306 vs 0.330	14.87	$p < 0.001$
	GRU	-0.445	$p < 0.001$	**0.154** vs 0.385 vs 0.361	17.95	$p < 0.001$
	LSTM	-0.399	$p < 0.001$	**0.152** vs 0.362 vs 0.360	13.21	$p < 0.001$
SM	CNN	-0.430	$p < 0.001$	0.176 vs **0.059** vs 0.281	13.28	$p < 0.001$
	GRU	-0.605	$p < 0.001$	0.160 vs **0.050** vs 0.330	21.67	$p < 0.001$
	LSTM	-0.558	$p < 0.001$	0.197 vs **0.047** vs 0.329	19.55	$p < 0.001$
SHAP	CNN	-0.497	$p < 0.001$	0.176 vs **0.059** vs 0.275	13.89	$p < 0.001$
	GRU	-0.595	$p < 0.001$	0.149 vs **0.054** vs 0.325	24.31	$p < 0.001$
	LSTM	-0.513	$p < 0.001$	0.181 vs **0.039** vs 0.333	23.39	$p < 0.001$
LEMNA	CNN	-0.544	$p < 0.001$	0.085 vs **0.034** vs 0.291	26.82	$p < 0.001$
	GRU	-0.537	$p < 0.001$	0.084 vs **0.034** vs 0.278	26.76	$p < 0.001$
	LSTM	-0.523	$p < 0.001$	0.084 vs **0.034** vs 0.248	27.23	$p < 0.001$

Finally, we compare the AvgRank between attacks that are detected and attacks that are missed by the anomaly-detection model, ("best-guess timing" vs "best-guess timing—missed detection"). For MSE, the performance is drastically worse when only considering attacks that are missed: the AvgRank increases from 0.164 to 0.334 for CNNs, from 0.146 to 0.352 for GRUs, and from 0.177 to 0.379 for LSTMs. However, when these same inputs are given to ML-based attribution methods, they perform approximately as well; for example, the LEMNA AvgRank on CNNs only changes from 0.112 to 0.106.

In summary, the timing of an attribution greatly affects its accuracy, and attributions computed at detection time (used in prior work [34, 42]) do not lead to the best performance. For best performance, ML-based attribution methods should be computed separately from detection outcomes; this could be performed with post-hoc incident analytics.

Manipulation magnitude After defining anomalies along common factors (as described in Sec. 4.2), we compute attributions at their best-guess timing and statistically test each factor's effect on AvgRank. The results of this analysis are shown in Table 4.2.

We first compute the effect of manipulation magnitude on AvgRank. Since the range of observed magnitudes in our dataset is large (0.06–91 standard deviations), we take the natural logarithm of the magnitude for our analysis. The first column of Table 4.2 shows, across all attacks, (i) the Pearson correlation coefficient between the log-scaled manipulation magnitude and AvgRank and (ii) the resulting p-value of the non-correlation test with Student's t-distribution. We



(a) "Best-guess" timing

(b) "Practical" timing

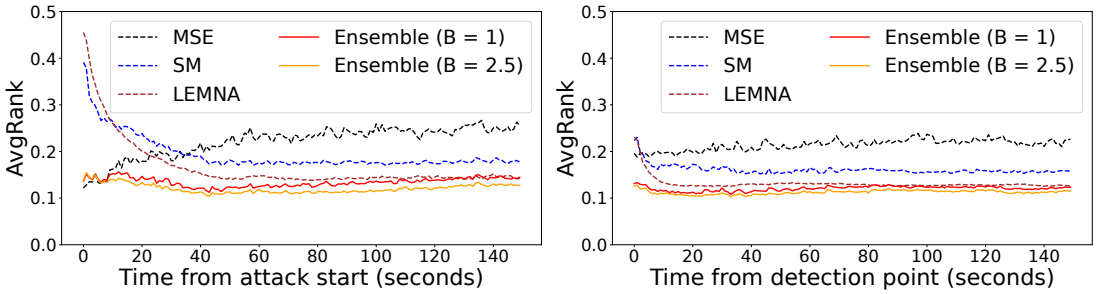
Figure 4.4: We evaluate different values of β for the ensemble's weighted averaging, at "best-guess" (left) and "practical" (right) timings. The optimal value of β is marked in red; for all settings, the optimal value of $\beta \in [1.5, 3.25]$, which indicates that our proposed weighted average outperforms the raw average.

observe a statistically significant relationship between manipulation magnitude and AvgRank. Since the anomaly-detection methods studied in this work rely on statistical modelling, lower-magnitude manipulations are more difficult to attribute. High-magnitude manipulations produce more immediate and obvious dispersions [30], so attribution methods that perform well on these manipulations may not be needed.

ICS feature types We next analyze how the type of feature that is manipulated affects AvgRank. We use a one-way ANOVA test to compare the distributions of AvgRank between manipulations on sensors, manipulations on categorical actuators, and manipulations on continuous actuators. The results of this test are provided in the second column of Table 4.2. Raw-error rankings perform significantly better for sensor-based attacks while ML-based attribution methods perform better on actuator-based attacks. For raw-error ranking (MSE), the AvgRank on sensors ranges from 0.152–0.162, whereas the AvgRank on actuators is always above 0.306. ML-based attribution methods perform best on continuous-valued actuators (AvgRank below 0.060), while still performing well on sensors (AvgRank below 0.198). This suggests that ML-based attribution methods may be able to capture relationships that connect sensors with their corresponding actuators, beyond what can be found by raw-error ranking. We also find that all attribution methods perform worst on categorical-valued actuators (AvgRank above 0.248), likely because our models cannot distinguish between categorical-valued and continuous-valued features during inference.

