Attributions for ML-based ICS Anomaly Detection: From Theory to Practice

Clement Fung
Carnegie Mellon University
clementf@cs.cmu.edu

Eric Zeng
Carnegie Mellon University
ericzeng@cmu.edu

Lujo Bauer
Carnegie Mellon University
lbauer@cmu.edu

Abstract*—Industrial Control Systems (ICS) govern critical infrastructure like power plants and water treatment plants. ICS can be attacked through manipulations of its sensor or actuator values, causing physical harm. A promising technique for detecting such attacks is machine-learning-based anomaly detection, but it does not identify which sensor or actuator was manipulated and makes it difficult for ICS operators to diagnose the anomaly's root cause. Prior work has proposed using attribution methods to identify what features caused an ICS anomaly-detection model to raise an alarm, but it is unclear how well these attribution methods work in practice. In this paper, we compare state-of-the-art attribution methods for the ICS domain with real attacks from multiple datasets. We find that attribution methods for ICS anomaly detection do not perform as well as suggested in prior work and identify two main reasons. First, anomaly detectors often detect attacks either immediately or significantly after the attack start; we find that attributions computed at these detection points are inaccurate. Second, attribution accuracy varies greatly across attack properties, and attribution methods struggle with attacks on categorical-valued actuators. Despite these challenges, we find that ensembles of attributions can compensate for weaknesses in individual attribution methods. Towards practical use of attributions for ICS anomaly detection, we provide recommendations for researchers and practitioners, such as the need to evaluate attributions with diverse datasets and the potential for attributions in non-real-time workflows.

I. Introduction

Industrial control systems (ICS) govern our critical infrastructure, including power grids, water treatment, and manufacturing processes. Because of this critical nature, attackers aim to infiltrate the ICS and interfere with their physical processes [58]; prominent examples include Stuxnet [41], attacks on the Ukrainian power grid [52], and an attack on a German blast furnace [70].

ICS operate on *process-level data*: data from *sensors*, which read information from a physical process, and data from *actuators*, which send commands to control a physical process [60]. A commonly studied type of attack involves manipulating process-level data of an ICS [15], [44]: an attacker gains access to an ICS and manipulates one or more sensor or actuator values, causing the ICS to react in a harmful way (e.g., a tank to overflow, a reactor to overheat).

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23216
www.ndss-symposium.org

To detect these manipulations in real time, a variety of anomaly-detection approaches have been proposed [10], [26], [31]. A common approach for ICS anomaly-detection uses machine learning (ML): process-level values (i.e., ICS states) are represented as features in a ML model, and the model is trained to predict future ICS states (i.e., a per-feature prediction of each sensor and actuator value) from an input window of prior states. At test time, the predicted state is compared with the observed state; if the total difference between the prediction and observation (i.e., reconstruction error) exceeds a threshold, an anomaly is declared [39], [72]. This approach has been shown to effectively detect attacks across various types of ICS [7], [19], [27] and with various model architectures [18], [24], [39], [72].

A key part of ICS operators' response to an attack is identifying its cause [13]. However, most proposals for using ML-based anomaly detection in ICS only identify whether an ICS as a whole is in a normal or anomalous state [72].

In this work, we investigate if *attribution* for anomaly detection methods could help operators identify which sensors or actuators are the cause of an ICS anomaly. Attribution is a technique for quantifying the impact of the input features on a model's prediction [51], [56], [61]. *Attribution methods*, such as saliency maps [56] and SHAP [45], have been developed for other domains of ML (e.g., images).

Prior work in ICS anomaly detection has proposed that suggesting features with the highest reconstruction errors can sufficiently guide operators to the location of the manipulation [34], [39]. However, these studies evaluate attribution with few attacks, and prior work in general has not systematically studied the accuracy and quality of attribution methods for ICS anomaly detection.

In this work, we investigate two research questions:

- **RQ1**: Can attribution methods accurately identify the manipulated feature in an ICS attack? Which method is most accurate?
- **RQ2**: What properties of ICS attacks affect the accuracy of attributions? E.g., the timing of the anomaly detection, the manipulation magnitude, or type of component attacked.

To investigate these questions, we conduct a comparative evaluation of several attribution methods for ICS anomaly detection across a large set of attacks and anomaly-detection models. We first implement five process-level anomaly-detection models for ICS. Next, we implement eight attribution methods for ICS anomaly detection and adapt them to attribute time-series reconstruction errors. We evaluate these anomaly-detection models and attribution methods across two datasets from prior work that contain real ICS attacks [7], [27] and a newly generated dataset of synthetic anomalies designed to increase anomaly diversity, created with an open-source ICS simulator [14].

We make the following contributions:

- In contrast to what is suggested by prior work, we find that ranking features by raw reconstruction error performs poorly for attributing ICS anomalies when evaluated over a broad set of attacks and anomaly-detection methods.
- We find that ML-based attribution methods outperform raw-error rankings, but only when the input to the attribution method coincides with the start of the anomaly.
- We evaluate attribution methods along manipulation properties and find that low-magnitude, categorical-actuator-based manipulations are most difficult to attribute.
- We show that an ensemble method that combines the outputs of multiple attribution methods outperforms all individual attribution methods.
- We provide recommendations for researchers and practitioners when designing and deploying attributions for ICS anomaly detection.
- To support further research on attributing ICS anomalies, we create (i) an open-source library of attribution methods for reconstruction-based, time-series ICS anomaly-detection models[1], (ii) a modified ICS simulator that performs well-defined sensor/actuator manipulations[2], and (iii) a dataset of 286 synthetic ICS anomalies for testing[3].

II. Background and related work

In this section, we discuss background from prior work: ICS attacks and datasets (Sec. II-A), ICS anomaly-detection methods (Sec. II-B), and attribution methods (Sec. II-C). We then discuss prior evaluations of attributions for ICS anomaly detection (Sec. II-D) and related, alternative techniques to attribution methods for ICS anomaly detection (Sec. II-E).

A. ICS attacks and datasets

ICS are interconnected systems that govern a physical, industrial process. ICS collect information from the process through *sensors* and control the process through commands sent to *actuators* [60].

When attackers obtain access to an ICS, they can manipulate the sensor and actuators values sent throughout the network, causing the ICS to react improperly [15], [44]. These attacks can be performed stealthily through a false-data-injection attack, which bypasses existing ICS controls such as state-estimation and programmable control logic [44]. We use an attacker model proposed in prior work [15], where an adversary replaces sensor/actuator readings with curated values over a period of time.

When ICS controllers observe and respond to readings that differ from their true value (e.g., a sensor value for the water tank level that does not represent reality), the effect is propagated through the system, achieving the attacker's harmful objective (e.g., overflowing a water tank).

We evaluate on two public water treatment datasets [7], [27]. These datasets include examples of manipulation-based ICS attacks, with descriptions of which sensors/actuators are manipulated, how they are manipulated, and the intended attack objective. In evaluating against these attacks, we assume that the attacks would genuinely cause the intended harm.

B. ICS anomaly-detection methods

In this work, we assume that the ICS is partially secured with anomaly detection, which predicts anomalies based on real-time observation of process values. We further assume that the ICS operator has full *white-box* access to the model: they are able to fully observe its inputs, outputs, and parameters. We evaluate attributions from two types of anomaly detection: statistical anomaly-detection methods and deep-learning-based anomaly-detection methods.

1) Statistical methods: PASAD uses a departure score to detect anomalies [10]. For each feature, the time-series signal is compressed into a lower-dimensional signal subspace. At test time, new inputs are projected onto the signal subspace, and the distance from the subspace centroid is used as the anomaly score; scores that exceed a threshold are predicted as anomalies.

An auto-regressive model (AR) is a linear model that predicts future process values from previous values [31]; each feature is modeled independently. Prior work has applied AR to ICS process data for anomaly detection by using a cumulative sum of prediction errors [64].

2) Deep-learning-based methods: In prior ICS anomaly detection work, deep-learning-based models are based on *unsupervised learning* [23], [39], [72]: rather than directly classifying an attack, the model predicts the next ICS state and prediction error is used to detect anomalies. Unsupervised learning is preferred to supervised learning in the ICS setting, as supervised learning requires explicit attack labels, and ICS attack data is rare and difficult to generalize [11].

At time t, given an input window of the previous h states $(x_{t-h}, ..., x_{t-1})$, the model predicts the next state $\hat{x}_t$. The predicted state $\hat{x}_t$ is compared with the observed state x_t; an anomaly is declared if the total difference exceeds a threshold.

Several model architectures have been evaluated on a variety of ICS datasets in prior work [23], [39], [72]; prominent examples include convolutional neural networks (CNNs) [39], gated-recurrent-unit networks (GRUs) [23] and long-short-term-memory networks (LSTMs) [72]. CNNs learn patterns through one-dimensional convolutional kernels, which perform convolutions across the time dimension and can capture temporal patterns in the sensor and actuator values [39]. GRUs and LSTMs are similar, but do not use fixed-size convolutional kernels [42]. GRU cells are instead trained with the capability to update or reset system states, while LSTM cells further include memory units that maintain states over time. In theory, GRUs and LSTMs are more complex than CNNs and can learn longer-term patterns from data [42].

[1] https://github.com/pwwl/ics-anomaly-attribution

[2] https://github.com/pwwl/tep-attack-simulator

[3] https://doi.org/10.1184/R1/23805552

Table I: A summary of the attribution methods used in this work. Each attribution method is listed with its type, whether it requires a baseline reference, and whether it uses randomness.

Method	Type	Needs baseline?	Randomness?
Counterfactual (CF-Add/CF-Sub) [17]	Black-box	Yes	No
LIME [51]	Black-box	No	Yes
SHAP [45]	Black-box	No	Yes
LEMNA [29]	Black-box	No	Yes
Saliency Map (SM) [56]	White-box	No	No
Smoothed Gradients (SG) [59]	White-box	No	Yes
Integrated Gradients (IG) [61]	White-box	Yes	No
Expected Gradients (EG) [21]	White-box	Yes	Yes

C. Attribution methods

Given a model, its input, and its prediction, attribution methods estimate the impact of each input feature on the prediction. A canonical example is in the image domain: given an image classification model, an input image, and a predicted label, attribution methods assign a score to each pixel in the input image that estimates its importance to the prediction [61], [21]. Broadly, two types of attribution methods exist: black-box methods, which use model queries, and white-box methods, which use internal model gradients. Table I lists the attribution methods used in this work and their properties.

1) Black-box attribution methods: Black-box attribution methods use repeated model queries to estimate model behavior; a set of input-output pairs is generated by locally perturbing the input and observing the corresponding model output. LIME uses these input-output pairs to fit a linear model and uses feature coefficients as attribution scores [51]. SHAP also fits a local model, but instead uses feature Shapley values as attribution scores [45]. LEMNA focuses on security-relevant applications [29]; it uses the input-output pairs to fit a combined fused Lasso regression and Gaussian mixture model and uses its feature coefficients as attribution scores.

Although LIME, SHAP, and LEMNA are designed for single-output classification and regression tasks, prior work uses SHAP for ICS anomaly detection by attributing the highest-error feature [9], [34]; we apply LIME, SHAP, and LEMNA to ICS anomaly detection with the same technique.

2) White-box attribution methods: White-box attribution methods use gradients on model parameters to determine how predictions change with respect to a given input. A variety of prior work improves white-box attribution methods by modifying how the gradients are computed and how the inputs to the gradient function are chosen; these techniques have each been empirically shown to improve attribution quality in other domains [21], [56], [59], [61].

We adapt four white-box methods from prior work (saliency maps [56], smoothed gradients [59], integrated gradients [61], and expected gradients [21]) for deep-learning-based anomaly-detection models and evaluate them on ICS attacks.

D. Prior work in attributing attacks on ICS

In this section, we summarize prior work that evaluates attributions of ICS anomaly detection. Although prior works have qualitatively evaluated explanations of ICS anomaly detection models with visualizations [18], [30], [37], [62], few prior works have *quantitatively* evaluated attributions by directly mapping anomaly scores produced from anomaly-detection models to the features manipulated in the attack.

We identify two prior works that employ quantitative evaluation: Kravchik and Shabtai evaluate “attack detection localization” of a CNN trained on SWaT by directly identifying if high-error features correspond to manipulated features (found in Appendix C in their work [39]), and Hwang and Lee evaluate a similar LSTM-based setup on a different dataset (found in Section VII-C1 of their work [34]).

