GeCos Replacing Experts: Generalizable and Comprehensible Industrial Intrusion Detection

Konrad Wolsing[*,‡] Eric Wagner[*,‡] Luisa Lux[*,‡] Klaus Wehrle[‡] Martin Henze[‡,*]

[*]*Fraunhofer FKIE* [‡]*RWTH Aachen University*

Abstract

Protecting industrial control systems against cyberattacks is crucial to counter escalating threats to critical infrastructure. To this end, Industrial Intrusion Detection Systems (IIDSs) provide an easily retrofittable approach to uncover attacks quickly and before they can cause significant damage. Current research focuses either on maximizing automation, usually through heavy use of machine learning, or on expert systems that rely on detailed knowledge of the monitored systems. While the former hinders the interpretability of alarms, the latter is impractical in real deployments due to excessive manual work for each individual deployment.

To bridge the gap between maximizing automation and leveraging expert knowledge, we introduce GeCo, a novel IIDS based on automatically derived comprehensible models of benign system behavior. GeCo leverages state-space models mined from historical process data to minimize manual effort for operators while maintaining high detection performance and generalizability across diverse industrial domains. Our evaluation against state-of-the-art IIDSs and datasets demonstrates GeCo's superior performance while remaining comprehensible and performing on par with expert-derived rules. GeCo represents a critical step towards empowering operators with control over their cybersecurity toolset, thereby enhancing the protection of valuable physical processes in industrial control systems and critical infrastructures.

1 Introduction

Cybersecurity for Industrial Control Systems (ICSs) is a flourishing research topic [30] as more attacks endanger the environment, humans, and, in the case of critical infrastructure, our society [11]. Consequently, tremendous efforts are invested in securing industrial facilities, especially with the help of Industrial Intrusion Detection Systems (IIDSs) [22, 30, 42]. IIDSs monitor the physical process of an ICS and alert the operators in the case of identified abnormal behavior. Thereby, they serve as a last line of defense if existing upstream security measures (e.g., firewalls or authentication) are breached.

Concerning the tremendous research efforts put into designing appropriate IIDSs, current approaches can be roughly divided into two areas: First, data-driven IIDSs that learn purely on historical data samples and, therefore, usually base their detection algorithm on machine learning. On the other end of the spectrum, some IIDSs depend on external expert- or system-information provided, e.g., in the form of mathematical models of the ICS process. Thus, the latter require significant manual engagement by operators.

However, both directions exhibit opposing advantages and disadvantages. Data-driven IIDSs facilitate applicability to different ICS applications as long as the required training data is available [63]. But, their machine learning background hinders the transparency needed to comprehend alerts [28]. In the end, a simple alert from the entire system does not suffice to efficiently identify false positives or localize anomalies or attacks [43]. In contrast, IIDSs relying on expert knowledge are severely limited w.r.t. transferability to new domains. These systems demand manual configuration based on scarce expert knowledge and thus risk operating with incomplete models. Still, as experts are an integral part of the training process, alerts promise to be more easily comprehensible, potentially leading to faster mitigation of attacks.

Large cooperations may tackle some of these disadvantages through e.g., additional personnel, but smaller facilities lack the resources for such measures. However, also smaller operators of ICSs, especially critical infrastructure, regularly become the target of cyberattacks. A recent attack on a Texas water facility, leading to overflowing tanks for up to 45 minutes before detection, is only one of an exploding number of recent cyberattacks on critical infrastructure [45]. To address these concerns, the upcoming European NIS-2 regulation even requires the deployment of appropriate intrusion detection for a broad scope of relatively small businesses [1].

With this paper, we strive to combine the advantages of explainability and automation into an IIDS. Thus, we want to offload the hard manual work of experts to computers, thereby avoiding human error or incompleteness in modeling. Meanwhile, alerts should remain easily explainable for quick

This paper appears in the *Proceedings of the 34th USENIX Security Symposium*.

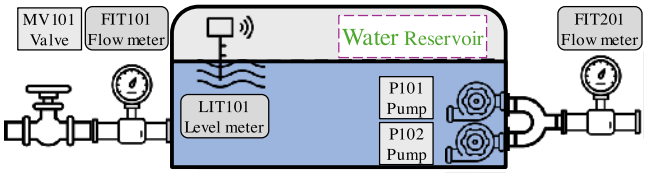


Figure 1: Exemplary ICS of a water reservoir to provide a constant supply of water to subsequent processes. Actuators are drawn with rectangles, sensors with rounded rectangles.

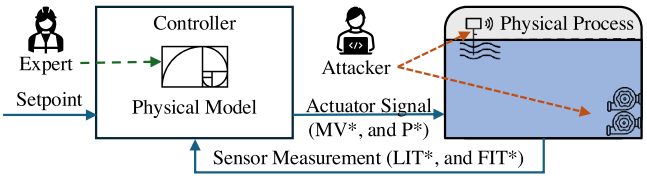


Figure 2: Digital control loops automatize physical processes according to a model provided by experts. Yet, ICSs' advances in digitalization pose vulnerabilities to cyberattacks.

reactions to eventual anomalies. As ICS operators likely have a background in control theory, we consider state-space models that describe the physical behavior and dependence of each sensor and actuator individually [17], as a viable modeling strategy for automatic learning and interpretability.

Concretely, we propose GeCo, a *ge*neralizable and *co*mprehensible IIDS that lowers the burden for ICS operators to deploy IIDSs. In contrast to prior works that rely on the labor-intensive manual specification of state-space models [9, 18, 51], GeCo automatically derives appropriate models expressed as mathematical correlations from historical process data based on *function templates* suitable for various ICSs and domains. An alarm is thus always associated with one or multiple sensor readings, such that it can be easily verified by operators who know the control logic.