4.3.3 Improving attribution with an ensemble of methods

Different attribution methods perform best in different scenarios. ML-based attribution methods work best for continuous-valued actuators and at best-guess timings, whereas raw-error ranking works best for sensors and at practical timings. Thus, we design an ensemble of attribution methods to combine the strengths of different attribution methods: our ensemble uses a weighted average of attributions and is computed differently for sensors and actuators. We compute our ensemble over attributions from the raw-error ranking (MSE), SM, and LEMNA. Since SM and



(a) AvgRank over "best-guess" timing

(b) AvgRank over "practical" timing

Figure 4.5: Attribution averaging is evaluated on CNNs at two timings ("best-guess" on left, "practical" on right). The AvgRank (lower is better) of attribution methods is reported over 150 timesteps. Regardless of timing strategy and the selected timestep, an average of attribution methods outperforms any individual method.

LEMNA perform better for actuators than sensors (as shown in Sec. 4.3.2), we increase the importance of SM and LEMNA when computing attribution for actuators. We let β represent the weight of SM and LEMNA relative to the MSE. Our ensemble is computed as follows:

$$s_{AVG_\beta} = \begin{cases} s_{MSE} + s_{SM} + s_{LEMNA} & \text{if sensor} \\ s_{MSE} + \beta(s_{SM}) + \beta(s_{LEMNA}) & \text{if actuator} \end{cases}$$

We first perform a grid search over values of β for different datasets and timing strategies. Fig. 4.4 shows that optimal values of $\beta \in [1.5, 3.25]$, which indicates that our proposed weighted average outperforms the raw average (i.e., $\beta > 1$ outperforms when $\beta = 1$). We next compare the ensemble to individual attributions from MSE, SM, and LEMNA. Fig. 4.5 shows the AvgRank for various attribution methods (individual and ensemble) for CNNs over 150 timesteps, at both the best-guess (left) and practical (right) timing. We find that ML-based attribution methods are initially less accurate, but their accuracy improves over time; conversely, attributions from MSE are initially accurate but become less accurate over time.

In summary, our ensemble-based method produces the lowest AvgRank over most timesteps at both timings, showing that it can be used in practice when ground-truth timing is not known. Our findings remain the same when our evaluation of the ensemble attribution is repeated for GRUs and LSTMs.

4.4 Summary

In this chapter, we evaluate how effective prior and newly adapted attribution methods are when used to identify the manipulated feature in an ICS attack. We performed the first broad evaluation of ICS anomaly attribution, comparing across anomaly-detection methods, model architectures, and datasets. We found that ICS anomaly attribution is dependent on several factors. We examine these factors and identify challenges related to the timing of anomaly detection and differences in feature types. We ultimately develop a strategy that uses an ensemble of attribution methods and show that it outperforms all individual attribution methods.

Chapter 5

Leveraging graphs for ICS anomaly detection and attribution

Most approaches that have previously been proposed for ICS anomaly detection are general applications of models that treat all input features identically. However, the work described in Chap. 4 suggests that the limitations of prior approaches results in poor attribution performance for certain types of features (e.g., for boolean-valued actuators). We hypothesize that detection and attribution of ICS anomalies can be improved with approaches that use feature representations that are more specific to ICS. In this chapter, I describe an in-progress work that investigates if incorporating spatial and logical information in the form of graphs makes anomaly-detection models more effective when detecting and attributing anomalies.

5.1 Introduction

The work described in Chap. 4 found that raw-error rankings were inaccurate for anomaly attribution; in over 60% of attacks, the feature with the highest error did not correspond to the ICS component that was manipulated. When using anomaly-detection models based on architectures where features are fully connected with each other (e.g., DNNs, CNNs, LSTMs), *values from any input feature can affect any output*, meaning that a manipulation on any feature can increase errors in all other features.

Some anomaly-detection approaches from prior work improve interpretability by using model architectures that are not fully connected. One such approach uses Bayesian networks and timed automata to model ICS behavior, named TABOR [50]. By learning probabilistic relationships between features, TABOR can localize alarms to small sets of input features. Another approach uses only narrow, shallow convolutional layers and is named FCDD (fully convolutional data description) [52]. FCDD enforces that *only* convolutions are used, ensuring that output values only depend on a limited number of input features. As a result, FCDD produces explanations that are sparser than explanations from fully connected architectures. FCDD relies on image convolutions as a primitive and is specifically designed for image data. Although image-based convolutions are not applicable to ICS process values, another type of convolution that is relevant to ICS is graph-based convolution; graphs can encode the spatial and logical relationships

between ICS features and can be used with GNNs (graph neural networks) for ICS anomaly detection [16].

Motivated by the design of TABOR, FCDD, and GNN, we propose a novel anomaly-detection model architecture that (i) is trained on a graph-based abstraction of ICS and (ii) ensures that input features can only affect outputs that are semantically relevant to the manipulated feature. We hypothesize that this architecture, which we name TGDD (time-graph data description), will be more effective for anomaly detection and attribution than architectures used in prior anomaly-detection models (e.g. CNNs).