Kravchik and Shabtai evaluate the attributions of a CNN-based anomaly detection model on the SWaT dataset and find that, when using only a few of the highest ranked raw-error features, eight out of ten attacks are correctly attributed to the original attack location [39]. However, the SWaT dataset contains 32 attacks, and an evaluation for the rest of these attacks is not performed. Hwang and Lee evaluate the attributions of an LSTM-based anomaly detection model [34], computing attributions with SHAP [45]. Evaluating on only two attacks, they find that using the three highest-error features can correctly identify which feature was manipulated.

In this work, we expand greatly on prior work by evaluating attributions on 156 manipulations from three datasets and evaluating attributions across a variety of timing and attack scenarios.

E. Alternatives to attribution for ICS anomaly detection

In this section, we present alternative approaches to ICS anomaly detection and attribution, while discussing their advantages and drawbacks.

Physics-based anomaly detection: Physics-based (also called model-based) anomaly detection is an alternative to the data-driven (also called model-free [20]) approaches used in this work [26]. Physics-based anomaly-detection models the physical process with a set of equations and is best suited for systems that closely follow the laws of physics, such as robotic motion [16], [50] or electric power grids [44], [57]. Properly implementing such approaches requires a strong understanding of the physical system, and attributions are less likely to be needed for fault diagnosis. In this work, we assume that operators do not have strong system knowledge and rely on data-driven anomaly-detection methods; attributions can therefore help operators diagnose anomalies (as suggested by our operator survey in Sec. V).

Rule-based anomaly detection: In rule-based anomaly detection [6], [22], [38], rules or invariants are used to describe benign ICS behavior, and a violated rule causes an anomaly to be declared. Designing rule-based anomaly detection requires process knowledge and engineering effort [46], so automated approaches have emerged to address this difficulty [6], [22], [43]. Engineered rules can also serve as features for ML-based ICS anomaly-detection [39], [43], [53]: when rules are sufficiently complex, their values can be used as explanations or indicators of compromise [6], [13]. Rule-based and ML-based anomaly-detection have been compared for ICS [22], [36], [68]; this work focuses on exploring ML-based anomaly-detection attributions for ICS that rely on such techniques.

Fault isolation: Fault isolation models predict attack types directly [25], [71], turning an anomaly-detection task into a classification task [35], [54]. Fault isolation requires an explicit definition of the types of faults expected in the system, which can be difficult to acquire [67]. Furthermore, applying fault isolation to ICS anomaly detection requires diverse attack examples across features and attack strategies for training. In this work, we assume that this approach is infeasible and would not scale to larger, complex ICS.

Comparing attribution and explanations for security tasks: While *attribution* refers to identifying input features responsible for a specific output, *explanation* more generally refers to understanding how a model behaves [48]. For example, explanation methods have been used in security-relevant ML domains to generate training data for model fine-tuning [33], to automatically detect bias [36], and to detect and explain concept drift [32], [69]. Explanation methods can be applied to our anomaly-detection models to improve detection accuracy.

In this work, we focus on using attribution to identify the manipulated feature in a specific ICS attack. Since the attacks in our datasets contain ground-truth labels for which sensors or actuators were manipulated, we quantitatively measure how accurate attributions are for this task.

Intrusion Detection Systems (IDS): ICS anomaly-detection systems are closely related to intrusion detection systems (IDS): both are given a time-series sequence of data, and both identify if and when anomalous behavior occurs [12], [47]. State-of-the-art ML-based approaches for IDS and ICS anomaly detection are similar; they commonly use time-series-based, unsupervised anomaly detection, as labeled ground-truth data is expensive to obtain in these domains [11].

However, ICS anomaly detection is distinct from IDS in the types of features processed by their ML models, introducing new challenges and requiring unique solutions. First, ICS anomaly detection observes a system with semantics governed by physical processes and process control logic; IDS are generally designed for network or host data, where relationships between features are often not as strong. Second, since features directly correspond to physical components in ICS anomaly detection, their attributions can be used as suggested locations for operator investigation without additional interpretation. As ML-based IDS often operate on engineered features from network or host data, solutions that help operators interpret their predictions are needed [36].

III. Methodology

In this section, we describe our methodology, which is composed of several steps; Fig. 1 shows our overall methodology in detecting and attributing ICS anomalies.

First, we prepare anomalous data for evaluating anomaly detection and attribution—Sec. III-A describes the datasets used in our work, spanning both publicly collected and newly generated datasets. Second, we train ICS anomaly-detection models, closely following techniques from prior work, as described in Sec. III-B. Third, we compute attributions of anomalies, using baseline methods from prior work (raw-error ranking) and ML-based attribution methods; Sec. III-C describes how we adapt attribution methods to account for the time-series and unsupervised aspects of ICS anomaly detection. Finally, we sort, score, and average attributions to produce an overall ranking of features for investigation; Sec. III-D describes our evaluation metric that captures attribution accuracy over a broad set of anomalies.

A. Datasets used for training and evaluation

We evaluate against two groups of anomalies: (1) real attacks found in public ICS datasets [7], [27] and (2) synthetic anomalies created with an open-source ICS simulator [14].

1) Real attacks: SWaT [27] and WADI [7] are two public datasets of time-series sensor and actuator data collected from real water treatment and distribution systems in Singapore; they are commonly used for training deep-learning-based anomaly-detection models [18], [24], [39], [49], [72]. SWaT and WADI each contain data from two separate system traces: one trace is collected from a benign execution of the system (for training anomaly-detection models); and the other trace is collected from an execution containing several attacks performed sequentially (for evaluating attack detection and attack attribution). Each attack is performed by the system operator: the value of one or more sensors or actuators is manipulated for a fixed duration, and the resulting response from the physical ICS is recorded. Each attack's start time, end time, and location (which sensors/actuators are manipulated) are labeled. In total, the SWaT and WADI datasets contain 47 attacks for testing: 32 in SWaT and 15 in WADI. These datasets also contain anomalies where multiple features are manipulated simultaneously; when evaluating attributions, we consider each manipulation independently. Across the 47 attacks in our datasets, 67 manipulations (43 in SWaT, 24 in WADI) are performed, forming the set of *real attacks*.

2) Synthetic anomalies: To further increase anomaly diversity for our attribution evaluation, we created an additional dataset of anomalies by manipulating a simulated ICS. We use a public MATLAB 7.0 simulator of the Tennessee Eastman process (TEP) [14], an anonymized chemical process [19].

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], [40]: for a feature j, a benign sequence $x_j(t)$ is replaced with a manipulated sequence $x'_j(t)$ for a time period T_a.

$$\tilde{x}_j(t) = \begin{cases} x_j(t) & \text{for } t \notin T_a \\ x'_j(t) & \text{for } t \in T_a \end{cases}$$

Using the modified simulator, we systematically perform ICS feature manipulations to generate a set of *synthetic anomalies*. For each anomaly, we execute a 40-hour TEP simulation, manipulate a chosen sensor/actuator, and record the resulting system states. For every sensor and actuator in TEP, we simulate four anomalies with different magnitudes, creating 89 anomalies[4].

[4] 11 out of 100 manipulations triggered a shutdown sequence, causing the MATLAB simulator to exit and preventing data collection.

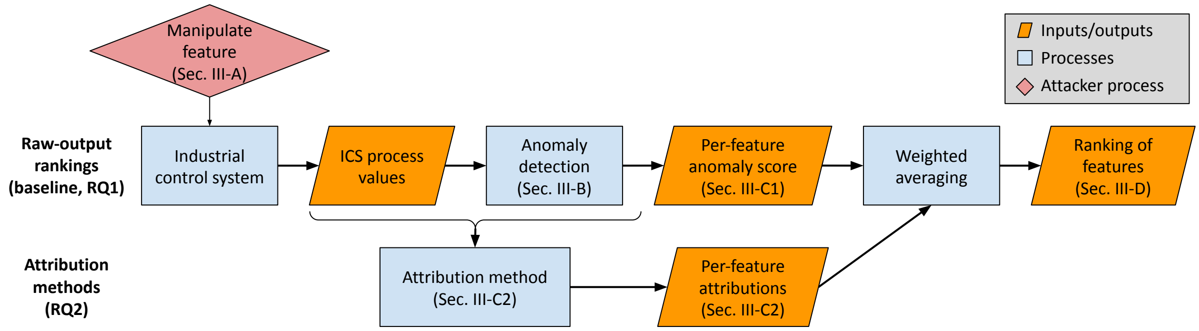

Manipulate feature (Sec. III-A)
Inputs/outputs
Processes
Attacker process
Raw-output rankings (baseline, RQ1)
Industrial control system
ICS process values
Anomaly detection (Sec. III-B)
Per-feature anomaly score (Sec. III-C1)
Weighted averaging
Ranking of features (Sec. III-D)
Attribution methods (RQ2)
Attribution method (Sec. III-C2)
Per-feature attributions (Sec. III-C2)

Figure 1: In this work, we describe and evaluate each step when attributing an ICS attack. Raw-output rankings (top) are described in Sec. III-C1 and are the baseline method from prior work. We introduce ML-based attribution methods (bottom) as an alternative in Sec. III-C2. Outputs/attributions are averaged and sorted to produce an overall ranking of suggested features for investigation, described in Sec. III-D.

Table II: A summary of the manipulations used for evaluation, across a set of real attacks from prior work (SWaT, WADI) and a set of synthetic anomalies generated with a public simulator (TEP).

Dataset	Magnitude	Location	Total
SWaT	0.06–36.31 std devs	24 sensors 19 actuators	43
WADI	0.5–91.00 std devs	16 sensors 8 actuators	24
TEP	2–5 std devs	56 sensors 33 actuators	89

Although these anomalies are generated synthetically, they simulate physically realizable anomalies. For each manipulation, the data collected from non-manipulated features were produced as the output of the physical process (i.e., the chemical process) responding to the manipulation. In cases where manipulations produced runtime errors in the MATLAB simulation, we excluded them from our dataset. Maintaining physical realizability is important to ensuring that the executed attacks are possible, and is essential when evaluating ML-based approaches in other security contexts, such as face recognition [55] and malware detection [63].

Though the anomalies in this dataset do not necessarily correspond to an intentional ICS attack outcome, they are genuine statistical anomalies (95th percentile or higher event) that share common properties (e.g., manipulation patterns and magnitudes) with the manipulations observed in the real attack dataset. We use the synthetic anomalies to support a systematic analysis of the relationship between manipulation properties and attributions.

3) Defining manipulation properties: Each anomaly is defined by its manipulation magnitude and the type of feature attacked; the anomalies contained in all datasets are summarized in Table II. In Sec. IV-C, we identify properties that significantly affect attribution accuracy and which attribution methods are optimal.

We define the *manipulation magnitude* using the difference in standard deviations between feature j's benign distribution and its replaced value:

$$\text{magnitude} = \frac{|\max(x'_j(t)) - \text{ mean}(x_j(t))|}{\text{stddev}(x_j(t))}$$

Attacks in SWaT and WADI are performed with manipulations that span a wide magnitude range, from small manipulations within the benign distribution (magnitude of 0.06) to large manipulations outside the benign distribution (magnitudes over 35). When generating synthetic anomalies, we perform manipulations at four different magnitudes: +2, -2, +3, and +5. Our synthetic anomalies are performed within the distribution of magnitudes observed in the real attack dataset, which has an average magnitude of 3.25 standard deviations.

We also define attacks by the *type of feature* that is manipulated. Features in TEP are grouped into three categories: *actuators*, which directly control the chemical process; *sensors*, which are read by controllers to compute future actuator values; and *out-of-loop features*, which do not impact the ICS process. For TEP, we only perform manipulations on sensors that result in changes to the physical process. SWaT and WADI provide documentation for each feature and each attack: we manually verified that each manipulation was executed as described and affects the physical ICS process.

B. Implementing ICS anomaly detection

In this section, we describe our implementation of statistical and deep-learning-based anomaly-detection models for ICS, closely following the methodology from prior work.

1) Statistical anomaly detection: We implement two statistical anomaly-detection methods: PASAD [10] and AR [31].

We use the open-source PASAD implementation by Aoudi et al. [5] and tune it for each dataset in our use case. PASAD is parameterized by the training length N, the input window length (called lag) L and the statistical dimension r. We refer to the anomaly-detection methodology and configurations by Aoudi et al. [10], using the same default parameters for the SWaT and TEP datasets. Since WADI is based on the SWaT system, we opt for the same parameter values for WADI and SWaT. For SWaT and WADI, our parameters are: $N = 30000$, $L = 5000$ and $r = 10$; for TEP, our parameters are: $N = 10000$, $L = 5000$ and $r = 16$.