Contributions. In order to address the need for effective and easy-to-use IIDSs, we make the following contributions:

- We design and implement GeCo, a highly effective IIDS automating the work of ICS experts, providing comprehensible alarms, and generalizing to various industrial domains.
- We demonstrate GeCo's effectiveness in a detailed evaluation against five state-of-the-art IIDSs, surpassing their detection performance in most metrics.
- We show that GeCo keeps up with expert-derived rules and prove its effectiveness and understandability across four datasets without requiring experts during training.

2 Background on Industrial Control Systems

ICSs enable the automation of e.g., water distribution or power generation. In the following, we introduce ICSs along an example, explain how digital control-loops implement the underlying process logic, and discuss the emerging risks of cyberattacks and their mitigation through intrusion detection.

2.1 A Simple ICS Example

Throughout this paper, we use a water storage tank subsystem of the prominent SWaT research testbed [31] as illustrative example. This system (cf. Fig. 1) stores water in a tank for further processing. The tank is equipped with sensors and actuators to monitor and manipulate the physical state. Flow meters measure the inflow (FIT101) and outflow (FIT201) of the water tank, and a level meter (LIT101) measures the water level. The actuators involved are a motorized valve (MV101) that controls the water inflow and a pump (P101) as well as a backup pump (P102) that forwards water for further processing. A Programmable Logic Controller (PLC) controls the binary state of these actuators (on or off). Here, the PLC aims to maintain a sufficient water supply and prevent the tank from draining or overflowing.

2.2 Digital Control Loops

Experts program PLCs to automatize the underlying process logic based on the sensors and actuators (cf. Fig. 2). The PLC continuously sends commands to the actuators, and obtains reports of their impact on the physical world through sensor readings. Usually, the effect is compared with a desired setpoint, which is called a closed-loop system [23].

Our example, i.e., checking whether the water level (LIT101) is low and then opening the valve (MV101), is a simple control loop. However, they can also control complex systems by leveraging multiple in- and outputs to model complicated functions. One method to describe the physical process is through state-space models [17], which are mathematical equations predicting the evolution of a process: The change in the water tank's level can be expressed as $\frac{\delta LIT101}{\delta t} = \frac{FIT101 - FIT201}{A}$, where A is the area of the tank [9].

2.3 Intrusion Detection for ICS Cybersecurity

ICSs usually have many control loops distributed across multiple PLCs that communicate via industrial protocols [37]. Ultimately, advanced connectivity demands lead to Internet-connected ICSs [20]. Previously isolated ICSs are thus now threatened by cyberattacks that e.g., manipulate the in- our outputs of a control loop (cf. attacker in Fig. 2). Indeed, ICSs face an increasing number of cyberattacks [11].

Industrial Intrusion Detection Systems (IIDSs) promise a viable and retrofittable protection mechanism, which has seen an enormous increase in (academic) interest, with the number of yearly publications rising from 5 IIDSs in 2008 to over 130 in 2021 [42]. This research body can roughly be divided along two important axes: *input data* and *detection technique* [42].

The *input data* may be network traffic, host data, or physical

process data. We are primarily interested in process-aware IIDSs that monitor the physical process to detect abnormal behavior, e.g., the upcoming overflowing of a water tank [45].

The *detection techniques* can be distinguished into knowledge- and behavior-based IIDSs. Behavior-based IIDSs expect training data of a normally operating ICS to learn normal behavior and consequently indicate any deviation as an anomaly. Knowledge-based IIDSs additionally expect samples of cyberattacks during training. However, knowledge-based IIDSs are criticized in the literature because (1) they tend to overfit detecting only known attacks and (2) it is difficult to extract attack samples in real world [7, 25, 41, 62]. Consequently, we focus on behavior-based IIDSs for which training data can be recorded during regular ICS operations.

3 Related Work

The overarching goal of an IIDS is to notify ICS operators in case of a cyberattack without emitting too many false positives [10]. Thus, detection performance is the primary subject of most scientific evaluations [30, 42]. However, two other factors are equally relevant, namely generalizability and comprehensibility. First, in the ICS sector, two facilities are rarely built identically, featuring heterogeneous hardware and a wide variety of industrial protocols [47]. Therefore, the generalizability of an IIDS, i.e., its ability to apply to multiple ICS applications or even domains, is crucial [53, 63]. Secondly, IIDSs should provide guidance for operators to understand the alert, trace its cause, or discard false alarms [25, 28, 61].

To understand to which extent existing IIDSs can satisfy the often ignored aspects of generalizability and comprehensibility, we investigate relevant publications (cf. Tab. 1). Assessing these, it becomes apparent that existing IIDSs roughly fall into two categories: Mechanisms that are purely data-driven and those that make use of external system-knowledge provided by experts during the IIDS's training phase.

Data-driven IIDSs employ vastly distinct detection methodologies ranging from classical machine learning such as one-class SVMs [34], Neural Networks [36], or hidden Markov models [5] to various ICS-specific approaches [12, 16, 26, 44, 61, 64]. One common advantage is the minimal amount of input required for training. They merely leverage historical data samples of the physical process, making them presumably applicable in various use cases [63]. Additionally, through their automation, they usually learn a complete model of the entire ICS, making it harder for attackers to circumvent detection. On the downside, as machine learning and especially custom detection methods are heavily used, understandability, e.g., through attribution methods, is difficult to achieve [28]. At the same time, this opaque operation risks model overfitting as the trained model can hardly be humanly validated [41, 62].

Knowledge-based IIDSs are enhanced with expert knowledge, of which a large portion requires input in the form of mathematical descriptions (cf. Sec. 2.2) either as a state-space model or a physical model. Given that the foundation is an expert-provided model, the IIDS becomes comprehensible for the operator and even allows them to perform corrections to the model. On the contrary, the IIDS is limited to those ICSs with the necessary resources to precisely model their processes, resulting in IIDSs that are often specifically designed for one scenario [18, 51]. Moreover, the models might be incomplete due to the amount of manual effort required by humans designing them, e.g., modeling just the water flows in a scenario similar to our example [9, 29], giving attackers the potential to hide their actions. Hence, the completeness and generalizability of these IIDSs are severely limited.