We also propose that changing the output format of anomaly-detection models can further improve attribution. Practitioners have reported that outputs from ML models and outputs from rule-based systems are both difficult to interpret [55], which indicates that neither a fully rule-based approach nor a fully ML-based approach is likely to be best for ICS anomaly detection and that new types of anomaly-detection output should be explored. Current anomaly-detection approaches produce errors and attributions that map directly to input features [25, 34, 42], which constrains explanations to formats that assign a score to each individual input feature. We propose that assigning scores to subgraphs, sets of a few ICS components, may be a more appropriate output format. Rather than ranking and identifying potentially anomalous features, identifying a potentially anomalous subset within an ICS (similar to the dependencies learned in TABOR [50]) may be more helpful when diagnosing and responding to anomalies.

5.2 Proposed Methodology

Our proposed use of TGDD relies on two steps. First, since we require graphs for training TGDD models, we describe methods for creating graph representations of ICS. Next, we describe how a TGDD model is constructed, using primitives from GNNs. We further describe how these same GNN primitives can be used to construct variants of TGDD models.

5.2.1 Creating graph representations of ICS

We use graphs as a tool for encoding the relationships between components in an ICS. Similar to an image convolution that performs computation over neighboring pixel values, a graph convolution is computed over neighboring node values defined by a graph. In our work, each node in the graph represents a sensor and actuator in the ICS, which is similar to prior work [16]. Edges between nodes can be learned dynamically (as in Deng et al. [16]) or manually specified. To dynamically learn graph edges, edges are created based on internal weight values during inference. Deng et al. use a threshold on embedding similarity to determine if an edge is created or not; we plan to implement and evaluate this approach, but also propose an approach that uses sigmoid activation functions and learned weights to dynamically create edges between features (described in Sec. 5.2.2). To manually specify graph edges, edges are created between nodes to represent relationships between sensors and actuators; these relationships can either be physical (i.e., the flow sensor of a pipe and the level sensor of the tank that the pipe flows into) or logical (i.e., a PLC uses the values from a sensor to set an actuator value). We manually create a graph

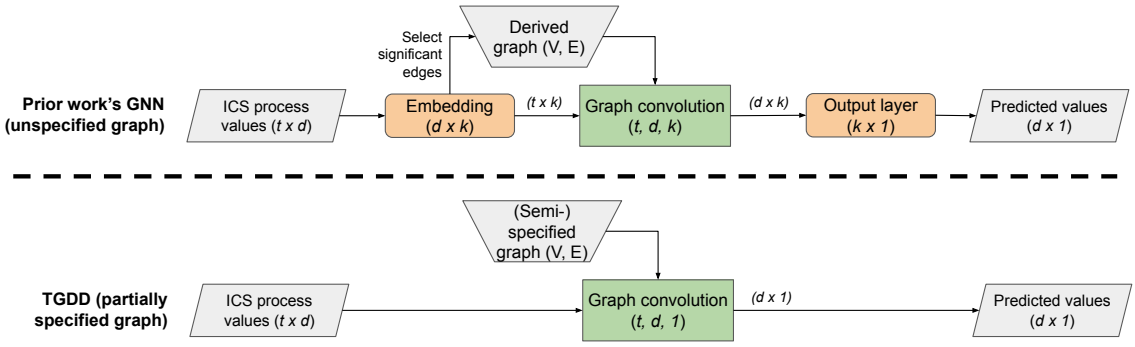


Figure 5.1: An overview of the differences between TGDD (our proposal), and unspecified GNNs from prior work [16]. Unlike prior work, TGDD encourages sparsity by specifying parts of the graph used in graph convolution, and by removing the embedding layer. We provide details on the different variants of graph specification that are supported in Sec. 5.2.2.

representation of TEP by manually inspecting the MATLAB simulator.[1]

5.2.2 Designing TGDD and its variants

TGDD, our proposed anomaly-detection architecture, relies on graph convolution layers as a primitive for inference. A graph convolution layer (d_{in}, n, d_{out}) is defined by three hyperparameters: the input dimension d_{in}, the number of nodes n, and the output dimension d_{out}. When performing inference, a graph convolution layer takes a set of d_{in}-by-n values and a graph as input and produces a set of n-by-d_{out} values as output. In designing TGDD, we apply graph convolution layers to ICS process values, but limit inter-feature influence by strictly enforcing that only shallow graph convolutions are performed. Figure 5.1 shows the differences between TGDD and GNNs from prior work [16].

In contrast to dynamically learning a graph structure from the full range of all possible edges and using a different set of edges at each inference, TGDD uses a fixed set of edges and therefore requires specification of a graph structure. Furthermore, TGDD uses an output dimension of one, removing the need for a final output layer; this maintains fewer model parameters and further encourages sparsity.

We suspect that specifying a full, correctly defined set of edges for an ICS may be difficult. To obviate this need, we support partial specification of graph edges: we define edges with weight values that pass through a sigmoid activation function. These edge weights are then learned during training; if these edge-weight values are below zero, then the sigmoid function is not activated and the edge is effectively removed from the graph. In order words, we use edge weights to abstractly define three options for how outputs can be influenced by neighboring node values:

- **(Fixed)**: An edge uses a fixed weight value of one; neighboring values are included in the graph convolution operation.

[1]Since the PLC code is explicitly written in MATLAB, edges from logical connections are precisely defined. The simulation of the physical process in TEP is imported as a C library and is more difficult to interpret, so edges from physical connections are estimated from the TEP process diagram

Table 5.1: For each architecture, we perturb an input feature and record the number of output features that change. We repeat and average over all 53 features in TEP. Far fewer features change when features are modified for TGDD, indicating that it may exhibit the sparsity properties that are desired for attribution.