The original AR implementation is in C++ [4], so we opt to implement a linear model on our own in Python. AR is parameterized by p, the number of prior states used in the linear model. Based on the default settings, we use $p = 10$, and train our linear models with the Adam optimizer.

2) Deep-learning-based anomaly detection: A variety of prior work performs hyperparameter tuning across model architectures to find optimal deep-learning-based anomaly-detection models for ICS [24], [39], [72]; *this is not the focus of our work*. Nevertheless, we perform a best-effort training of anomaly-detection models for attributions.

We evaluate across three model architectures: convolutional neural networks (CNNs) [39], gated-recurrent-unit networks (GRUs) [23], and long-short-term-memory units (LSTMs) [49], [72]. To best compare across model architectures, we use a similar model size for each architecture: a 2-layer, 64-unit, 50-length-history model is trained for each combination of dataset (SWaT, WADI, TEP) and architecture (CNN, GRU, LSTM), with the default Adam optimizer. For each training dataset, 80% of the benign dataset is used for training and 20% of the benign dataset is used for validation. As suggested by prior work [24], we implement early stopping, which halts training when the validation loss stops decreasing. Finally, we test the anomaly-detection models against the manipulations from SWaT, WADI, and TEP. The results are shown in Table IX in Appendix B. Similar to prior work, we find that the size of the underlying reconstruction model has a small effect on detection accuracy [24]; each model's benign validation error is below 0.25.

C. Attribution methods for ICS anomaly detection

After implementing ICS anomaly-detection models (described in Sec. III), in this section we describe how we adapt attribution methods for anomaly-detection outputs.

1) Baseline attribution method from prior work: Our baseline attribution method uses the raw, per-feature anomaly scores produced by each model as the attribution, as suggested in prior evaluations of ICS anomaly-detection attribution [39].

For AR and deep-learning-based anomaly detection models, we use the per-feature prediction error (i.e., before taking the average for MSE) between input x_j and its prediction $\hat{x}_j$ as the anomaly score. Each error s_j corresponds to the anomaly score for an ICS feature j: $s_j = (x_j - \hat{x}_j)^2$.

With PASAD, each feature's input x_j produces a departure distance that represents its deviation from normal[5]; we use the departure distance as the anomaly score: $s_j = (\tilde{c} - Px_j)^2$.

2) Adapted attribution methods: We next describe how attribution methods can be adapted for ICS anomaly detection. Each attribution method is given a trained anomaly-detection model and an input of interest, and produces an attribution for each input feature. We briefly describe our adaptations to attribution methods; additional implementation details can be found in Appendix B.

Counterfactuals: Given an input and a baseline, a counterfactual attribution is computed by changing feature values and measuring the change in a quantity of interest [65]. We use MSE as the quantity of interest for anomaly detection. Based on prior work [17], we implement two counterfactual attribution methods: an additive method that adds values to the baseline; and a subtractive method that subtracts values from the provided input.

LIME [51], SHAP [45], and LEMNA [29]: LIME, SHAP, and LEMNA 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 public LIME [2] and SHAP [3] libraries maintained by their original authors. We implement LEMNA based on its published description [29] 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 LIME, SHAP, or LEMNA attributions. Thus, we adapt the anomaly-detection task as a single-output regression task to comply with the design and API of LIME, SHAP, and LEMNA.

Saliency maps [56]: The saliency map uses internal model gradients as the attribution. 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. For a traditional classification model, the quantity of interest is a given class label; for our anomaly-detection models, the quantity of interest is the MSE. Thus, for a given model, the saliency map computes each input feature's influence on increasing the MSE.

Other white-box variants: Prior work has found that saliency maps are sensitive to small input changes; in response, several extensions to saliency maps have been proposed [21], [59], [61]. SmoothGrad adds random noise to the input before computing the saliency map to reduce variance [59]. Integrated gradients replace the quantity of interest with a path integral to produce more meaningful outputs [61]. Expected gradients reduce the variance of integrated gradients by sampling inputs from the training dataset [21]. We discuss the detailed implementation of each white-box variant in Appendix B. We evaluate the white-box variants in Appendix C but find that they do not outperform the saliency map when applied to ICS anomaly detection models.

D. Evaluation metric for attributions: AvgRank

Prior work that evaluated explanations of ICS anomaly detection uses a mix of qualitative evaluations of visualizations and analyses of individual attacks [18], [34]. To quantitatively compare attribution methods over full datasets, we propose the metric *AvgRank*. Across a set of ICS attacks, AvgRank represents the average ranking of the manipulated feature when features are ranked by their attributions.

For a given anomaly's attribution s, we sort each feature's attribution s_j in descending order and identify the placement of the manipulated feature j'. Since the analysis in this paper is across three ICS of varying dimension, we normalize rankings by dividing the placement by the number of features in the evaluated ICS dataset. We then report the average placement across all anomalies for a given attribution method.

This produces the AvgRank: a score $\in [0, 1]$ (lower is better) that represents the average proportional ranking of attacked features when identified by an attribution method. In other words, an attribution method with an AvgRank of 0.2

[5] Details on how to compute projection matrix P and subspace centroid $\tilde{c}$ can be found in the original publication [10]

Table III: 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% overflow attacks are identified by the highest score.

	Total	Top-1	Top-5	Top-10	AvgRank
AR	128	10 (8%)	30 (23%)	45 (35%)	0.383
PASAD	130	24 (18%)	43 (33%)	61 (47%)	0.304
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

indicates that this attribution method will, on average, place the attacked feature in the top 20% of features.

AvgRank's design is drawn from prior work, which proposed criteria for explanations of ML models when applied to security-relevant tasks [66]. Although these proposed criteria assume a classification task, some of them are relevant for attributions of ICS anomaly detection. First, *descriptive sparsity* requires that attributions identify a small set of features; our proposed use of attributions filters out sensors and actuators in an ICS. AvgRank could be interpreted as the average number of sensors and actuators that would need to be displayed in an anomaly alert to ensure that the manipulated feature is shown. Second, *completeness* requires that attribution methods perform well over a variety of inputs; AvgRank measures performance over a set of anomalies. In Sec. IV-C, we demonstrate the importance of evaluating attributions over diverse datasets by using AvgRank to reveal discrepancies in attribution accuracy across attack properties.

IV. Results: evaluating attributions of ICS anomalies

In this section, we report on the evaluation of two sets of attribution methods for ICS anomaly detection:

- Raw-error ranking: a baseline method that ranks features in descending order by their reconstruction error (described in Sec. III-C1)
- ML-based attribution methods: attribution methods from other ML domains, adapted for ICS anomaly detection (described in Sec. III-C2)

Sec. IV-A describes our evaluation of attribution methods using strategies from prior work: we find that raw-error rankings perform much less well that previously reported, and ML-based attribution methods also perform worse than anticipated.

To better understand why attributions fail, we perform a more detailed analysis of ICS domain-specific characteristics: the timing of the attribution relative to the time the manipulation occurred (Sec. IV-B), and properties of the manipulations (Sec. IV-C). We find that different attribution methods perform best in different situations, and thus we propose an ensemble attribution method and find that it outperforms all individual methods (Sec. IV-D).

A. Assessing prior attribution strategies

In this section, we first explore prior attribution strategies: when an ICS anomaly-detection model raises an alarm, can a raw-ranking of its outputs identify which feature was attacked? And how do ML-based attribution methods compare?

1) Evaluating raw-error rankings: We first describe our results for raw-error rankings: ranking features in descending order by their error (i.e., by the amount they deviate from the predicted value).

Method: We apply raw-error ranking to all detected attacks in all three datasets: 128 attacks for the AR model, 93 attacks for PASAD, 103 attacks for the CNN, 83 attacks for the GRU, and 113 attacks for the LSTM. Table IX (in Appendix B) shows the breakdown of detected attacks.

We find the first detection point for each labeled attack and use the per-feature reconstruction errors at that timestep as attributions. We rank all features by descending attribution and use the rank of the manipulated feature to compute AvgRank (e.g., an AvgRank of 0.25 implies that the manipulated feature is on average, ranked within the top 25%), repeating the process for each anomaly-detection model.

To find the detection points for our deep-learning-based models, we use each model's 99.95-th percentile validation MSE as a threshold, employing a common strategy from prior work [24], [39], [72]. For the AR model and PASAD, we use the 99.5-th percentile validation error, as very few anomalies are detected at the 99.95-th percentile threshold. At test time, an anomaly is detected if any input within the labeled anomaly region produces an MSE above the threshold.

Results: Table III shows the accuracy of the prior attribution strategy for each anomaly-detection method, spanning both statistical methods and deep-learning-based models (CNNs, GRUs, and LSTMs). Although prior work has reported high attribution accuracies (e.g., 80% accuracy within the top few features[6] [39]), we find that raw-error rankings are not as effective as reported; for all models, less than half of all attacked features are correctly identified by the highest-error feature, and at least one quarter of attacked features could not be identified within the top 10 features. Across all three deep-learning-based models, the AvgRank ranges from 0.14 to 0.19; in other words, on average, the manipulated feature is ranked within the top 14–19%, which is far more than the top few, as suggested in prior work.

Although the AR model and PASAD are effective at detecting attacks, their attributions perform far worse than for the deep-learning-based models; their AvgRank is much higher (over 0.3) and over 50% of all attacks are not correctly attributed by the top 10 features. This is likely because AR and PASAD model each feature independently, meaning that they cannot consider inter-feature relationships when performing anomaly detection. Thus, when an ICS feature is manipulated and subsequent components respond to the change, AR and PASAD identify are particularly likely to identify those additional features as anomalous.

2) Evaluating raw ranking of attribution method outputs: Next, we compare raw-error ranking to ML-based attribution methods. To select a group of best-performing attribution methods as candidates, we design and use a synthetic

[6] The number of features selected for investigation varied by attack: from one single feature to as many as four.

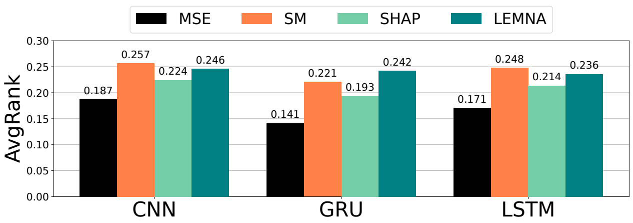


Figure 2: When attributing ICS anomalies at the time of detection, the raw-error feature (MSE) produces a lower AvgRank (lower is better) than all best-performing ML-based attribution methods: the saliency map (SM), SHAP, and LEMNA.

benchmark to systematically compare attribution methods; our benchmark is described in Appendix C. We find that the saliency map (SM), SHAP, and LEMNA are the best-performing attribution methods.

Method: For each detected attack, we find the first detection point using the same methodology as with raw-error rankings. We then find the corresponding model input for the detection point—using the range of data from the 50 timesteps prior to the detection point. The model input is used as input to an attribution method, producing a score for each ICS feature. We use these scores to rank features and compute AvgRank, repeating the process for each combination of deep-learning-based anomaly-detection model and attribution method.

Results: Fig. 2 shows the resulting AvgRank for all attribution methods and deep-learning anomaly-detection models. Although the results of our benchmark (described in Appendix C) suggest that, in theory, ML-based attribution methods outperform raw-error ranking, ML-based attribution methods perform far worse when applied in practice to attack scenarios: for each model architecture, raw-error ranking produces a lower AvgRank than all ML-based attribution methods (0.14–0.19 for raw-error ranking, compared to over 0.2 for most ML-based attribution methods).

Finding 1: When computed at the timestep when the anomaly is detected, attributions based on ranking raw reconstruction errors from anomaly-detection models are less accurate than previously reported, and ML-based attribution methods have similar or worse accuracy.

To better understand why attribution methods fail for ICS anomaly detection when applied to attack scenarios, we identify and investigate two ICS-specific characteristics which make attributions difficult and affect which techniques work best: Sec. IV-B describes the effect of timing on attributions and Sec. IV-C describes how attack properties affect attributions. We identify configurations under which both raw-error rankings and ML-based attribution methods perform better than prior attribution strategies.