Table 1: Research is split into IIDSs using data-driven learning methods and those depending on (manual) input, which usually leads to incomplete models due to human efforts.

	Paper	Venue	External Information	Comp. Model
Data-driven	Aggarwal et al. [5]	CPS-SPC	✗	✓
	Aoudi et al. [12]	CCS	✗	✗
	Cardenas et al. [15]	AsiaCCS	✗	✓
	Castellanos et al. [16]	ACNS	✗	✓
	Feng et al. [26]	NDSS	✗	✓
	Hau et al. [32]	CPSS	✗	✗
	Inoue et al. [34]	IEEE ICDM	✗	✓
	Kim et al. [36]	CyberICPS	✗	✗
	Krotofil et al. [40]	AsiaCCS	✗	✓
	Lin et al. [44]	AsiaCCS	✗	✗
	Wolsing et al. [61]	ESORICS	✗	✓
	Yang et al. [64]	RAID	✗	✓
Knowledge-based	Adepu et al. [3]	AsiaCCS	Process Invariants	✗
	Adepu et al. [4]	IFIP	Process Invariants	✗
	Ahmed et al. [8]	AsiaCCS	State-Space Model	✗
	Ahmed et al. [9]	WiSec	State-Space Model	✗
	Palleti et al. [49]	J. Process Control	State-Space Model	✗
	Choi et al. [18]	CCS	Physical Model	✗
	Ghaeini et al. [29]	SAC	Physical Model	✗
	Quinonez et al. [51]	USENIX Sec	Physical Model	✗

It is difficult to judge whether data-driven or knowledge-based IIDSs yield better results due to inconsistent evaluation methodologies [27, 30, 42]. Yet, their competing advantages motivate us to combine their strengths in a system that can be configured automatically based on historical data, but is also generally applicable and delivers comprehensible alarms.

4 Design of GeCo

Manual approaches for constructing expert-based IIDSs do not scale with an increasing number of ICS components. Hence, the focus of this work is the development of a data-driven procedure that does not rely on domain experts but, at the same time, mitigates the current shortcomings of data-driven IIDSs (cf. Sec. 3). Capable of a precise prediction of ICS behavior, state-space models have long been considered a reliable method for system analysis [17, 35, 46]. In the context of industrial intrusion detection, they have been considered in proof-of-concept deployments [15], use-case-specific scenarios still with human input [51], or only with linear models [8, 15]. However, their broader scalability, generalizability,

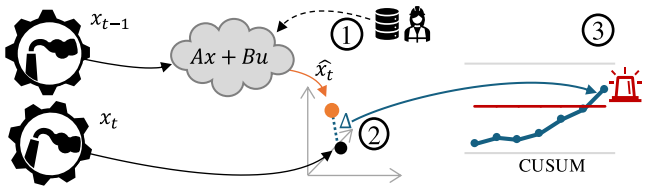


Figure 3: GECO combines expert-knowledge with data-driven techniques to find state-space models. Then, it analyses the difference between predicted $\hat{x}_t$ and observed system state x_t.

and explainability for IIDSs remain unexplored.

To close this research gap, we design GECO relying on the automatic data-driven derivation of state-space representations. Their precise modeling of system behavior combined with clear comprehensibility of its alert decisions, automated construction, and quick transferability among different ICS domains thus promises to alleviate the urgent need for flexible and easily deployable cybersecurity solutions [1, 45].

4.1 High-Level Design Overview

As sensors and actuators of ICSs measure values belonging to correlated physical processes, data by different components is frequently coupled to each other. Assuming that one component is compromised, i.e., its data deviates from normal patterns, this can be identified by calculating the expected data based on a subset of correlated components. The core idea of this work thus consists of expressing the benign behavior of every component in a given ICS as a linear or non-linear combination of suitable sensor measurements and actuator states. Based on this mathematical model, the future component behavior can be predicted and compared to currently observed measurements, thereby detecting compromises of one or more components. As all industrial processes abide to the laws of physics, this high-level approach promises to mitigate limitations w.r.t. to deployments in different scenarios.

Concretely, we propose GECO, which is based on automatic learning of a so-called *state-space model* [17, 35] describing the temporal behavior of an ICS's underlying physics. Our model is based on a set of generic *function templates*, which are fitted against historic ICS data to find the equation best describing the observed ICS behavior (cf. ① in Fig. 3).

Once learned, the state-space model can be employed to power GECO. First, the sensor state x_t of the ICS at time t is measured. Then, ② GECO predicts the state $\hat{x}_t$ based on the trained model and previously extracted sensor state x_{t-1} and actuator state u_{t-1}. GECO ③ keeps track of the cumulative distance Δ between the predicted and measured states in the weighted past. If this cumulative distance reaches a predefined threshold, an alarm is raised as the system has abnormally different behavior compared previous observations.

4.2 Modeling the Behavior of an ICS

GECO models the behavior of ICSs leveraging state-space representations. Generally, feedback flows within control loops determine its future state and can be mathematically modeled with state-space models [17, 35]. This model describes how the state of a system component evolves over time $t \in \mathbb{R}$. It depends on the observed state $x_t \in \mathbb{R}^n$ and the values of the control input $u_t \in \mathbb{R}^p$ of an ICS containing n measured physical values and controlling the process with p actuators (cf. Fig. 2). Data of sensors and actuators $y_t \in \mathbb{R}^n$ can be modeled as a combination of the system state x_t and the control input u_t. State-space models are generically defined by the following equation [17] that describes the observed state based on its current observation and the control input with two matrix functions $\mathbf{A} : \mathbb{R} \to \mathbb{R}^{n \times n}$, and $\mathbf{B} : \mathbb{R} \to \mathbb{R}^{n \times p}$:

$$x_t = \mathbf{A}x_{t-1} + \mathbf{B}u_{t-1} \tag{1}$$

The $\mathbf{A}$ matrix models the behavior of the physical process under undisturbed conditions. Depending on the dynamics of the physical system, such undisturbed conditions exhibit different results. E.g., while the water level of a tank remains constant without water inflow or outflow, the temperature loss of a heated system over time represents a system that changes without control input. A simplification is to model the change in undisturbed conditions by only depending on the current state. Then, the $\mathbf{A}$ matrix takes the form of a diagonal matrix:

$$\mathbf{A} = \begin{bmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{bmatrix} \in \mathbb{R}^{n \times n} \tag{2}$$

Here, a value of 1 e.g., describes the uncontrolled water level staying constant, while a value < 1, e.g., describes the decreasing temperature without external heating.