Architecture	Average # of features changed (out of 53)
CNN (from Chap. 4)	53.0
GNN [16]	48.8
TGDD	2.7

- **(Learnable)**: An edge uses a sigmoid-activated weight; only neighboring values from activated edges are included in the graph convolution operation.
- **(Removed)**: An edge uses fixed weight value of zero; it is never included in the graph convolution operation.

In addition to using fully specified or fully unspecified graphs, we define a special instance of TGDD, which we call *feature-set* TGDD. Configuring feature-set TGDD does not require any edges to be specified; only *feature sets* are given. Given a feature set, learnable edges are defined between all pairs of features within this set, and edges to any feature not defined in this feature set are removed; thus, the ICS is represented by a set of disjoint subgraphs. Some features may be relevant to multiple subgraphs, so we learn separate representations of these features for each subgraph. In other words, when k feature sets are defined, we learn k TGDD submodels and maintain a separate set of predictions and detections for each submodel.

Feature-set TGDD maintains sparsity, as only features within the defined feature set can influence each other. Since feature-set TGDD is an intermediate solution in the design tradeoffs between TGDD and GNNs, we hypothesize that feature-set TGDD may be a more effective approach for graph-based attribution of ICS anomalies.

5.3 Preliminary Results

In this section, I present two preliminary results which indicate that TGDD has properties that are desirable for anomaly detection and attribution. First, we perform a set of perturbation experiments to confirm that changes in input values have a limited affect on TGDD model outputs. We first train an anomaly-detection model for each architecture on the TEP dataset. We then perform anomaly detection with each model on two inputs: (i) a benign instance from the training data and (ii) a copy of this benign instance where a single feature value is increased by two standard deviations. We then measure how many features in the output differ between these two inputs; in other words, we measure how many values in the output change when a single input is changed. We repeat this test for each anomaly-detection model, perturbing all 53 features in TEP and reporting the average. The results of this test are shown in Table 5.1; for TGDD, far fewer output values change when inputs are perturbed. This preliminary result suggests that TGDD exhibits the sparsity that is desirable for anomaly attribution.

Second, we implement various anomaly-detection model architectures: CNN, GNN, and our

Table 5.2: We compare the attribution accuracy (using AvgRank) across model architectures (and their hyperparameter variants). We find that approaches based on feature-set GNNs outperform all baseline approaches (CNNs, GNNs with no specified graph, and a fully-specified graph), suggesting that partial specification is likely to be the best approach for anomaly attribution. Feature-set GNNs based only on PLCs produce the lowest AvgRank; we propose further analysis of how the level of graph specification affects attribution.

Architecture and hyperparameters	**AvgRank**
CNN (from Chap. 4)	0.246
GNN with no specified graph [16]	0.379
Fully-specified TGDD	0.369
Feature-set TGDD (Fine-grained, $k = 53$)	0.233
Feature-set TGDD (PLC-based, $k = 9$)	**0.138**
Feature-set TGDD (PLC-based + process, $k = 10$)	0.237

various proposed configurations of TGDD. We then compare our trained model's AvgRank over a baseline set of 22 TEP attacks[2]. We compare a TGDD model that uses a fully specified graph to multiple feature-set TGDD models that use varying levels of granularity:

- Fine-grained feature-set TGDD defines a feature set for all 53 components in TEP. Each feature set is defined by the neighbors in the fully specified TEP graph.
- PLC-based feature-set TGDD defines a feature set for all 9 TEP PLCs. Each feature set is defined by the sensors and actuators that are used as input to a PLC.
- We also add to the PLC-based feature-set TGDD a single feature set that includes all sensors used in the TEP process, to capture physical relationships between features that span multiple PLCs.

When performing attribution with the feature-set TGDD models, we identify the submodel that detects the attack with the lowest latency, and provide its entire feature set as attribution output. If the manipulated feature is not included in this set, the feature set of the submodel with the next lowest latency is used, continuing until the manipulated feature is found. Table 5.2 shows the AvgRank for each approach. We find that the PLC-based feature-set TGDD model produces the lowest AvgRank (0.138). We also find that all feature-set TGDD models outperform the CNN and GNN (i.e, AvgRank is lowest for all feature-set TGDD models).

We declare that an attribution from a feature-set TGDD model is correct when the manipulated ICS component is included in its first predicted feature set. We find that relatively few attacks are correctly attributed by the PLC-based feature-set TGDD model (only 5 out of 22 attacks are attributed in the first feature set). After including a feature set for the physical process, 20 out of 22 attacks are correctly attributed, which suggests that an approach based only on PLC relationships is insufficient for attribution. However, the size of the feature set that corresponds to the TEP process is large (25 features), so adding this feature set lowers the resulting AvgRank

[2]Using the same definition and methodology as in Sec. 4.2

from 0.138 to 0.237. These preliminary results suggest that feature-set TGDD models can be a more effective approach for attributing anomalies than prior alternatives (e.g., CNNs and GNNs), but further experiments are needed to investigate what level of granularity is best and how their attributions are affected by different attack types.

5.4 Proposed Work

I propose two areas of work for improving and evaluating TGDD: (i) evaluating the implications of TGDD's design elements and (ii) creating new attack scenarios to better evaluate TGDD.

Evaluating TGDD Our current abstraction of TGDD supports instantiations of feature-set TGDD and fully specified TGDD; however, models with an intermediate level of specificity are also supported. For example, a heterogenous solution could include several feature sets in combination with a partially specified subgraph. I propose further experiments that investigate such configurations, varying the level of specificity required by the user. Furthermore, I propose experiments that explore how the size of feature sets used in feature-set TGDD affect anomaly detection and attribution; one potential variant of feature-set TGDD could represent the physical process with multiple, more granular feature sets. Lastly, inspired by Deng et al. [16], I propose to explore ways in which graphs, subgraphs, or feature sets can be generated automatically: either from process data or from other auxiliary sources of information. An ideal approach improves anomaly detection and attribution effectiveness, while only minimally increasing the amount of configuration required by the practitioner who deploys and maintains this approach.