B. Effect of detection timing on attributions

ICS anomaly detection is performed over a time-series input, so selecting the best timing for attributions is an important consideration for optimal performance. The data in the selected input window may contain too much noise or too little signal to make an accurate attribution. We observe that there exists a

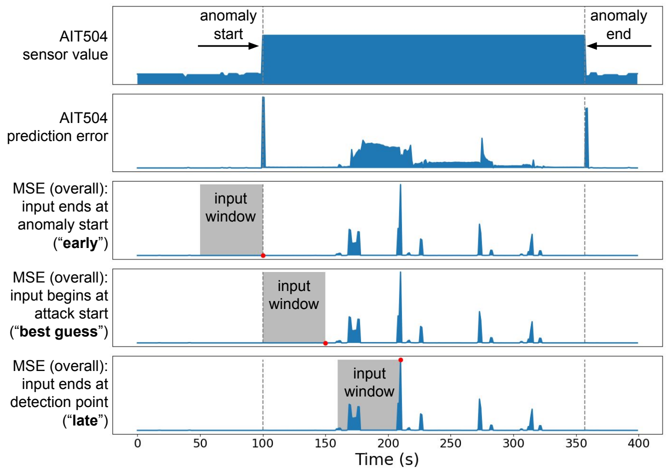


Figure 3: Outputs of a GRU-based anomaly-detection model on SWaT attack #10 are shown: when sensor AIT504 is manipulated (top), its prediction error (2nd) is insufficient to trigger an anomaly. As the ICS responds and the total error increases, the model detects the anomaly over 100 seconds later. From this example, attributions can be computed at three points (shown in red) with corresponding input windows (shown in grey): at the anomaly start (3rd), when the input window coincides with the anomaly (4th), or when the anomaly is detected (5th, bottom).

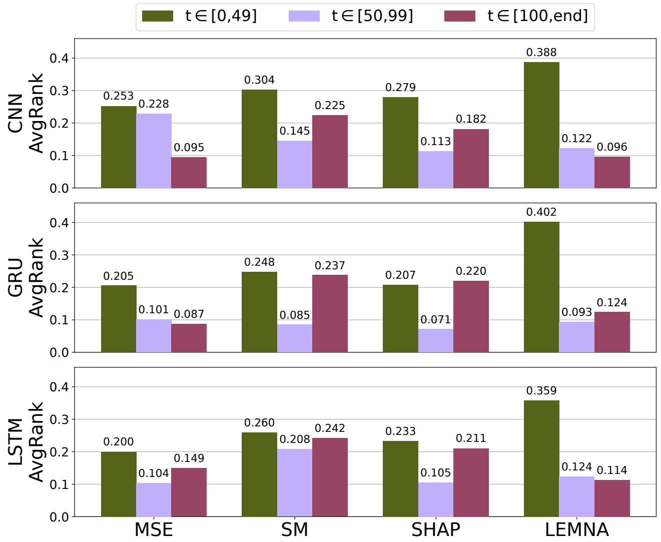


Figure 4: Across all detected attacks, we compare AvgRank (lower is better) across three timing cases, based on the detection time t relative to the start of the anomaly ($t = 0$). Considering that 50 timesteps are used for the model input, we divide attacks based on if $t \in [0, 49]$, $t \in [50, 99]$, or $t \in [100, \text{end}]$. For most cases, the AvgRank is lowest when $t \in [50, 99]$.

Table IV: We categorize each attack by its detection time t relative to the start of the anomaly (considered $t = 0$), dividing into cases where the detection is early ($t \in [0, 49]$), slightly late $t \in [50, 99]$, or very late ($t \in [100, \text{end}]$). For each model architecture, the number of attacks that fall into each case is shown.

	Total	$t \in [0, 49]$	$t \in [50, 99]$	$t \in [100, \text{end}]$
CNNs	103	52	10	41
GRUs	86	38	12	36
LSTMs	113	56	9	48

"best-guess" timing: when ML-based attribution methods are computed at this timing, their accuracy improves.

1) Identifying how timing can affect attribution: To illustrate how timing can affect attribution, Fig. 3 shows the sensor value, sensor prediction error, and the total MSE for a GRU-based anomaly-detection model for the duration of SWaT attack #10. In this attack, a chemical sensor's value (AIT504) is increased to 16, causing a reverse-osmosis sequence to shut down; the GRU model detects the anomaly within two minutes.

If the input window is selected immediately before the detection point ("late", shown in bottom row of Fig. 3), the overall MSE is high: many features have drifted from their expected values in reaction to the attack and will appear anomalous, making it difficult to attribute the attack to the correct feature.

If the input window is selected immediately at the start of the attack[7] ("early", shown in third row of Fig. 3), observing such feature drift can be avoided; however, since our anomaly-detection models rely on historical input, the input window will contain benign signal, complicating attribution.

We observe that, in our dataset, the detection point can vary relative to the start of the anomaly. Given the 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. Table IV shows the number of occurrences for each of the described three cases: all three cases are prominent. We analyze AvgRank across these three cases and show the results in Fig. 4. In most settings, attributions computed within 50 seconds of the anomaly start perform the worst, and attributions computed within 50 to 100 seconds of the anomaly start perform the best.

Finding 2: ICS anomalies vary in when they are detected relative to their start time. Differences in detection timing affect attribution accuracy.

Based on these observations, the ideal timing (for performing attribution) should be sufficiently near the start of the anomaly to avoid observing the original manipulation's subsequent effects; and should be sufficiently after the start of the anomaly, such that the input window contains sufficient manipulated information. Towards achieving these goals, we select an input window that starts at the same time as the anomaly, as shown in the fourth row of Fig. 3 (e.g., given a 50-timestep input window length, we would use the 51st timestep after the anomaly start), and we call the strategy using this input window the "best-guess" timing[8].

2) Comparing timing strategies: Based on the observed differences in attribution accuracy across timing, we evaluate attribution methods across two timing strategies:

- "Practical" timing: when the input window immediately precedes the detection time (used in prior evaluation)
- "Best-guess" timing: when the input window starts at the same time as the anomaly

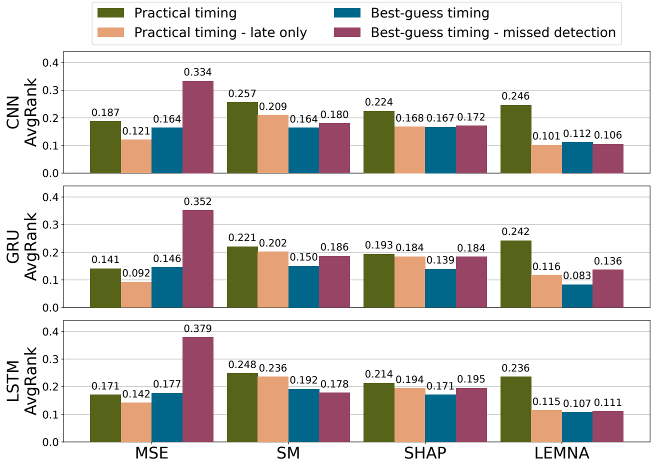


Figure 5: For all datasets, the AvgRank (lower is better) is reported after attributions are computed with different timing strategies: "practical timing", the prior attribution strategy that computes attributions immediately when anomalies are detected, and "best-guess timing", which computes attributions such that the input starts with the anomaly. Two additional variants are reported: practical timing with early detections removed, and best-guess timing for attacks that are not detected by the underlying anomaly-detection model. Results for the CNN (top), GRU (middle), and LSTM (bottom) are shown. In all cases, choosing an alternate timing strategy from the "practical" strategy improves attribution accuracy.

We compare the AvgRank at the best-guess timing to the practical timing, across all datasets (SWaT, WADI, and TEP) and all deep-learning-based model architectures (CNNs, GRUs, LSTMs). When evaluating the best-guess timing for each anomaly, the input is *exactly the same* across attribution methods and models, even if models detect the anomaly at different times.

Fig. 5 shows the AvgRank for different timing strategies. We first compare the best-guess and practical timings ("best-guess timing" vs "practical timing"): when an anomaly is detected, if instead the best-guess timing is used, does AvgRank improve? We find that in 10 out of 12 cases (including all cases with ML-based attribution methods), the best-guess timing outperforms the practical timing. For example, LEMNA improves for all models: 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. Furthermore, when the best-guess timing is used, LEMNA is the best-performing attribution method for all models.

Finding 3: ML-based attribution methods outperform raw-MSE rankings when attributions are computed with inputs beginning at the start of the anomaly.

To analyze the impact of early detections on practical timings, we compare the AvgRank after removing attacks that are detected before the 50th timestep, shown in Fig. 5 ("practical timing" vs "practical timing—late only"). We discuss the results of the CNN; the results for the GRU and LSTM show similar observations. Although early detection of anomalies is clearly a beneficial outcome in practice, it results in worse attributions: removing the early detections improves the MSE AvgRank from 0.187 to 0.121. In all 12 cases, AvgRank improves after removing early detections.

[7] This is a hypothetical solution; in practice the start time is not known.
[8] "Best-guess" since in practice the start time is not known.

3) Separating timing from detection outcome: Finally, we investigate if attribution methods could be useful even in cases where anomaly detection fails. Using the best-guess timing, we compare the AvgRank between anomalies that are detected and anomalies that are missed by the anomaly-detection model, shown in Fig. 5 ("best-guess timing" vs "best-guess timing—missed detection").

For raw-MSE rankings, the performance is drastically worse for anomalies 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. This is expected: when the MSEs are insufficient to detect the anomaly, they are also insufficient to identify the manipulated feature. However, when these same inputs are used with ML-based attribution methods, the AvgRank performs far better for missed attacks. ML-based attribution methods perform approximately as well, regardless of whether the anomaly is detected or not. For example, the CNN LEMNA attribution AvgRank changes from 0.112 to 0.106. One potential implication of this observation is that attribution methods should be computed separately from detection times and detection outcomes; this could potentially be accomplished with a data historian or other post-hoc incident analytics for anomalies that are not detected in real time.

In summary, we found that the timing of attributions has a large impact on attribution accuracy. Although we do not advocate for a specific timing strategy, we would like to highlight that computing attributions at "practical" timings, the strategy most commonly used in prior work [34], [39], does not lead to best attribution outcomes, although it reflects how attribution methods might be used in practice.

C. Effect of attack properties on attributions

Although broad evaluations of attribution methods can reveal general trends, a deeper analysis across attack properties reveals imbalances in attribution accuracy between different attacks. In this section, we investigate how (i) the magnitude of the manipulation used in the attack and (ii) the type of feature attacked affect the accuracy of attribution methods. To enable such analysis, we define all anomalies from our three datasets (SWaT, WADI, and TEP) along common properties (as described in Sec. III-A). We compute attributions at their best-guess anomaly timing (as described in Sec. IV-B) and perform statistical tests to quantify each dimension's effect on AvgRank. The results of this analysis are shown in Table V.

1) Magnitude: We compare 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 log of the magnitude for our analysis. The first column of Table V 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.

For all methods, we 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 [28], so attribution methods that perform well on these manipulations may not be needed. An important area of future work would be to design effective attribution methods that are robust to low-magnitude manipulations.

2) Feature type: Sensors and actuators are fundamentally different: actuators induce changes in the industrial process, while sensors provide feedback from the industrial process. Some actuators are also encoded as categorical variables (e.g., a valve in SWaT is ON (1) or OFF (0)), while all sensors in our datasets are continuous-valued. Although sensors and actuators differ in context and representation, current statistical and deep-learning-based anomaly-detection models treat these features equally as raw-valued features.

We analyze if whether a sensor or actuator was manipulated affects AvgRank. Raw-error rankings perform significantly better for sensor-based attacks while ML-based attribution methods perform better on actuator-based attacks. We empirically explore this difference by using a one-way ANOVA test to compare the AvgRank distributions when separating 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 V.

For raw-error ranking (MSE), the AvgRank for sensors ranges from 0.152 to 0.162, whereas the AvgRank for actuators is always above 0.306. For ML-based attribution methods, the findings are different: 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 attribution methods may be able to capture relationships that connect sensors with their corresponding actuators, beyond what can be found by the raw-error ranking. We also find that attribution methods perform worst on categorical-valued actuators (AvgRank above 0.248), likely because attribution methods compute attributions for categorical-valued features as if they were continuous-valued features.

In general, anomaly-detection models would likely benefit from modelling sensors and actuators differently when computing attributions, in ways that consider (i) interdependencies between sensors and actuators and (ii) categorical variables as states, rather than continuous values. Attacker models for sensors and actuators are also different; prior work has argued that actuator attacks require more ICS knowledge and are more difficult to execute in practice [64].