The $\mathbf{B}$ matrix, on the other hand, describes the influence of control inputs on the system. Accurately describing $\mathbf{A}$ and $\mathbf{B}$ enables the precise prediction of an ICS's behavior at any time. Consequently, any substantial deviation (besides noise or inaccuracies) from this prediction can be assumed to be an anomaly we aim to detect. We thus intend to automatically learn these state-space models on historical data logs without any manual engagement by domain experts.

4.3 Learning the State-Space Model

State-space models can be complex differential equations that may be generated by domain experts for IIDSs. Such manual effort quickly runs into cost and scaling limitations, both for small companies that lack expertise as well as larger companies with complex processes. The failure of a group of experts from Singapore's Public Utility Board to derive a system model for the SWaT testbed despite months of work shows how challenging this process is [59]. Conveniently, GECO only needs to predict changes over small time steps.

For the rest of this paper, we consider the state-space model for each process value individually and refer to the *state-transition function* of the i^{th} process value as F^i. The main idea of GeCo is thus to approximate these state-transition functions. The approximation of the **A** matrix in Eq. 2 gives us a state-transition function of the form:

$$F^i(x_{t-1}, u_{t-1}) \equiv \hat{x}_t^i = a_i \cdot x_{t-1}^i + B(u_{t-1})$$

$B(u_{t-1})$ is the change due to control inputs. Since interactions from control inputs can underlie various physical effects, we model these effects more precisely compared to our one-fits-all approach for the A matrix. Thus, we refer to a flexible set of *function templates* that are multivariant functions with any amount of control inputs u_{t-1} and parameters b.

To determine $B(u_{t-1})$ for a given process, GeCo exhaustively tests which function template best describes the state transition from x_{t-1} to x_t. Therefore, each potential function is fitted with parameters (a and bs) by the least squares optimization algorithm (we use SciPy's `curve_fit` function) and keeping track of the best fit according to the lowest mean squared error for the training data. The best function for each process value gives us the state-space model that we use during the attack detection phase.

For now, we only consider additive B^+ and multiplicative B^* function templates of the form:

$$B^+ \equiv B(u_{t-1}) = b_0 + \sum_{0<i<l\leq p} b_i \cdot u_{t-1}^{a_i}$$

$$B^* \equiv B(u_{t-1}) = b_0 + \prod_{0<i<l\leq p} b_i \cdot u_{t-1}^{a_i}$$

These two sets of function templates consider up to their length l many control inputs. We observe promising results even when limiting l to reduce the search space. Yet, the list of function templates can be expanded by domain-experts to more accurately model specific physical phenomena. They could even be pooled together by different experts. New function templates can, after all, only improve GeCo's performance at the cost of slightly prolonged training.

4.4 Two Illustrative Examples

GeCo learns the physical processes of an ICS based on function templates. To better understand how GeCo works, we review two illustrative examples: (i) the water level in the processes introduced in Fig. 1 and (ii) the water temperature in a tank with hot inflow as seen in the HAI [54] process.

Water Level. In particular, we look at the learned state-transition function for the `LIT101` sensor of the SWaT dataset [31] with a function length of 2 for the B matrix:

$$\hat{x}_t^{\texttt{LIT101}} = \underbrace{1.0 \cdot x_{t-1}^{\texttt{LIT101}}}_{A} + \underbrace{0.192 \cdot u_{t-1}^{\texttt{FIT101}} - 0.197 \cdot u_{t-1}^{\texttt{FIT201}} + 0.009}_{B} \quad (3)$$

The water level `LIT101` is described by the inflow and outflow meters, `FIT101` and `FIT201` respectively. The A-coefficient matrix of the mined function describes how the system behaves if no control command is issued, i.e., the water level remains constant if no flow is recorded at `FIT101` or `FIT201`. The B matrix describes how the water level increases or drops if the in- or outflow meters measure movement. The factors 0.192 and -0.197 are automatically derived by the mining algorithm and the unit measured by the flow meters is converted into a change of water level for the given tank size. Finally, we see a mostly negligible corrective summand b_0 that likely stems from inaccuracies and noise in training data.

Water Temperature. To see how additional function templates can improve a model's accuracy, we look at the approximated temperature of a tank with an inflow of heated water. Currently, GeCo approximates process linearly. However, we can easily extend the set of function templates with a template that describes the change in temperature of two mixed fluids (assuming a constant specific heat capacity):

$$B(u_{t-1}) = b_0 \cdot \frac{b_1 \cdot u_{t-1}^{a_0}}{b_2 \cdot u_{t-1}^{a_1} + b_1 \cdot u_{t-1}^{a_0}} \cdot (u_{t-1}^{a_2} - u_{t-1}^{a_3})$$

Here, $u_{t-1}^{a_0}$ is the volume of the inflowing fluid, $u_{t-1}^{a_1}$ is the volume of fluid in the tank, and $u_{t-1}^{a_3}$ and $u_{t-1}^{a_2}$ are the respective temperatures. Indeed, when adding this function template and training on the HAI, GeCo identifies the correct control inputs and finds fitting parameters to predict the temperature `TWIT04` of the tank `TK03` in the HAI testbed, where `TIT01` is the temperature and `FT01Z` is the flow rate of the incoming warm water, and `LIT01` is its fill level of `TK03`:

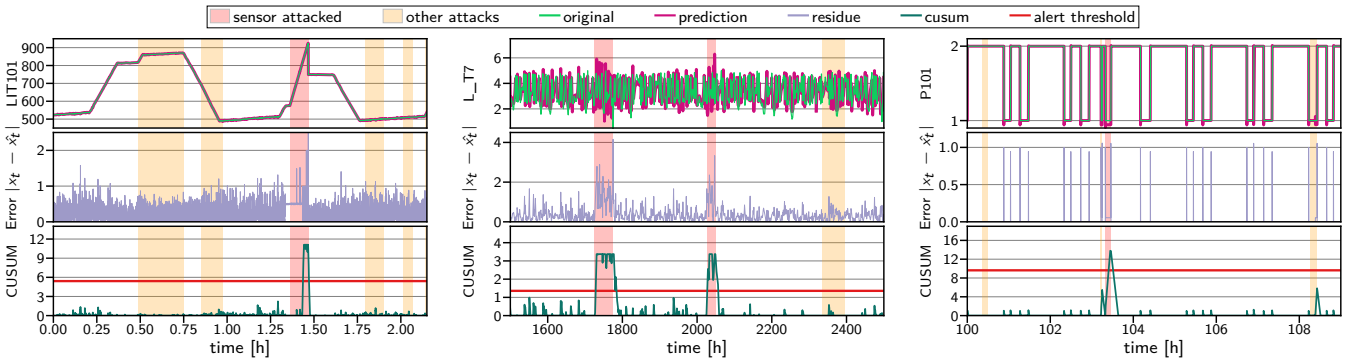
$$\hat{x}_t^{\texttt{TWIT04}} = \underbrace{1.0 \cdot x_{t-1}^{\texttt{TWIT04}}}_{A} + \underbrace{\frac{0.00217 \cdot 9.70 \cdot u_{t-1}^{\texttt{FT01Z}}}{22.6 \cdot u_{t-1}^{\texttt{LIT01}} + 9.70 \cdot u_{t-1}^{\texttt{FT01Z}}} \cdot (u_{t-1}^{\texttt{TIT01}} - u_{t-1}^{\texttt{TWIT04}})}_{B}$$

Among all the possible combinations, the learned state-transition functions for `LIT101` and `TWIT04` explain the physical process similarly to how an expert may have described the system [9]. Importantly, GeCo does not require an expert to describe the system dynamics. The automatic learning of state-space representations can uncover complex correlations across different parts of the ICS that even domain experts may not be aware of, making anomaly detection more robust.

4.5 Online Attack Detection with GeCo

The trained model is used to perform anomaly detection by analyzing the deviation of real-time system data from the predictions. To this end, GeCo periodically extracts the current state of the system x_t, and the control inputs u_t. Together with the learned state-space model, GeCo computes a prediction for the next system state $\hat{x}_t$ as described in Sec. 4.3.

For anomaly detection, GeCo compares the predicted state $\hat{x}_t$ and the later measured actual physical state x_t by calculating the absolute residue $\|x_t - \hat{x}_t\|$. Aiming to make the detection more robust, we not only consider the step-wise residuals, but take into account their development over time as



(a) Example of an detected attack (red) by GeCo against the level meter LIT101 increasing its measurements by 1 mm every second.

(b) Example for the effective alert decisions of GeCo monitoring sensor L-T7 of the BATADAL dataset displaying noisy behavior.

(c) Example of GeCo monitoring the discrete values of the pump P101 which was forced to remain turned off by the attacker.

Figure 4: Applying our IIDS to three exemplary components, we find that GeCo correctly alerts not only attacks within the opening example discussed in Sec. 4.4, cf. Fig. 4a, but also demonstrates high detection confidence for components with volatile behavior and noisy residues, cf. Fig. 4b, as well as components working only with discrete values, cf. Fig. 4c.

recommended in related work [51, 58]. To this end, we apply the change detection algorithm CUmulative SUM (CUSUM), which accumulates the deviations relative to a specific target mean over time and warns whenever the accumulated sum exceeds a predefined threshold. This way, small gradual deviations are better picked up and errors within our predictions introduced by noise or inaccuracies in the modeling and mining step are compensated for. We again consider state-transition functions F^i individually for the CUSUM calculation as they may have different scalings and sensitivities to influences from the outside, resulting in the following definition:

$$\begin{aligned} CUSUM_0^i &:= 0 \\ CUSUM_t^i &:= max(0, CUSUM_{t-1}^i + \|x_t^i - \hat{x}_t^i\| - \delta^i) \end{aligned} \tag{4}$$

$CUSUM_t^i$ iteratively computes the cumulative absolute prediction error of F^i up to the time t. In each step, this value is corrected by the drift δ^i, which expresses a constant but expected drift tendency. This drift δ^i is set during the training phase as the mean residue plus one standard deviation. Next, T^i is the *detection threshold* for the state variable x^i that determines when an alert should be raised. For T^i, we use the maximum CUSUM value observed during training and consider a safety margin expressed by a scaling factor S.

$$T^i = S * \max(CUSUM_t^i) \tag{5}$$

Lastly, and in line with common practice [58], we limit the absolute growth of the CUSUM score to prevent massive overshooting of the threshold and long recovery periods to the area below the threshold, especially if δ^i is small. Thus, a second parameter G (growth factor) restricts the maximum $CUSUM$ value to $min(CUSUM_t^i, T^i + G \cdot \delta^i)$.

In summary, GeCo computes the $CUSUM^i$ of the deviations between the predicted $\hat{x}^i$ and the measured value x^i at each time step for every process value. If the $CUSUM^i$ value of any state variable surpasses the predefined threshold T^i, GeCo assumes anomalous behavior and raises an alert until the value has fallen below the threshold. These alerts not only inform the operator of anomalous activities but also provide context information to help locate the source of the disturbance as each sensor is monitored individually.