New attacks for evaluating TGDD Attacks in prior ICS attack datasets commonly use constant manipulation patterns and attack one (or at most a few) feature(s) at a time [6, 29]. Since we propose that using feature relationships can improve anomaly-detection models, we also propose that these relationships can be used to improve attacks on ICS. For example, a pair of ICS components could be attacked simultaneously such that the effects of one manipulation negates the expected increase in MSE from another. We hypothesize that an investigation of this attack strategy will reveal that TGDD and other graph-based anomaly-detection approaches are more robust to these attacks than prior MSE-based anomaly-detection approaches.

In a similar type of attack, prior work uses a blackbox adversarial search to find candidate features for stealthy manipulations [19, 20], we hypothesize that information from graphs can be used to (i) outperform prior blackbox attacks or (ii) make such adversarial attacks more efficient. Ultimately, feature relationships can likely be used to inform more stealthy attacks on ICS, and building a better understanding of attack strategies is required to design ICS anomaly-detection approaches that are robust to such attacks.

Chapter 6

Examining practitioner's perspectives of ICS monitoring and alarms

The other works described in this thesis measure the effectiveness of existing anomaly-detection approaches (Chapter 3 and 4) and propose adaptations to them (Chapter 4 and 5), but these works are based on experimental settings. It remains unclear how effective anomaly-detection approaches would be when deployed and used in practice, and a strong consideration of both organizational and end-user needs is required to build this understanding [12].

In this chapter, I describe an in-progress study of practitioners that monitor and control ICS. We ask practitioners about their perspectives when configuring and responding to ICS alarms, and we analyze their responses to determine if current anomaly-detection approaches are suitable for ICS environments.

6.1 Introduction

Systems for monitoring and securing ICS are needed to detect anomalies and enable timely response; such systems could rely on pre-defined rules [27], deep-learning models [54], and/or human involvement [49, 75]. Although several prior works in research have focused on deep-learning-based models for ICS anomaly detection [24, 42, 57, 68, 77], it is unclear if deep-learning-based approaches have been adopted in industry, and if they are perceived to be effective. This study builds a stronger understanding of what tools ICS practitioners use, what aspects of these tools they find useful, and what challenges they currently experience when responding to alarms from these tools.

Prior work has studied the perspectives of practitioners in security contexts that are similar to ICS anomaly detection. For example, prior work studies analysts in security operations centers (SOCs) and their experiences with anomaly-detection systems [8, 39, 72]; aspects such as the type of detection system, error rates, and usability were identified as important factors in evaluating how effective anomaly-detection systems were. Other prior works study the various perceptions that practitioners have of machine-learning-based tools for general (i.e., not specific to anomaly detection) security tasks [11, 55]; these works focus on factors such as false-positive and false-negative rates, and the interpretability of ML models.

Perspectives of tools depend on the context in which they are used, and we observe that the findings in these studies can differ. Alahmadi et al. find that SOC analysts are often overwhelmed with a high rate of false positives, but many false positives are trivial to dismiss and were therefore not of high concern [8]. Conversely, Mink et al. report that false positives were of greater concern than false negatives to practitioners [55] and that false positives were a significant challenge when using ML-based solutions.

To best understand the needs of practitioners that work with ICS, a study that focuses specifically on their perspective is needed. Although prior works focus on areas that are relevant to ICS anomaly detection, none of them focus specifically on organizations that manage and operate ICS. ICS are unique from most computer-based systems in their widespread impact, real-time critical nature and the organizational structures that are used to manage them [26]. In this work, we explore how prior findings related to anomaly detection and ML may differ when applied in the context of securing ICS. Our study focuses on the following research questions:

- **RQ1**: In organizations that manage and operate ICS, what are current barriers to adopting ICS anomaly detection systems?
- **RQ2**: What criteria are most important to practitioners when selecting, using, and evaluating ICS anomaly-detection systems?
- **RQ3**: How do unique aspects of ICS (e.g., their critical nature, domain expertise needed for operators) affect what is preferred when selecting, using, and evaluating ICS anomaly-detection systems?

6.2 Methodology

We are conducting a series of semi-structured interviews with practitioners in various industries and eliciting responses regarding their day-to-day use of anomaly detection, their perceived benefits and drawbacks of different aspects of anomaly detection, and their suggestions for future design and implementation of anomaly-detection systems.

Our study focuses on practitioners who directly monitor an ICS for its safe operation: depending on the organization, this includes roles such as SCADA engineer and control-room operator, as well as data scientists that investigate alarms based on real-time process information. We are currently recruiting study participants through a variety of channels: emailing within our personal networks; through public-facing contact pages, forms, and mailing lists; and through social media networks, such as LinkedIn and X (formerly Twitter). Our study's recruitment materials and methods have been approved by Carnegie Mellon University's Internal Review Board (IRB).

Our interviews follow a semi-structured script that covers three major topics: (i) how anomaly detection is performed at the participant's organization, (ii) how anomaly-detection approaches are evaluated at the participant's organization, and (iii) what the participant's perceptions are of different anomaly-detection approaches. Since our participants will differ in their industries and levels of expertise, we primarily ask broad, open-ended questions and allow room for follow-up questions and feedback. To ensure question clarity, we tested our interview script with two other researchers (who are not directly involved with this work) with experience in human subjects

research with practitioners that work with ICS.