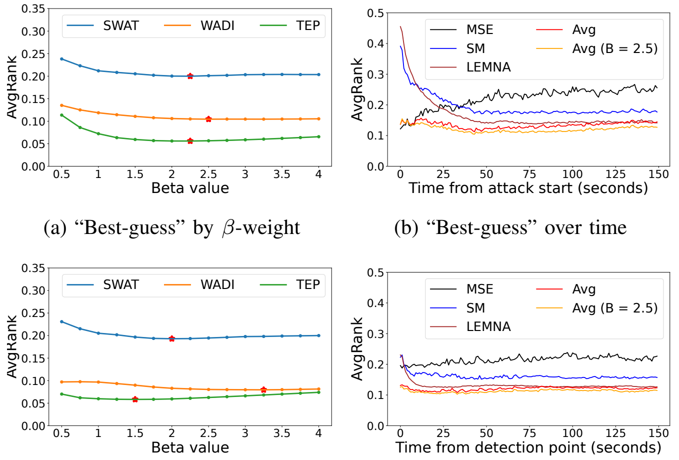
3) Effects of other attack strategies: Beyond manipulation magnitude and feature type, we also explore if attackers can use alternate strategies to make attributions less accurate. We find that attacking multiple features simultaneously or using stealthier manipulation patterns can bypass attributions, assuming increased attacker capabilities. We describe the details and results for these experiments in Appendix D.

D. Evaluating ensembles of attribution methods

Different attribution methods are 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 (Sections IV-B–IV-C). Thus, in this section, we design an ensemble of attribution methods to combine the strengths of both types of attribution methods, by using a weighted average over

Table V: We perform three statistical tests across our attribution results under “best-guess” timing: we compare the effect on AvgRank from manipulation magnitude (left), the type of feature attacked (middle), and whether the attack is multi-point (right). Results with p-value below 0.016 (applying Bonferroni correction) are bolded. Raw-error rankings (MSE) perform better on high-magnitude, sensor-based attacks, whereas attribution methods perform better on high-magnitude, actuator-based attacks. This analysis suggests that no attribution method is always optimal across a variety of attacks.

Attribution Method	Model	Magnitude (Pearson)		Sensor vs actuator (continuous) vs actuator (categorical) (ANOVA)			Single-point vs multi-point (ANOVA)		
		Corr.	p-value	AvgRank	F(2, 153)	p-value	AvgRank	F(1, 154)	p-value
MSE	AR	-0.214	**p = 0.007**	0.227 vs 0.486 vs 0.244	9.02	**p < 0.001**	0.300 vs 0.232	1.18	p = 0.279
	CNN	-0.336	**p < 0.001**	0.162 vs 0.306 vs 0.330	14.87	**p < 0.001**	0.199 vs 0.304	4.60	p = 0.034
	GRU	-0.445	**p < 0.001**	0.154 vs 0.385 vs 0.361	17.95	**p < 0.001**	0.224 vs 0.288	1.75	p = 0.189
	LSTM	-0.399	**p < 0.001**	0.152 vs 0.362 vs 0.360	13.21	**p < 0.001**	0.224 vs 0.265	0.65	p = 0.421
SM	CNN	-0.430	**p < 0.001**	0.176 vs 0.059 vs 0.281	13.28	**p < 0.001**	0.160 vs 0.205	1.72	p = 0.192
	GRU	-0.605	**p < 0.001**	0.160 vs 0.050 vs 0.330	21.67	**p < 0.001**	0.156 vs 0.204	1.79	p = 0.183
	LSTM	-0.558	**p < 0.001**	0.197 vs 0.047 vs 0.329	19.55	**p < 0.001**	0.172 vs 0.245	3.71	p = 0.056
SHAP	CNN	-0.497	**p < 0.001**	0.176 vs 0.059 vs 0.275	13.89	**p < 0.001**	0.151 vs 0.231	5.86	p = 0.017
	GRU	-0.595	**p < 0.001**	0.149 vs 0.054 vs 0.325	24.31	**p < 0.001**	0.141 vs 0.225	6.47	p = 0.012
	LSTM	-0.513	**p < 0.001**	0.181 vs 0.039 vs 0.333	23.39	**p < 0.001**	0.159 vs 0.243	5.41	p = 0.021
LEMNA	CNN	-0.544	**p < 0.001**	0.085 vs 0.034 vs 0.291	26.82	**p < 0.001**	0.070 vs 0.252	39.06	**p < 0.001**
	GRU	-0.537	**p < 0.001**	0.084 vs 0.034 vs 0.278	26.76	**p < 0.001**	0.067 vs 0.252	46.80	**p < 0.001**
	LSTM	-0.523	**p < 0.001**	0.084 vs 0.034 vs 0.248	27.23	**p < 0.001**	0.070 vs 0.248	40.55	**p < 0.001**



(a) “Best-guess” by β-weight

(b) “Best-guess” over time

(c) “Practical” by β-weight

(d) “Practical” over time

Figure 6: Attribution averaging is evaluated on CNNs at two timings (“best-guess” on top, “practical” on bottom) and in two ways. On left, 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. On right, across datasets, different β values are used when performing a weighted average, across all six settings, $\beta \in (1.5, 3.25)$ are optimal (red star).

attributions from multiple methods. We find that this ensemble outperforms all individual attribution methods.

Our ensemble of attribution methods computes attributions differently for sensors and actuators; the ensemble uses a raw average of attributions for sensors and a β-average of attributions for actuators, where β represents the relative weight of ML-based attribution methods. We compute our ensemble over three attributions: the normalized MSE, the normalized SM attribution, and the normalized LEMNA attribution.

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

We compare our ensemble of attribution methods to the raw average of attributions: the left half of Fig. 6 shows the AvgRank for our ensemble with different β-values; we compare averaging strategies for CNNs using the best-guess (Fig. 6a) and practical (Fig. 6c) timing strategies (described in Sec. IV-B). For all datasets and timing strategies, the best-performing value of $\beta \in [1.5, 3.25]$ (i.e., higher weight for ML-based attribution methods for actuators) outperforms the raw average (i.e., when $\beta = 1$). We also compare our ensemble to the raw average with GRUs and LSTMs; in all cases the best-performing $\beta \geq 1$, which shows that our ensemble outperforms the raw average.

We next compare our ensemble attribution method to individual, best-performing attribution methods: the raw-error ranking (MSE), SM, and LEMNA. Fig. 6b shows the AvgRank for various attribution methods (individual and ensemble) for the CNN model over 150 timesteps, beginning with the start of the anomaly. Similar to what was found in Sec. IV-B, we find that ML-based attribution methods are initially less accurate, but their accuracy improves over time; conversely, a ranking of raw errors is initially accurate but becomes less accurate over time. Our ensemble attribution method combines strengths of each individual method and produces the lowest AvgRank over most timesteps.

Our ensemble attribution method also outperforms all individual attribution methods when used with a practical timing strategy (Fig. 6d), showing that it can also be used in practice when ground-truth timing is not known. This finding holds when our evaluation is repeated with GRUs and LSTMs.

Finally, we determine what the best-performing configuration for our ensemble attribution method is in practice: Fig. 6d shows that when using our ensemble attribution method with $\beta = 2.5$, and attribution are computed between 25 and 50 seconds after the anomaly is detected, the AvgRank is lowest. Thus, we observe this configuration of our ensemble attribution method to be the best-performing attribution method across all attribution methods evaluated in this work.

Finding 4: An ensemble of attribution methods outperforms all individual attribution methods at identifying which feature was manipulated in an ICS anomaly.

V. Survey: ICS operator perceptions of attributions

Our experiments showed that attribution methods do not achieve perfect accuracy: the feature with the highest attribution score was actually the manipulated feature for less than 39% of all attacks (see Sec. IV-A). This raises the question: are imperfect attributions for anomaly detectors useful to ICS operators?

Concurrently with our experiments, we conducted a preliminary survey of ICS operators to better understand whether and how attributions would be helpful for responding to anomalies. We sought operators' perspectives on the following questions:

1) How do ICS operators respond to anomaly-detection alerts and how would attributions fit into their workflow?
2) Assuming imperfect attribution performance, would operators find it more useful to be shown fewer features, but with a higher chance of omitting the feature that caused the anomaly; or more features, with a lower chance that the manipulated feature will be missed?

Survey: First, we asked participants to describe the type of ICS they have experience with and their role in operating ICS. Next, to surface how operators integrate anomaly detectors into their workflows, we asked participants what kinds of anomaly detectors they have experience with, the benefits and challenges of using them, and what their role is in diagnosing the root causes of an anomaly.

Then, we asked participants to evaluate the tradeoff between the number of reported features and attribution accuracy. We showed participants a sample output for an attack in the SWaT dataset: a subset of the sensors and actuators, their values, and their attribution scores (in descending order). We then asked the participants to rate on a five-point Likert scale how useful the output would be in an attack scenario. We varied the number of features shown in the output, between the top two, five, ten, 20, or all 34 features[9]. We also varied the error rate (the percentage of attacks where the attacked feature is not included in the subset of features shown in the output) between a "low", "medium", and "high" level, which changes depending on the number of features shown (See Table VIII). The error rates were based on our initial estimates of attribution accuracy; later we observed that the "high" error rate roughly matches the empirical error rate of raw-error rankings for the LSTM-based detection model using practical timings on SWaT, and the "medium" error rate roughly matches the empirical error rate of the best-performing LSTM-based ensemble attributions on SWaT. Lastly, we asked participants to explain the reasoning behind their ratings, and how they would integrate attributions into their workflow. The full survey text is available in Appendix A.

Participants and ethics: We recruited participants by sending email flyers to employees at organizations that run ICS and by sending private messages via LinkedIn to people with job titles relating to ICS security (e.g., OT Security Architect). We recruited seven participants in total. Participants were compensated with $5 gift cards. Our survey was approved under Exempt Review by our institutional review board.

Table VI: List of survey participants: their participant code, the type of ICS they operate, and their role at their organization.

	ICS Type	Role/Title
P1	Distributed control system	Cybersecurity, design and acquisition
P2	(Not disclosed)	Security engineer
P3	Unmanned vehicle ground control	Operator supervisor
P4	Various	Consulting and Research
P5	Electrical power generation and transmission	SCADA Engineering, System Integration and Security
P6	Electric Transmission	Security Engineer
P7	Various (manufacturing and distribution)	CISO

Table VII: Participants' ratings for the usefulness of hypothetical attribution outputs, varying the number of features shown and the error rate (1 being "not at all useful" and 5 being "extremely useful"). "Error rate" is the likelihood that the manipulated feature is not in the output. Bolded values indicate that the average rating was "moderately useful" or above.

# Output Features	**Usefulness (1-5)**		
Error rate: (H, M, L)	**(High)**	**(Medium)**	**(Low)**
2 (70%, 40%, 20%)	1.29	2.00	2.86
5 (50%, 30%, 10%)	1.57	2.57	**4.14**
10 (40%, 20%, 5%)	1.57	2.86	**4.29**
20 (30%, 10%, 5%)	2.14	**3.14**	**4.43**
All 34 (0%)			2.43

Results: Participants worked on a variety of ICS, ranging from electric transmission to controls for unmanned vehicles (see Table VI for a summary of their roles and types of ICS operated). All had experience with rule-based anomaly detection, and four reported using ML-based anomaly detection.

Participants described anomaly detection as the first step in the attack mitigation process, followed by manual investigation and correction (P1, P2, P4, P6). Some challenges reported by participants included (i) excessive false positives that led to "wild goose chases", requiring manual mitigation by operators (P1, P2, P5), and (ii) a lack of data and context in alerts raised by detectors, which made it difficult to trace root causes of anomalies (P4, P6). Prior work has found that, more broadly, SOC analysts have similar challenges with security alerts [8].

Participants provided several examples of how attributions would be integrated into their workflow. Attributions could help provide context on the relationship between system components (P3, P4) and inform follow-up diagnostic steps, such as running tests and consulting runbooks (P2, P4). Attributions could also be integrated with other data sources such as control system logs (e.g., SCADA) in a security information and event management (SIEM) system (P5, P7).

Participants perceived attribution outputs with more features and low error rates to be more useful (Table VIII). Participants reported attributions in the low-error-rate condition to be "very useful" (average score 4.14–4.43), if 20, ten, or five features were shown. For the medium-error-rate condition, roughly corresponding to our ensemble model's performance, participants reported attributions to be "moderately useful" if 20 features were shown (3.14) and "slightly useful" (2.00–2.86) for fewer features. For the high-error-rate condition, roughly corresponding to raw-error ranking, participants reported attributions to be between "slightly useful" and "not

[9] This survey was performed with an earlier version of SWaT with a feature selection step, and thus only 34 features were used.

at all useful" (1.29–2.14). Lastly, attributions showing all 34 features were reported as only "slightly useful" (2.43) by participants.