4.6 GeCo in Practice

To summarize the construction procedure, we demonstrate GeCo's operations as a well-functional IIDS. To this end, we show the alert behavior for three examples from diverse data types (linear, noisy, and binary data) to provide evidence that GeCo can function in all these situations in contrast to related work usually focusing on a single, linear use-case [15].

Starting with Fig. 4a, GeCo is applied to the level meter `LIT101` (cf. Sec. 4.4), while the testbed is under attack. This also includes one direct attack against `LIT101`, steadily increasing its measurements by 1mm per second (cf. red area in Fig. 4a). GeCo successfully registers this compromise and correctly alerts during the ongoing attack. This is initiated by higher than normal residues between predictions and real data (cf. middle plot). The final alert is later launched by the corresponding CUSUM values distinctly overshooting the predefined threshold (cf. red line). The alert precisely overlapping with the relevant attack shows that GeCo quickly and precisely unveils the `LIT101` sensor's abnormal behavior. Meanwhile, no alert is raised while other system components are under attack (cf. yellow areas), such that GeCo also helps in understanding and localizing the source of anomalies.

To demonstrate that GeCo also correctly alerts if the behavior of a component is less foreseeable, we examine the sensor `L-T7` included in the BATADAL dataset [55]. This

sensor monitors the water level in a tank of a municipal water distribution network with just a single measurement per hour. Thus, the residues between predictions and real data are much noisier (cf. Fig. 4b). Still, both attacks targeting `L-T7` are identified by GECO as both the residues and the CUSUM values are deviating clearly from their norm. Even though the value dispersion among the residues is much higher making it more difficult to detect attack. This confirms the assumption from Sec. 4.3 that even a model with a lower level of precision in the predictions suffices for accurate attack detection.

Lastly, we consider a discrete data example with the pump `P101` in Fig. 4c. This example showcases the need for the CUSUM algorithm instead of deriving alert decision directly from the calculated residues. While the momentary deviations between predictions and real data plotted in the middle subplot of Fig. 4c do not allow a definite revelation of anomalous behavior produced by the attack against `P101`, the CUSUM mechanism results in a successful attack detection. Therefore, we conclude that CUSUM, as a stateful change detection algorithm, is a necessary design element of GECO.

4.7 Data Input required by GECO

To showcase the generalizability of GECO to any given domain and use case, we further refine the input GECO needs to train a model and perform online attack detection. Similar to other data-driven IIDSs, GECO works fully automatically. The only required information that must be included in training and live datasets are historical time series of process values with regular time intervals. The amount of data varies across processes, as we later show in Sec. 6.3. To make our implementation easily accessible, we implemented GECO on top of the IPAL framework, providing a proven standardized format of such time series data [63]. If GECO is supplied with these time series of process values and suitable hyperparameters, GECO automatically infers the relationships between the process values during training without any outside assistance. This sets GECO apart from knowledge-based IIDSs that require manual and time-intensive labour by ICS operators to derive meaningful invariants [3, 4] or state-space models [9, 49, 51] (cf. Tab. 1). During live operation, GECO then also only requires state snapshots, i.e., a list up-to-date process values, in regular time intervals.

5 Implementation & Evaluation Setup

Bringing the theoretical design of GECO into practice, we sketch our implementation in Sec. 5.1 and shape our evaluation setup to assess the capabilities of GECO in Sec. 5.2.

5.1 Implementation Details

As our implementation platform, we select the Industrial Protocol Abstraction Layer (IPAL) framework [63]. IPAL's core idea is to facilitate coherent IIDS research by offering a generic data format with access to plenty of common datasets and validated (re-)implementations of prominent IIDSs from related work. Internally, IPAL is based on Python, and thus, we leverage the `curve_fit` function from the SciPy library to find suitable parameters for the functions defined in Sec. 4.3.

One aspect assumed in Sec. 4.3 is that it is well-defined which process values belong to the observed state x_t and which to the control input u_t. To keep the input from external experts minimal, we relax from that assumption and simply test all combinations. Also, during training of the state-space model and the CUSUM detection threshold, we use the first 80 % of the training data to learn the state-space model and use all 100 % of the data to calculate the CUSUM's drift and detection threshold. Thereby, we avoid overfitting the CUSUM detection method to data on which the state-space model was trained. Appx. A.1 provides a sketch of the code.

5.2 Evaluation Setup

For our evaluation setup, we now outline our decisions for the utilized datasets, IIDSs from related work compared against, performance metrics, and hyperparameter decisions.

Datasets. To obtain meaningful results [19], we apply GECO to four diverse, publicly accessible, and commonly used datasets in IIDS research, namely SWaT [31], WADI [6], HAI [54], and BATADAL [55]. All datasets ship with at least one attack-free part, which we use for training. The remaining parts contain varying numbers of attacks: 36 attacks in SWaT, 14 for WADI, 50 for HAI, and 14 for BATADAL.

IIDSs. To cover a wide range of IIDSs for comparison, we rely on the IPAL framework providing various IIDSs from related work. We select TABOR [44], SIMPLE [61], Seq2SeqNN [36], PASAD [12], and Invariant [26] as those are prominently published and openly accessible (cf. Tab. 1). *TABOR* [44] is an IIDS that divides process-data into linear segments to derive timed automata which are complemented with a Bayesian network to incorporate the relationship to actuators and an out-of-bounds check for miscellaneous data. In contrast, *SIMPLE* aims to reduce complexity to a minimum [61], showing that a combination of four straightforward heuristics suffices to perform on par with far more complex IIDSs. *Seq2SeqNN* [36], similar to GECO, predicts the next measurement and alerts deviations, yet based on neuronal networks. *PASAD* [12] utilizes singular value decomposition and calculates the difference to vectors seen during training. Lastly, Invariant [26] mines state-invariants as boolean statements that must always be fulfilled. These IIDSs resemble a broad collection of relevant publications as they were designed for at least one of the datasets and were already subject in comparison studies [24, 27, 61, 62]. Since these IIDSs were evaluated originally on a subset of the considered datasets, we applied them to the other datasets where necessary.