We have completed thirteen participant interviews and are beginning to analyze their transcripts. Once the interviews are fully transcribed, multiple authors of this work will perform qualitative coding to identify themes in participant responses. These themes will be compared for consistency, and then subsequently analyzed to identify broader trends and patterns in responses. We will then continue to recruit participants and conduct interviews until achieving concept saturation (i.e., when new themes no longer emerge).

6.3 Proposed Analysis

Based on our preliminary results from these interviews, we anticipate that themes will be categorized in the following areas:

- Data ownership: where process data is being read from, where process data is being shown, and who is responsible for these channels.
- Organizational structure around anomaly-detection systems: who decides if anomaly detection is used, how it is used, and what types are systems are used.
- Organizational structure around alarm remediation: whether alarms from anomaly detection can be acted on independently, or whether other individuals or organizations are required in this workflow.
- Benefits and drawbacks of anomaly detection based on their type: rule-based systems, machine-learning-based systems, or others.
- Perceptions of rule-based and machine-learning-based anomaly-detection systems: how trustworthy their outputs are, how technically challenging they are to deploy, and what is possible with them.

Based on the patterns we identify, we will make recommendations for researchers to design anomaly-detection systems with properties that are better aligned to organizations that work with ICS, easing adoption. We will also compare and contrast our findings to those found in studies that focus on SOC operations [8, 55], machine learning for computer security [11, 55], and organizational structures of ICS [26]. In doing so, we will identify similarities and differences between ICS anomaly detection and other computer-security domains, and we will make distinct recommendations for the ICS anomaly detection context.

By drawing conclusions based on the experiences of real ICS practitioners, this study will add context to the findings from prior studies when anomaly-detection solutions are applied in practice and suggest new challenges and directions for making research in ICS anomaly detection more useful in practice.

Chapter 7

Timeline

The proposed timeline for the remaining work is as follows:

1. **July 2024**: Propose thesis.
2. **from July to September 2024**: Conduct and analyze remaining practitioner interviews.
3. **September 2024**: Finish analysis of interview-based practitioner study (Chap. 6), write results into paper, and submit to ACM CHI 2025.
4. **from July to December 2024**: Conduct remaining experiments on graph-based detection and attribution.
5. **January 2025**: Finish analysis of graph-based detection and attribution models (Chap. 5) write results into paper, and submit to ACM CCS 2025 (exact deadline to be confirmed).
6. **February 2025**: Finish writing thesis.
7. **March 2025**: Defend thesis.

Bibliography

[1] Black-box explanation of deep learning models using mixture regression models. `https://github.com/Henrygwb/Explaining-DL`, 2021.

[2] SHAP: A game theoretic approach to explain the output of any machine learning model. `https://github.com/shap/shap`, 2021.

[3] Maged Abdelaty, Roberto Doriguzzi-Corin, and Domenico Siracusa. DAICS: A deep learning solution for anomaly detection in industrial control systems. *arXiv preprint arXiv:2009.06299*, 2020.

[4] Ahmed A. Abokifa, Kelsey Haddad, Cynthia S. Lo, and Pratim Biswas. Detection of cyber physical attacks on water distribution systems via principal component analysis and artificial neural networks. In *World Environmental and Water Resources Congress*, pages 676–691, 2017.

[5] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6, 2018.

[6] Chuadhry Mujeeb Ahmed, Venkata Reddy Palleti, and Aditya P Mathur. WADI: a water distribution testbed for research in the design of secure cyber physical systems. In *3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, 2017.

[7] Chuadhry Mujeeb Ahmed, Gauthama Raman MR, and Aditya P Mathur. Challenges in machine learning based approaches for real-time anomaly detection in industrial control systems. In *6th ACM Cyber-Physical System Security Workshop*, 2020.

[8] Bushra A. Alahmadi, Louise Axon, and Ivan Martinovic. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In *31st USENIX Security Symposium*, 2022.

[9] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. Explaining anomalies detected by autoencoders using shapley additive explanations. *Expert systems with applications*, 186, 2021.

[10] Wissam Aoudi, Mikel Iturbe, and Magnus Almgren. Truth will out: Departure-based process-level detection of stealthy attacks on control systems. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[11] Giovanni Apruzzese, Pavel Laskov, Edgardo Montes de Oca, Wissam Mallouli, Luis Brdalo Rapa, Athanasios Vasileios Grammatopoulos, and Fabio Di Franco. The role of machine learning in cybersecurity. *Digital Threats: Research and Practice*, 4(1), 2023.

[12] Mohammed Asiri, Neetesh Saxena, Rigel Gjomemo, and Pete Burnap. Understanding indicators of compromise against cyber-attacks in industrial control systems: A security perspective. *ACM Transactions on Cyber-Physical Systems*, 2023.

[13] Andreas Bathelt, N Lawrence Ricker, and Mohieddine Jelali. Revision of the Tennessee Eastman process model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.

[14] Margret Bauer, Alexander Horch, Lei Xie, Mohieddine Jelali, and Nina Thornhill. The current state of control loop performance monitoring–a survey of application in industry. *Journal of Process Control*, 38, 2016.

[15] Alvaro A Cardenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: risk assessment, detection, and response. In *6th ACM Symposium on Information, Computer and Communications Security*, 2011.

[16] Ailin Deng and Bryan Hooi. Graph neural network-based anomaly detection in multivariate time series. In *AAAI Conference on Artificial Intelligence*, 2021.