Participants mentioned several considerations regarding attribution accuracy and quantity of information. P1 and P3 generally preferred having more information available to the operator. P7 preferred seeing less information, as having too many false positives would require too much effort to investigate. P5 and P6 said that operators and organizations would be unlikely to trust models with high error rates. P2 and P4 explicitly weighed the tradeoff between error rates and the amount of information shown:

> A balanced trade-off is needed. Often having [a] list of max 10 [sensors] with minimal error rate is more useful than having less with high error rate. Depends on the needed follow-up testing effort to identify the one culprit finally. –P4

These results suggest that attributions could help ICS operators respond to anomalies, even without perfect accuracy. Attributions could provide a starting point for operators when investigating anomalies; for this use case, operators reported that an attribution method that performs as well as our proposed ensemble method would be moderately helpful, and preferred to see attribution scores for the top 10–20 features to balance accuracy and the amount of information shown.

Finding 5: ICS operators are likely to find our current best-performing attribution methods for anomaly detection models to be moderately useful when responding to incidents, even if the single manipulated sensor or actuator cannot be identified with high accuracy.

VI. Discussion and recommendations

In this section, we use our findings to provide recommendations for researchers and practitioners in ICS security.

A. Recommendations for researchers

Evaluate on diverse, complex ICS attacks: The accuracy of attribution methods depend heavily on ICS attack properties (Sec. IV-C). Despite the wide range of potential attack strategies, public ICS datasets predominantly contain high-magnitude, constant-valued manipulations [7], [27], and prior work evaluates attributions on only a small number of attacks from these datasets [34], [39].

When developing ICS anomaly detection and attribution methods, evaluations should be performed on a complex and diverse set of ICS attacks. This would ensure that attributions generalize across attack strategies and perform well on attacks that are most difficult to attribute with currently existing methods (low-magnitude, categorical-actuator-based attacks).

Design attribution methods specifically for ICS anomaly detection: When tested on full ICS datasets, we found that prior attribution strategies performed less well than previously suggested, and that our adapted ML-based attribution methods performed less well than anticipated (Sec. IV-A). Attributing ICS anomaly detection presents unique challenges: attributions are time-dependent (Sec. IV-B) and affected by additional feature dependencies (Sec. IV-C). Furthermore, the results of our survey (Sec. V) suggest that ICS operators would prefer attributions to show a list of 10–20 features to provide context for their investigation, rather than just the top few features.

Future work that designs attribution methods for ICS should be designed to directly address the aspects of ICS anomalies that make their attributions uniquely challenging, such as considering separate designs for sensors and actuators. These methods should also be evaluated with operators' preferences and workflows in mind, rather than optimizing solely for top feature accuracy.

B. Recommendations for practitioners

Consider attributions in workflows beyond the real-time detection case: Although it may seem most intuitive to compute attributions in real time at the moment when anomalies are detected, this strategy is suboptimal for attribution accuracy.

We found that that attributions are most effective when computed with an input that closely follows the start of the anomaly (e.g., 25 seconds): this is when the input to the attribution method includes some data on how the ICS has responded to the initial manipulation, but before the input is dominated by the manipulation's side effects (which may increase with time). However, in many attacks, the anomaly detector does not generate alerts at this ideal point in time.

Furthermore, we found that ML-based attribution methods can identify manipulated features, even when the input contains insufficient information for the anomaly-detection model to generate an alert (Sec. IV-B). Hence, to overcome the limitations of anomaly detection in providing timely detection for optimal attribution, we suggest using attribution methods in post-hoc settings, using tools like a data historian, which would allow the operator to leverage their domain expertise to explore optimal timings for attributions.

Use an ensemble of attribution methods: The optimal choice of attribution method differs based on the ICS, anomaly-detection model, and properties of the attack being attributed. A "silver bullet" solution does not yet exist for attributions of ICS anomalies. We found that, on average, a weighted ensemble of attributions from raw reconstruction error, saliency maps, and LEMNA outperforms all individual attribution methods (Sec. IV-D).

VII. Conclusion

In this work, we investigate how attribution methods can be used to identify the manipulated feature in an ICS attack. We compare across anomaly-detection methods, model architectures, and datasets, ultimately finding that attribution methods outperform raw-error rankings proposed in prior work, but only when their input coincides with the anomaly start. We also evaluate attribution methods across attack properties, finding that they are less accurate on manipulations with low magnitude, and manipulations on categorical actuators. Finally, we develop a strategy that uses an ensemble of attribution methods and show that it outperforms all individual attribution methods. We provide insights into the design and application of attribution methods for ICS anomaly detection, with recommendations for researchers and practitioners.

Acknowledgments

We thank the anonymous reviews and shepherd for their feedback in improving this work; Zifan Wang, Klas Leino, and Matt Fredrikson for insightful discussions about attribution adaptations; and Stephen Dai for help with software implementation and testing. This paper is based on work supported in part by the Secure and Private IoT initiative at Carnegie Mellon CyLab (IoT@CyLab) and by Mitsubishi Heavy Industries through the Carnegie Mellon CyLab partnership program; by the U.S. Army Research Office and the U.S. Army Futures Command under contract W911NF-20-D-0002; by DARPA under contract HR00112020006; and by the U.S. Department of Defense under contract FA8702-15-D-0002.

References

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

[2] "Lime: Explaining the predictions of any machine learning classifier," https://github.com/marcotcr/lime, 2021.

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

[4] "Autoregressive modeling tools in header-only C++," https://github.com/RhysU/ar, 2022.

[5] "PASAD: Process-aware stealthy attack detector," https://github.com/mikeliturbe/pasad, 2022.

[6] S. Adepu, N. Li, E. Kang, and D. Garlan, "Modeling and analysis of explanation for secure industrial control systems," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 17, no. 3-4, 2022.

[7] C. M. Ahmed, V. R. Palleti, and A. 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.

[8] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of SOC analysts' perspectives on security alarms," in *31st USENIX Security Symposium*, 2022.

[9] L. Antwarg, R. M. Miller, B. Shapira, and L. Rokach, "Explaining anomalies detected by autoencoders using shapley additive explanations," *Expert systems with applications*, vol. 186, 2021.

[10] W. Aoudi, M. Iturbe, and M. 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] G. Apruzzese, P. Laskov, E. Montes de Oca, W. Mallouli, L. Brdalo Rapa, A. V. Grammatopoulos, and F. Di Franco, "The role of machine learning in cybersecurity," *Digital Threats: Research and Practice*, vol. 4, no. 1, 2023.

[12] G. Apruzzese, P. Laskov, and J. Schneider, "SoK: Pragmatic assessment of machine learning for network intrusion detection," in *IEEE European Symposium on Security and Privacy*, 2023.

[13] M. Asiri, N. Saxena, R. Gjomemo, and P. Burnap, "Understanding indicators of compromise against cyber-attacks in industrial control systems: A security perspective," *ACM Transactions on Cyber-Physical Systems*, 2023.

[14] A. Bathelt, N. L. Ricker, and M. Jelali, "Revision of the Tennessee Eastman process model," *IFAC-PapersOnLine*, vol. 48, no. 8, pp. 309–314, 2015.

[15] A. A. Cardenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *6th ACM Symposium on Information, Computer and Communications Security*, 2011.

[16] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[17] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," *arXiv preprint arXiv:1705.07857*, 2017.

[18] A. Deng and B. Hooi, "Graph neural network-based anomaly detection in multivariate time series," in *AAAI Conference on Artificial Intelligence*, vol. 35, no. 5, 2021.

[19] J. J. Downs and E. F. Vogel, "A plant-wide industrial process control problem," *Computers & Chemical Engineering*, vol. 17, no. 3, pp. 245–255, 1993.

[20] A. Erba and N. O. Tippenhauer, "Assessing model-free anomaly detection in industrial control systems against generic concealment attacks," in *38th Annual Computer Security Applications Conference*, 2022.

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

[22] C. Feng, V. R. Palleti, A. Mathur, and D. Chana, "A systematic framework to generate invariants for anomaly detection in industrial control systems," in *Network and Distributed System Security Symposium*, 2019.

[23] P. Filonov, F. Kitashov, and A. Lavrentyev, "RNN-based early cyber-attack detection for the Tennessee Eastman process," *arXiv preprint arXiv:1709.02232*, 2017.

[24] C. Fung, S. Srinarasi, K. Lucas, H. B. Phee, and L. 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] Z. Gao, C. Cecati, and S. X. Ding, "A survey of fault diagnosis and fault-tolerant techniques—Part I: Fault diagnosis with model-based and signal-based approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, 2015.

[26] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, 2018.

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

[28] B. Green, M. Krotofil, and A. Abbasi, "On the significance of process comprehension for conducting targeted ICS attacks," in *Workshop on Cyber-Physical Systems Security and Privacy*, 2017.

[29] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "LEMNA: Explaining deep learning based security applications," in *ACM SIGSAC Conference on Computer and Communications Security*, 2018.

[30] D. T. Ha, N. X. Hoang, N. V. Hoang, N. H. Du, T. T. Huong, and K. P. Tran, "Explainable anomaly detection for industrial control system cybersecurity," *IFAC-PapersOnLine*, vol. 55, no. 10, 2022, 10th IFAC Conference on Manufacturing Modelling, Management and Control.

[31] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *30th Annual Computer Security Applications Conference*, 2014.

[32] D. Han, Z. Wang, W. Chen, K. Wang, R. Yu, S. Wang, H. Zhang, Z. Wang, M. Jin, J. Yang, X. Shi, and X. Yin, "Anomaly detection in the open world: Normality shift detection, explanation, and adaptation," in *Network and Distributed System Security Symposium*, 2023.

[33] D. Han, Z. Wang, W. Chen, Y. Zhong, S. Wang, H. Zhang, J. Yang, X. Shi, and X. Yin, "DeepAID: Interpreting and improving deep learning-based anomaly detection in security applications," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[34] C. Hwang and T. Lee, "E-SFD: Explainable sensor fault detection in the ICS anomaly detection system," *IEEE Access*, vol. 9, pp. 140 470–140 486, 2021.

[35] R. A. Jacob, S. Senemmar, and J. Zhang, "Fault diagnostics in shipboard power systems using graph neural networks," in *IEEE 13th International Symposium on Diagnostics for Electrical Machines, Power Electronics and Drives (SDEMPED)*, 2021.

[36] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "AI/ML for network security: The emperor has no clothes," in *ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[37] J. Jakubowski, P. Stanisz, S. Bobek, and G. J. Nalepa, "Explainable anomaly detection for hot-rolling industrial process," in *IEEE 8th International Conference on Data Science and Advanced Analytics*, 2021.

[38] E. Kang, S. Adepu, D. Jackson, and A. P. Mathur, “Model-based security analysis of a water treatment system,” in *2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems*, 2016.

[39] M. Kravchik and A. Shabtai, “Efficient cyber attack detection in industrial control systems using lightweight neural networks and PCA,” *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 4, 2022.

[40] M. Krotofil and J. Larsen, “Rocking the pocket book: Hacking chemical plants,” in *DEFCON Conference*, 2015.

[41] D. Kushner, “The real story of Stuxnet,” *IEEE Spectrum*, vol. 53, no. 3, 2013.

[42] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[43] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, “TABOR: A graphical model-based approach for anomaly detection in industrial control systems,” in *Asia Conference on Computer and Communications Security*, 2018.

[44] Y. Liu, P. Ning, and M. K. Reiter, “False data injection attacks against state estimation in electric power grids,” *ACM Transactions on Information and System Security*, 2011.

[45] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, 2017.

[46] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao, “Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities,” *ACM Computing Surveys*, vol. 54, no. 5, pp. 1–36, 2021.

[47] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, “Kitsune: an ensemble of autoencoders for online network intrusion detection,” in *Network and Distributed System Security Symposium*, 2018.

[48] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer, “SoK: Explainable machine learning for computer security applications,” in *IEEE European Symposium on Security and Privacy*, 2023.

[49] Á. L. Perales Gómez, L. Fernández Maimó, A. Huertas Celdrán, and F. J. García Clemente, “MADICS: A methodology for anomaly detection in industrial control systems,” *Symmetry*, vol. 12, no. 10, 2020.

[50] R. Quinonez, J. Giraldo, L. Salazar, E. Bauman, A. Cardenas, and Z. Lin, “SAVIOR: Securing autonomous vehicles with robust physical invariants,” in *29th USENIX Security Symposium*, 2020.