Metrics. To quantify the detection performance, we use

Table 2: GECO performs above average compared to relevant related work and across many metrics and datasets. Cells in grey resemble the best-performing IIDS for a given dataset and metric. The performance of related work was measured with the re-produced implementations from the IPAL framework to obtain numbers for all metrics and datasets. The scarcely stated performance results for related work are added in brackets if provided in the respective original publications.

	IDS	Prec.	Rec.	F1	eTaP	eTaR	eTaF1	FPA	Scen.
SWaT	GECO	94.8	79.0	86.2	83.1	60.7	70.2	4	86.1
	SIMPLE	70.7 (71.0)	86.7 (87.0)	77.9 (78.0)	58.7	47.2	52.3	18 (23)	75.0 (75.0)
	TABOR	81.5 (86.2)	74.7 (78.8)	77.9 (82.3)	49.1	18.9	27.3	27	55.6 (66.7)
	Invariant	97.3	69.1 (78.8)	80.8	54.7	29.8	38.6	182	86.1 (91.7)
	Seq2SeqNN	44.0	10.9	17.5	42.8	47.2	44.9	36 (20)	75.0 (80.6)
	PASAD	32.4	71.5	44.6	16.0	4.9	7.5	14	44.4
WADI	GECO	92.6	32.1	47.7	91.3	56.3	69.7	0	78.6
	SIMPLE	58.2 (58.0)	43.6 (44.0)	49.8 (50.0)	57.0	52.1	54.4	8 (9)	64.3 (64.3)
	TABOR	19.1	43.7	26.6	14.9	13.0	13.9	3	57.1
	Invariant	90.0	21.9 (47.4)	35.2	92.3	32.6	48.1	2	42.9 (100.0)
	Seq2SeqNN	44.4	13.4	20.5	45.4	31.3	37.1	7	64.3
	PASAD	16.4	23.9	19.5	5.4	4.3	4.8	3	35.7
HAI	GECO	75.4	55.5	63.9	73.8	65.4	69.4	8	90.0
	SIMPLE	87.0 (87.0)	39.8 (40.0)	54.7 (55.0)	86.0	61.5	71.7	4 (26)	88.0 (86.0)
	TABOR	4.8	45.1	8.7	0.0	0.0	0	4	46.0
	Invariant	77.1	9.1	16.2	79.2	25.2	38.3	12	50.0
	Seq2SeqNN	8.5	4.6	6.0	7.7	2.9	4.2	14	26.0
	PASAD	3.3	12.7	5.3	1.0	2.0	1.3	51	16.0
BATADAL	GECO	93.8	73.4	82.3	97.0	88.1	92.4	0	100.0
	SIMPLE	52.0	43.3	47.2	49.0	42.8	45.7	4	71.4
	TABOR	78.5	6.9	12.7	77.7	14.3	24.1	2	14.3
	Invariant	27.2	45.5	34.0	18.2	74.9	29.3	865	100.0
	Seq2SeqNN	34.2	5.6	9.6	27.0	6.9	11.0	1	14.3
	PASAD	20.1	52.1	29.1	10.5	21.5	14.1	32	78.6

Precision, Recall, and the F1-Score, as these are the most commonly used metrics [42]. However, as they received criticism in recent years if leveraged in time-aware IIDS research [33], e.g., since long attacks are over-weighted by these scores, we also consider their newer time-aware pendants eTaP (Precision), eTaR (Recall), and eTaF1 (F1) [33]. Additionally, we state the fraction of detected scenarios (Scen.), i.e., attacks from each dataset, and the number of continuous false-positive alarms (FPA). Note that we count an alert only as a false positive if it is at least 60s away from an attack since an attack's effect may persist even shortly after the attack's end, resulting from inaccurate labeling for some datasets [61].

Hyperparameters. GECO has just hyperparameters. The maximum function length (k) is always set to $k \leq 3$. The hyperparameters for CUSUM, the scale factor S and drift factor G, are set individually for each dataset (cf. Appx. A.3).

Overall, our setup meets the standards stated in the literature [42] as we evaluate four datasets and compare our results with modern metrics against five IIDSs. In comparison, the entire research domain evaluates just 1.3 datasets on average and compares against 0.5 other approaches on average [42], which sets our evaluation of GECO significantly apart.

6 Evaluation of GeCo

While GECO detects attacks in various data types in Fig. 4, we now assess whether GECO holds up to its claims to achieve a middle ground between required expert knowledge, generalization, comprehensibility, and automation. First, Sec. 6.1 considers its detection performance. Next, Sec. 6.2 concerns the comprehensibility of GECO and its alerts. Lastly, Sec. 6.3 measures the computational performance to judge the tradeoff between time invested by the computer and the expert.

6.1 Detection Performance

We now analyze GECO's detection performance in a threefold analysis. First, we compare GECO to state-of-the-art related work in Sec. 6.1.1. Sec. 6.1.2 analyzes GECO's generalizability to other industrial domains and we finish with a comparison to knowledge-based approaches in Sec. 6.1.3.

6.1.1 GeCo Compared to Related Work

One of the most important aspects defining the significance of an IIDS is its detection performance [13], i.e., its capability to detect the majority of cyberattacks whilst emitting few false-positives [10]. To this end, we now discuss the results objectively with metrics in Tab. 2 and visually along their alerts displayed in Fig. 5. As the algorithm of GECO is deterministic, a single snapshot of its performance suffices.