[17] Alessandro Di Pinto, Younes Dragoni, and Andrea Carcano. TRITON: The first ICS cyber attack on safety instrument systems. In *Black Hat USA*, 2018.

[18] James J Downs and Ernest F Vogel. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, 17(3):245–255, 1993.

[19] Alessandro Erba and Nils Ole Tippenhauer. Assessing model-free anomaly detection in industrial control systems against generic concealment attacks. In *38th Annual Computer Security Applications Conference*, 2022.

[20] Alessandro Erba, Riccardo Taormina, Stefano Galelli, Marcello Pogliani, Michele Carminati, Stefano Zanero, and Nils Ole Tippenhauer. Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems. In *36th Annual Computer Security Applications Conference*, 2020.

[21] G Erion, JD Janizek, P Sturmfels, S Lundberg, and SI Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *Nature Machine Intelligence*, 3, 2021.

[22] Cheng Feng, Venkata Reddy Palleti, Aditya Mathur, and Deeph Chana. A systematic framework to generate invariants for anomaly detection in industrial control systems. In *Network and Distributed System Security Symposium*, 2019.

[23] Pavel Filonov, Fedor Kitashov, and Andrey Lavrentyev. RNN-based early cyber-attack detection for the Tennessee Eastman process. *arXiv preprint arXiv:1709.02232*, 2017.

[24] Clement Fung, Shreya Srinarasi, Keane Lucas, Hay Bryan Phee, and Lujo Bauer. Perspectives from a comprehensive evaluation of reconstruction-based anomaly detection in industrial control systems. In *27th European Symposium on Research in Computer Security*, 2022.

[25] Clement Fung, Eric Zeng, and Lujo Bauer. Attributions for ML-based ICS anomaly detection: From theory to practice. In *31st Network and Distributed System Security Symposium*, 2024.

[26] Andrea Gallardo, Robert Erbes, Katya Le Blanc, Lujo Bauer, and Lorrie Faith Cranor. Interdisciplinary approaches to cybervulnerability impact assessment for energy critical infrastructure. In *CHI Conference on Human Factors in Computing Systems*, 2024.

[27] Jairo Giraldo, David Urbina, Alvaro Cardenas, Junia Valente, Mustafa Faisal, Justin Ruths, Nils Ole Tippenhauer, Henrik Sandberg, and Richard Candell. A survey of physics-based attack detection in cyber-physical systems. *ACM Computing Surveys*, 51(4):1–36, 2018.

[28] J. Goh, S. Adepu, M. Tan, and Z. S. Lee. Anomaly detection in cyber physical systems using recurrent neural networks. In *18th International Symposium on High Assurance Systems Engineering*, 2017.

[29] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*, 2016.

[30] Benjamin Green, Marina Krotofil, and Ali Abbasi. On the significance of process comprehension for conducting targeted ICS attacks. In *Workshop on Cyber-Physical Systems Security and Privacy*, 2017.

[31] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, Gang Wang, and Xinyu Xing. LEMNA: Explaining deep learning based security applications. In *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[32] Dina Hadžiosmanović, Robin Sommer, Emmanuele Zambon, and Pieter H. Hartel. Through the eye of the PLC: Semantic security monitoring for industrial processes. In *30th Annual Computer Security Applications Conference*, 2014.

[33] Grant Ho, Mayank Dhiman, Devdatta Akhawe, Vern Paxson, Stefan Savage, Geoffrey M. Voelker, and David Wagner. Hopper: Modeling and detecting lateral movement. In *30th USENIX Security Symposium*, 2021.

[34] Chanwoong Hwang and Taejin Lee. E-SFD: Explainable sensor fault detection in the ICS anomaly detection system. *IEEE Access*, 9:140470–140486, 2021.

[35] Won-Seok Hwang, Jeong-Han Yun, Jonguk Kim, and Hyoung Chun Kim. Time-series aware precision and recall for anomaly detection: Considering variety of detection result and addressing ambiguous labeling. In *28th ACM International Conference on Information and Knowledge Management*, 2019.

[36] J. Inoue, Y. Yamagata, Y. Chen, C. M. Poskitt, and J. Sun. Anomaly detection for a water treatment system using unsupervised machine learning. In *IEEE International Conference on Data Mining Workshops*, 2017.

[37] Albert T. Jones and Charles R. McLean. A proposed hierarchical control model for automated manufacturing systems. *Journal of Manufacturing Systems*, 5(1), 1986.

[38] Jonguk Kim, Jeong-Han Yun, and Hyoung Chun Kim. Anomaly detection for industrial control systems using sequence-to-sequence neural networks. In *5th Workshop on the Security of Industrial Control Systems and of Cyber-Physical Systems*, 2019.

[39] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. Matched and mismatched SOCs: A qualitative study on secu-

rity operations center issues. In *ACM SIGSAC Conference on Computer and Communications Security*, 2019.

[40] Moshe Kravchik and Asaf Shabtai. Detecting cyber attacks in industrial control systems using convolutional neural networks. In *Workshop on Cyber-Physical Systems Security and Privacy*, 2018.

[41] Moshe Kravchik and Asaf Shabtai. Efficient cyber attacks detection in industrial control systems using lightweight neural networks and PCA. *arXiv preprint arXiv:1907.01216*, 2019.

[42] Moshe Kravchik and Asaf Shabtai. Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA. *IEEE Transactions on Dependable and Secure Computing*, 19(4), 2022.

[43] Marina Krotofil and Jason Larsen. Rocking the pocket book: Hacking chemical plants. In *DEFCON Conference*, 2015.