[51] M. T. Ribeiro, S. Singh, and C. 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.

[52] Robert M. Lee, Michael J. Assante and Tim Conway, “Analysis of the cyber attack on the Ukrainian power grid,” *Electricity Information Sharing and Analysis Center (E-ISAC)*, vol. 388, 2016.

[53] S. Sapkota, A. K. M. N. Mehdy, S. Reese, and H. Mehrpouyan, “FALCON: Framework for anomaly detection in industrial control systems,” *Electronics*, vol. 9, no. 8, 2020.

[54] S. Senemmar and J. Zhang, “Deep learning-based fault detection, classification, and locating in shipboard power systems,” in *2021 IEEE Electric Ship Technologies Symposium (ESTS)*, 2021.

[55] M. Sharif, S. Bhagavatula, L. Bauer, and M. K. Reiter, “Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition,” in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[56] K. Simonyan, A. Vedaldi, and A. Zisserman, “Deep inside convolutional networks: Visualising image classification models and saliency maps,” *arXiv preprint arXiv:1312.6034*, 2013.

[57] B. Singer, A. Pandey, S. Li, L. Bauer, C. Miller, L. Pileggi, and V. Sekar, “Shedding light on inconsistencies in grid cybersecurity: Disconnects and recommendations,” in *IEEE Symposium on Security and Privacy*, 2022.

[58] J. Slowik, “Evolution of ICS attacks and the prospects for future disruptive events,” *Threat Intelligence Centre Dragos Inc*, 2019.

[59] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, “Smoothgrad: removing noise by adding noise,” *arXiv preprint arXiv:1706.03825*, 2017.

[60] K. Stouffer, “Guide to industrial control systems (ICS) security,” *NIST special publication*, vol. 800, no. 82, 2011.

[61] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” in *34th International Conference on Machine Learning*, 2017.

[62] C. Tang, L. Xu, B. Yang, Y. Tang, and D. Zhao, “GRU-Based interpretable multivariate time series anomaly detection in industrial control system,” *Computers & Security*, vol. 127, p. 103094, 2023.

[63] L. Tong, B. Li, C. Hajaj, C. Xiao, N. Zhang, and Y. Vorobeychik, “Improving robustness of ML classifiers against realizable evasion attacks using conserved features,” in *28th USENIX Security Symposium*, 2019.

[64] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, “Limiting the impact of stealthy attacks on industrial control systems,” in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[65] S. Verma, J. Dickerson, and K. Hines, “Counterfactual explanations for machine learning: A review,” *arXiv preprint arXiv:2010.10596*, 2020.

[66] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, “Evaluating explanation methods for deep learning in security,” in *IEEE European Symposium on Security and Privacy*, 2020.

[67] L. Wen, X. Li, L. Gao, and Y. Zhang, “A new convolutional neural network-based data-driven fault diagnosis method,” *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, 2018.

[68] K. Wolsing, L. Thiemt, C. v. Sloun, E. Wagner, K. Wehrle, and M. Henze, “Can industrial intrusion detection be SIMPLE?” in *27th European Symposium on Research in Computer Security*, 2022.

[69] L. Yang, W. Guo, Q. Hao, A. Ciptadi, A. Ahmadzadeh, X. Xing, and G. Wang, “CADE: Detecting and explaining concept drift samples for security applications,” in *30th USENIX Security Symposium*, 2021.

[70] K. Zetter, “A cyberattack has caused confirmed physical damage for the second time ever,” WIRED, 2015, https://www.wired.com/2015/01/german-steel-mill-hack-destruction/.

[71] X. Zhang, T. Parisini, and M. Polycarpou, “Sensor bias fault isolation in a class of nonlinear systems,” *IEEE Transactions on Automatic Control*, vol. 50, no. 3, 2005.

[72] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones, “Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks,” *arXiv preprint arXiv:1911.04278*, 2019.

Appendix

A. *Full text of ICS operator survey and results*

In this Appendix, we show the questions in the survey of ICS experts, which we report on in Sec. V, and a summary of the responses where participants rated the usefulness of different hypothetical attributions, varying the error rate and the number of features shown (Table VII).

1) What type of industrial control system (ICS) do you operate? (Free response)
2) What is your role in operating this ICS? (Free response)
3) Do you, or does your organization use an anomaly detection system to detect potential attacks on the ICS you operate? (Yes / No / Not Sure)
4) Do you have any current or prior experience working with anomaly detection systems? (Yes / No)
5) What method(s) does the anomaly detection system that you have experience with use to detect anomalies? (choose all that apply: Rule-based anomaly detection, Machine learning-based anomaly detection, Not sure, Other (please specify))
6) What kind of information does the anomaly detection system that you have experience with provide you, and what does the human interface look like? (Free response)

7) What aspects of the anomaly detection system that you have experience with are most useful to you when monitoring the system or responding to potential attacks? (Free response)
8) What aspects of the anomaly detection system that you have experience with are most challenging or confusing to work with when monitoring the system or responding to potential attacks? (Free response)

For the last part of this survey, we describe a simulated attack on the Secure Water Treatment (SWaT) testbed, and outputs from a proposed, machine-learning based anomaly detection system.

In this scenario, the anomaly detection system has detected an anomaly. In addition to alerting the operator to the occurrence of an anomaly, the system also reports a list of sensors and actuators that the model predicts to be the cause of the anomaly. Here is the output of the anomaly detection system for a simulated attack on SWaT: (Table VIII)

Table VIII: Sample output from detector used in the survey.

Feature	Current Value	Anomaly Score	Alert Level
DPIT301	19.59	10.11	HIGH
MV302	2.00	8.74	HIGH
P302	2.00	2.03	HIGH
FIT201	2.44	0.54	MEDIUM
P203	2.00	0.54	MEDIUM
P101	2.00	0.53	MEDIUM
MV101	2.00	0.49	MEDIUM
FIT101	2.67	0.48	MEDIUM
MV201	2.00	0.46	MEDIUM
P403	1.00	0.45	MEDIUM

Currently, the anomaly detection system is configured to show the top 10 most likely sensors and actuators to be responsible for the anomaly, out of 34 total sensors. However, it can be configured to show more or fewer sensors or actuators. The more sensors it shows, the more likely the sensor or actuator that is the root cause of the anomaly is in the list.

In this next part, we propose several alternative configurations of the system, with different numbers of sensors/actuators shown.

9) How useful would an anomaly detection system be if it showed 2 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale: not at all useful, slightly useful, moderately useful, very useful, extremely useful)
   - 70% • 40% • 20%
10) How useful would an anomaly detection system be if it showed 5 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   - 50% • 30% • 10%
11) How useful would an anomaly detection system be if it showed 10 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   - 40% • 20% • 5%
12) How useful would an anomaly detection system be if it showed 20 sensors or actuators (out of 34), with the following error rates? (5 point Likert scale)
   - 30% • 10% • 5%
13) How useful would an anomaly detection system be if it showed all 34 sensors or actuators? (5 point Likert scale)

Table IX: The number of detected attacks (at least one example exceeds the MSE threshold), when using a validation-error-based tuning for the error threshold (99.5% for AR/PASAD and 99.95% for deep learning models).

		Total Detected				
	Total	**AR**	**PASAD**	**CNNs**	**GRUs**	**LSTMs**
SWaT	43	22	40	33	20	34
WADI	24	21	4	15	8	15
TEP	89	85	86	55	58	64

14) Please explain your responses above: do you think it would be helpful for the anomaly detection system to display more sensors and actuators with lower error rates, or fewer sensors and actuators with higher error rates? (Free response)
15) Which of these two options would you rather have, for this anomaly detection system? (Choose one)
   - The entire list of sensors and actuators, and anomaly scores and alert levels for all of them
   - A list of 10 sensors and actuators the anomaly detection model thinks are most likely to be the cause of the anomaly, which correctly identifies the root cause 95% of the time
16) If this anomaly detection system showed 10 sensors and actuators, what percent of the time does the anomaly detection system need to correctly include the cause of the anomaly for it to be useful? (0%-100%)
17) If you had an anomaly detection system like this, how would you use the information provided to find the root cause of the anomaly? How would you integrate it into your workflow? (Free response)

B. *Technical implementation details*

In this section, we provide additional implementation details. Table IX shows the number of attacks for which one least one example exceeds a validation-error-defined threshold: these examples are used as the basis of evaluation in Sec. IV-A.

Next, we provide the technical definitions of our attribution methods, adapted for ICS anomaly detection. Each attribution method A requires an anomaly-detection model $F(x_{t-h}, ..., x_{t-1}) \rightarrow \hat{x}_t$ and time-series input $X_e \in \mathbb{R}^{dxh}$: computing an attribution $A(X_e)_j$ for feature j.

Counterfactuals: Given input X_e and baseline X_b, a counterfactual attribution is computed by changing the value of each feature and measuring the change in the MSE. We use the feature-wise average benign value as X_b and define a masking function $M_j(X)$, which removes all but the j-th feature from X. We define the additive counterfactual A_A:

$$A_A(X_e)_j = MSE(X_b - M_j(X_b) + M_j(X_e)) - MSE(X_b)$$

We define the subtractive counterfactual A_S:

$$A_S(X_e)_j = MSE(X_e) - MSE(X_e - M_j(X_e) + M_j(X_b))$$

Saliency map [56]: 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:

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

SmoothGrad [59]: Prior work found that saliency maps are sensitive to small input changes; in response, SmoothGrad averages saliency maps over multiple perturbations of X_e.

$$A_{SG}(X_e) = X_e \times \frac{1}{n} \sum \frac{\partial MSE(X_e + \varepsilon)}{\partial X}, \varepsilon \sim N(0, \sigma)$$

SmoothGrad is defined by n, the number of samples used, and σ, the sampled noise variance. We use the suggested values $\sigma = 0.1 * (X^{max} - X^{min})$ and $n = 50$.

Integrated gradients [61]: Integrated gradients calculate the change in a quantity of interest between X_e and a baseline X_b, producing more meaningful results. Attributions are computed through an approximate path integral that interpolates between X_b and X_e.

$$A_{IG}(X_e) \approx (X_e - X_b) \times \frac{1}{n} \sum_{k=1}^{n} \frac{\partial MSE(X_b + \frac{k}{n}(X_e - X_b))}{\partial X}$$

Using larger n increases the accuracy of the path-integral estimate. In our work, we use $n = 200$ and the feature-wise average benign value as the baseline X_b.

Expected gradients [21]: Instead of assuming a single baseline, expected gradients use samples from the training distribution D. Attributions are computed as the expectation over baseline examples and interpolation points. The expectation is approximated by averaging over estimates: for each estimate i, a sample baseline X_{bi} is drawn from the benign training dataset and an interpolation point α_i is drawn from the uniform distribution.

$$A_{EG}(X_e) \approx \frac{1}{n} \sum_{i=0}^{n} [(X_e - X_{bi}) \frac{\partial MSE(X_{bi} + \alpha_i(X_e - X_{bi}))}{\partial X}]$$
$$\alpha_i \sim U(0,1), X_{bi} \sim D$$

To sample baselines from our training dataset, we sample a timestep t from the benign dataset and use its process values and corresponding history. As the number of samples n increases, the stability of the expected gradients increases. We use the suggested $n = 200$ for convergence.

C. Selecting attribution methods with a synthetic benchmark

ICS anomaly detection is inherently noisy: benign features produce (small) errors and interactions between sensors and actuators can complicate the analysis of attribution methods. To remove these effects from our evaluation, we craft synthetic inputs to systematically evaluate an attribution method for a given anomaly-detection model.

Method: We evaluate attribution methods in a synthetic setting where a controlled manipulation introduced to a single input feature is the *only* source of error in an unsupervised anomaly-detection model.

First, we craft a zero-MSE, input-output pair by selecting an input window X^{base} from the benign training data, feeding it to an anomaly-detection model F, and storing the corresponding process-value prediction $Y^{base} = F(X^{base})$. We then perturb a feature j in X^{base} by two-standard-deviations to generate X^{pert}: when computing errors, we compare the prediction $F(X^{pert})$ with the synthetic ground-truth Y^{base}. To compute attributions, an attribution method uses X^{pert} and Y^{base} as the input window and ground-truth respectively.

The perturbation is the only change introduced in the zero-MSE input-output pair, so a correct attribution would assign the perturbed feature j the highest score. We rank feature j in the attribution and repeat this measurement for all features, ultimately computing the AvgRank for each method.

Results: We evaluate each attribution method for all nine combinations of model architecture (CNNs, GRUs, and LSTMs) and dataset (SWaT, WADI, TEP). Figure 7 shows the resulting AvgRank across all synthetic inputs. Three attribution methods perform well: the saliency map (SM), SHAP, and LEMNA. These three methods outperform the MSE on all models and datasets, with the exception of SHAP on TEP (e.g., for SWAT CNNs, the MSE AvgRank is 16.4, whereas the SM, SHAP, and LEMNA AvgRanks are 2.6, 2.6 and 3.8 respectively). This suggests that attribution methods (black-box or white-box) can provide stronger insight than raw MSEs when attributing ICS anomalies.

White-box variants (SG, IG, and EG) outperform saliency maps on images [59], [61], [21], and our results suggest that the performance of these methods may not translate to ICS anomaly detection. We suggest two reasons why: First, the dynamics of ICS are more precise than in images; although adding random noise to images (as done in SmoothGrad) helps generalize attributions for images, randomness does not provide this benefit for ICS anomaly detection. Second, benign ICS behavior cannot be well-represented with a single (or sample of) reference input(s), and thus choosing an effective baseline (required for IG and EG) is difficult.

When comparing black-box attribution methods, we find that SHAP and LEMNA outperform LIME. LIME uses a linear approximation: in contrast, SHAP (which uses Shapley values) and LEMNA (which uses a fused-lasso, Gaussian mixture model) can better capture the inter-feature dynamics of an ICS.

D. Evaluating against stealthier manipulations

Although attackers can adjust modify manipulation properties to reduce attribution accuracy, stealthier manipulations strategies can be even more effective. In this section, we explore how attribution methods are affected by stealthier manipulation strategies, extending beyond strategies used for current datasets. We find that: (i) multi-point attacks are more difficult to attribute and (ii) summing and linear manipulations are particularly effective at reducing attribution accuracy.

1) Multi-point attacks: We first consider multi-point attacks: when multiple features are manipulated simultaneously. Multi-point attacks are included in the SWaT and WADI datasets. In general, correctly attributing multi-point attacks is more difficult, since the effects of multiple manipulations are observed in the ICS. We use a one-way ANOVA test to compare the AvgRank distributions for single-point attacks and multi-point attacks; results are in the third column of Table V.

We find that LEMNA is significantly more accurate for single-point attacks; the AvgRank is over three times higher when a multi-point attack is performed. In general, all attribution methods (except raw AR-score ranking) are less accurate on multi-point attacks.

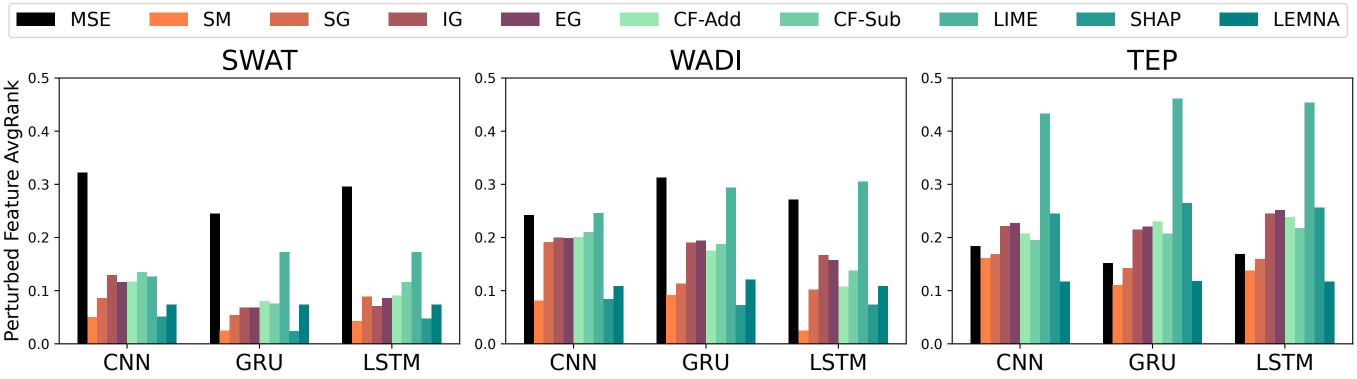


Figure 7: Results for all datasets (SWaT, WADI, TEP) and model architectures (CNN, GRU, LSTM) over our synthetic benchmark; the AvgRank is shown (lower is better). Within white-box attribution methods (SM, SG, IG, EG), the saliency map (SM) always performs the best, and within black-box attribution methods (CF-Add, CF-Sub, LIME, SHAP, LEMNA), SHAP and LEMNA generally perform best. Notably, SM and LEMNA always outperform the raw-ranking of MSEs (MSE).

Table X: By introducing summing or linear manipulations, attackers can reduce attribution method accuracy. When comparing summing/linear manipulations to their constant alternatives: (i) the manipulation is detected later and (ii) the AvgRank for all attribution methods increases.

		Constant	Summing	Linear
CNNs	Detection latency	200s	694s	1232s
	MSE AvgRank	0.075	0.140	0.147
	SM AvgRank	0.287	0.536	0.619
	SHAP AvgRank	0.223	0.415	0.362
	LEMNA AvgRank	0.117	0.423	0.551
GRUs	Detection latency	242s	1132s	1316s
	MSE AvgRank	0.064	0.102	0.072
	SM AvgRank	0.279	0.513	0.525
	SHAP AvgRank	0.204	0.472	0.366
	LEMNA AvgRank	0.087	0.389	0.468
LSTMs	Detection latency	174s	571s	1090s
	MSE AvgRank	0.087	0.151	0.113
	SM AvgRank	0.355	0.551	0.574
	SHAP AvgRank	0.174	0.377	0.464
	LEMNA AvgRank	0.072	0.343	0.634

2) Summing and linear manipulations: We implement two alternate manipulation types with the TEP simulator: linear and summing manipulations[10]. These manipulations achieve the same sensor value as the constant-valued manipulations prevalent in SWaT, WADI, TEP, and prior work [40], [15], yet are more stealthy (i.e., harder to detect and attribute correctly).

Method: A linear manipulation incrementally increases in magnitude with each timestep, and a constant-sum manipulation adds a constant value to the original sensor value at each timestep, which maintains the natural amount of noise. We define the stealthier manipulation types based on the original attack model used in Sec. III-A. For a summing manipulation:

$$x'_j(t) = x_j(t) + c$$

For a linear manipulation, where t_a is the initial point of the attack, and m is the slope:

$$x'_j(t) = m(t - t_a) + x_j(t_a)$$

We use the modified TEP simulator to perform the stealthier manipulations on every sensor in the system. For each sensor, we perform a two-standard-deviation-magnitude manipulation with the stealthier manipulation types. We compare the AvgRank and detection latency for the five cases where, regardless of manipulation type, all attacks are detected by all three models.

Results: Table X shows the resulting AvgRank and detection latency for constant, constant-sum, and linear manipulations. On average, performing an attack with a stealthier manipulation causes the attack to be detected later: compared to constant manipulations, constant-sum manipulations are detected at least three times later and linear manipulations are detected at least five times later. In addition, the attributions computed at these detection points are less accurate: the AvgRank increases in all cases. When using alternate manipulation types, attribution accuracy decreases while the same target sensor value is achieved.

E. Artifact appendix

Description, requirements, and how to access: Our artifact is comprised of three independent components:

- A Python library[11] for attributions on ICS anomaly detection models and datasets (focus of this Appendix)
- An ICS simulator[12], adapted from a publicly available version, that supports manipulation of sensors/actuators and subsequent data collection.
- A dataset containing 286 manipulation traces[13], produced from the above simulator.

Dependencies: Our artifact can be executed on commodity hardware. Software dependencies are listed for each artifact:

- **Attribution library**: Our code depends on Python 3.7, and a variety of Python libraries from pip. A requirements file has been provided with installation instructions.

[10] Included in our public set of 286 manipulations.

[11] https://github.com/pwwl/ics-anomaly-attribution

[12] https://github.com/pwwl/tep-attack-simulator

[13] https://doi.org/10.1184/R1/23805552

- **ICS simulator**: The simulator depends on MATLAB R2021a. Installation instructions can be found in the associated README.
- **Dataset of synthetic manipulations**: None

Our experiments depend on three datasets: SWaT (provided by iTrust), WADI (provided by iTrust), and TEP (generated by our ICS simulator). The authors of SWaT and WADI have requested that the dataset not be re-shared, and users must request the datasets through the iTrust website[14].

Artifact installation & configuration: Our library requires Python 3.7 and a variety of pip packages. We recommend using conda to manage dependencies; installation instructions are provided in the repository README.

Major claims: The artifact supports the contribution listed in the paper introduction: we create (i) an open-source library of attribution methods for reconstruction-based, time-series ICS anomaly-detection models, (ii) a modified ICS simulator that performs well-defined sensor/actuator manipulations, and (iii) a dataset of 286 synthetic ICS anomalies for testing.

Evaluation: We provide a minimal example to show that our attribution library is functional, configurable, and usable. All the main results in our work follow the same structure as our minimal examples. To scale up the examples and produce the full results in this paper, users must evaluate and compare widely across additional models, datasets, and attribution methods. The artifact README contains a more complete set of instructions, with specific commands and command-line arguments for each workflow.

Experiment (E1): [CNN SWaT attribution example] [10 human-minutes + up to 2 compute hours]: In this experiment, we execute a minimal attribution example using a CNN model and the SWaT dataset. Four attribution methods are compared: raw MSE, saliency maps (SM), SHAP, and LEMNA. For each attribution method, the attribution scores are computed and a final ranking of the attacked feature is determined.

[Preparation] First, create the needed directories that will be populated with metadata with the `make_dirs.sh` script. Next, request the SWaT dataset and set up the dataset environment by following the instructions in the `data/SWAT` directory.

Next, train a CNN model on the SWaT dataset, using the `main_train.py` script, specifying a CNN architecture and the SWaT dataset. The default configuration trains a 2-layer, 64-unit, 3-kernel, 50-history CNN, which is the same architecture used in our experiments.

Next, prepare metadata with the `save_model_mses.py` and `save_detection_points.py` scripts, which compute and store the test MSEs and detection times respectively.

[Execution] To compute attributions for the sample SWaT attack, use the scripts in the `explain-eval-attacks` directory: `main_grad_explain_attacks.py` and `main_bbox_explain_attacks.py`. In this workflow, we compute attributions with saliency maps (SM), SHAP, and LEMNA. Each script will compute and store all attribution scores for 150 timesteps, at the two timings described in our work: (i) the best-guess timing (using the labeled beginning of the attack) and (ii) the practical timing (using the point of detection).

Bash scripts `expl-full-bbox.sh` and `expl-full-swat.sh` are provided for reference. Note: running the explanations may take anywhere from 20 minutes to two hours depending on your machine. If the experiments are prohibitively expensive, one can reduce the number of timesteps with the `num_samples` parameter; this will limit the results of the timing average measurements in the next section. At the end of this section, `.pkl` files should be created for each attribution method.

[Results] To interpret the results of the attribution methods, the script `main_feature_properties.py` is used. This script will determine the ranking of the attacked feature over the list of attacks, and compute the AvgRank. For the minimal example, presented here, we only print the ranking of each attribution method for a single SWaT attack. Sample results are shown in Table XI; due to randomness in model training and attribution methods, the actual results may differ.

Table XI: Example experiment results for SWaT attack #1.

Attribution Method	**Best-guess Rank**	**Practical Rank**	**Best-guess Timing Rank**	**Practical Timing Rank**
MSE	1	3	11.59	13.98
SM	4	15	2.23	1.31
SHAP	10	15	16.03	16.43
LEMNA	7	9	12.84	13.79
Ensemble			3.74	2.81

To produce the full results used in our paper, attributions must be computed for all attacks. Averaging attributions over the full set of attacks will produce the results in Fig. 5. Averaging attributions within individual attack subsets will produce the results in Table V. Finally, averaging attributions while maintaining the full timing of 150 timesteps will produce the results in Fig. 6.

[Customization] Each script's function (model training, attribution methods, computing rankings) is configured to accept command line arguments that can modify the model architecture (CNNs, GRUs or LSTMs), dataset (SWaT, WADI, or TEP), or attribution method used (SM, SG, IG, EG, LIME, SHAP, LEMNA). For a full description of all command-line arguments, please see the artifact README.

[14] https://itrust.sutd.edu.sg/itrust-labs_datasets/