[44] Nir Kshetri and Jeffrey Voas. Hacking power grids: A current problem. *Computer*, 50(12), 2017.

[45] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the Numenta anomaly benchmark. In *14th International Conference on Machine Learning and Applications*, 2015.

[46] D. Lavrova, D. Zegzhda, and A. Yarmak. Using GRU neural network for cyber-attack detection in automated process control systems. In *IEEE International Black Sea Conference on Communications and Networking*, 2019.

[47] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[48] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, 2019.

[49] Karen Li, Awais Rashid, and Anne Roudaut. Usable security model for industrial control systems - authentication and authorisation workflow. In *Proceedings of the 2023 European Symposium on Usable Security*, 2023.

[50] Qin Lin, Sridha Adepu, Sicco Verwer, and Aditya Mathur. TABOR: A graphical model-based approach for anomaly detection in industrial control systems. In *Asia Conference on Computer and Communications Security*, 2018.

[51] Yao Liu, Peng Ning, and Michael K. Reiter. False data injection attacks against state estimation in electric power grids. *ACM Transactions on Information and System Security*, 2011.

[52] Philipp Liznerski, Lukas Ruff, Robert A Vandermeulen, Billy Joe Franks, Marius Kloft, and Klaus Robert Muller. Explainable deep one-class classification. In *International Conference on Learning Representations*, 2021.

[53] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, 2017.

[54] Yuan Luo, Ya Xiao, Long Cheng, Guojun Peng, and Danfeng Yao. Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities. *ACM Computing Surveys*, 54(5):1–36, 2021.

[55] Jaron Mink, Hadjer Benkraouda, Limin Yang, Arridhana Ciptadi, Ali Ahmadzadeh, Daniel Votipka, and Gang Wang. Everybody's got ML, tell me what else you have: Practitioners' perception of ML-based security tools and explanations. In *IEEE Symposium on Security and Privacy*, 2023.

[56] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. Kitsune: an ensemble of autoencoders for online network intrusion detection. In *Network and Distributed System Security Symposium*, 2018.

[57] Ángel Luis Perales Gómez, Lorenzo Fernández Maimó, Alberto Huertas Celdrán, and Félix J. García Clemente. MADICS: A methodology for anomaly detection in industrial control systems. *Symmetry*, 12(10), 2020.

[58] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?" explaining the predictions of any classifier. In *22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[59] Dmitry Shalyga, Pavel Filonov, and Andrey Lavrentyev. Anomaly detection for water treatment system based on neural network with automatic architecture optimization. *arXiv:1807.07282*, 2018.

[60] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[61] Brian Singer, Amritanshu Pandey, Shimiao Li, Lujo Bauer, Craig Miller, Lawrence Pileggi, and Vyas Sekar. Shedding light on inconsistencies in grid cybersecurity: Disconnects and recommendations. In *IEEE Symposium on Security and Privacy*, 2023.

[62] Pooja Singh and Lalit Kumar Singh. Instrumentation and control systems design for nuclear power plant: An interview study with industry practitioners. *Nuclear Engineering and Technology*, 53(11), 2021.

[63] Joseph Slowik. Evolution of ICS attacks and the prospects for future disruptive events. *Threat Intelligence Centre Dragos Inc*, 2019.

[64] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[65] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4), 2009.

[66] Keith Stouffer. Guide to industrial control systems (ICS) security. *NIST special publication*, 800(82), 2011.

[67] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *34th International Conference on Machine Learning*, 2017.

[68] Riccardo Taormina and Stefano Galelli. Deep-learning approach to the detection and localization of cyber-physical attacks on water distribution systems. *Journal of Water Resources*

Planning and Management, 144(10), 2018.

[69] Riccardo Taormina, Stefano Galelli, Nils Ole Tippenhauer, Elad Salomons, Avi Ostfeld, Demetrios G Eliades, Mohsen Aghashahi, Raanju Sundararajan, Mohsen Pourahmadi, M Katherine Banks, et al. Battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. *Journal of Water Resources Planning and Management*, 144(8), 2018.

[70] Nesime Tatbul, Tae Jun Lee, Stan Zdonik, Mejbah Alam, and Justin Gottschlich. Precision and recall for time series. In *Advances in Neural Information Processing Systems*, 2018.

[71] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the impact of stealthy attacks on industrial control systems. In *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[72] Mathew Vermeer, Natalia Kadenko, Michel van Eeten, Carlos Gañán, and Simon Parkin. Alert alchemy: SOC workflows and decisions in the management of NIDS rules. In *ACM SIGSAC Conference on Computer and Communications Security*, 2023.

[73] Konrad Wolsing, Lea Thiemt, Christian van Sloun, Eric Wagner, Klaus Wehrle, and Martin Henze. Can industrial intrusion detection be SIMPLE? In *27th European Symposium on Research in Computer Security*, 2022.

[74] Renjie Wu and Eamonn J Keogh. Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE Transactions on Knowledge and Data Engineering*, 35(3), 2021.

[75] Alberto Zanutto, Benjamin Oliver Shreeve, Karolina Follis, Jeremy Simon Busby, and Awais Rashid. The shadow warriors: In the no man's land between industrial control systems and enterprise IT systems. In *Symposium on Usable Privacy and Security*, 2017.

[76] Carson Zimmerman. Cybersecurity operations center. *The MITRE Corporation*, 2014.

[77] Giulio Zizzo, Chris Hankin, Sergio Maffeis, and Kevin Jones. Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks. *arXiv preprint arXiv:1911.04278*, 2019.