While GECO is not perfect, it outperforms existing IIDSs in 23 of 32 metrics across all datasets (cf. grey cells of Tab. 2). Most notably, on BATADAL, our approach performs optimally detecting all attack scenarios with zero false positives, a currently unmatched performance. Also, on SWaT, the most prominent evaluation dataset [42], the results are outstanding, especially since it detects the most attacks, together with Invariant, but with the fewest false positives. GECO only falls

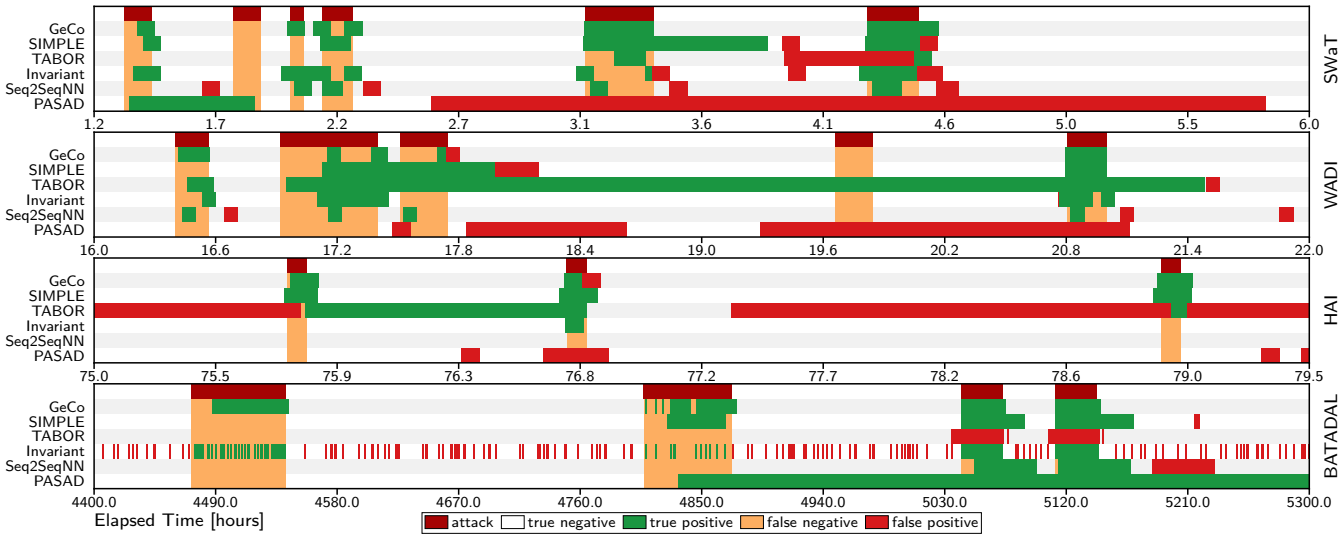


Figure 5: GECO precisely indicates the attacks, which can be difficult to infer from other IIDSs as their alerts do not overlap as consistently with the attacks as GECO. The whole image of all alerts is contained in the appendix in Fig. 12.

behind in precision and recall on SWaT, which are contradictory, valuing either the most detected attacks (recall) or correct alerts (precision). Thus, it is unsurprising that one IIDS from related work is better for each metric. Noteworthy, GECO beats all IIDSs in the F1 score on SWaT, which mediates between these two extremes. On WADI, we exceed in eTaF1 and again have zero FPA whilst detecting the most attacks. On HAI, GECO performs over average compared to related work. Here, GECO again detects the most scenarios with few FPA while being dominant in the F1 score and being just 2.4 percent points behind SIMPLE in eTaF1. In the end, ICS operators have to decide which characteristic of an IIDS expressed through the different metrics best fits their needs [42], of which GECO can satisfy a broad range.

For this reason, we also take a visual approach by looking at the individual alerting behavior of GECO and related work. Fig. 5 depicts an excerpt of each dataset (the complete figure can be found in the appendix in Fig. 12). GECO precisely indicates the ranges of the attacks in the datasets with only little deviation and, especially during benign behavior of the ICS, no apparent false positives are raised. The two false positives of GECO visible in this example in WADI at around 17.8 and HAI at about 76.8 occur shortly after the attacks finished and are thus likely still related to fluctuations in the data from the attack. In contrast, the alerts from related work are not as consistent throughout all datasets, which highlights GECO's ability to function well in all these scenarios.

Lastly, we take a look at attacks that are not detected by GECO (false-negatives). On SWaT, six attacks are missed. SIMPLE also misses all these attacks, showing that GECO does not fail to identify easily detectable attacks [61]. Furthermore, five of GECO's six false-negatives on SWaT are neither detected by TABOR, Seq2SeqNN, nor PASAD. Across all datasets, only five false negatives of GECO are detected by another IIDS. But, while Invariant detects two false negatives of GECO on SWaT, Invariant also has many false-positives. Additionally, while Invariant and Seq2SeqNN detect the first attack in WADI, this only happens at the end of the attack when the system is stabilizing again (cf. Fig. 12). Thus, judging whether GECO should also be able to detect all false-negatives is difficult as (i) other IIDSs might detect those only by chance and (ii) datasets contain attacks that are reported as non-detectable [36]. Still, GECO shows no systematic deficiency compared to related work w.r.t. false-negatives.

GECO shows competitive performance, often better than related work. Notably, on two datasets, GECO has zero false positives and just 4 on SWaT, which is important as false-positives and alert fatigue are issues in practice [10].

6.1.2 Generalizability of GeCo to a New Domain

To show that GECO practically generalizes to another domain, we analyze the Tennessee Eastman Process [14]. This process is a chemical reaction modeling the mixing of substances and condensation of gasses to synthesize new products. We leverage the dataset recorded to evaluate PASAD [12], which includes 41 process values. We use 5.5h of the dataset for training and 6.7h for evaluation containing five attacks. Two attacks directly change a critical valve setting or pressure reading while the other three perform more subtly modifications to be more stealthy (cf. Sec. 3.1.1 in [12] for more details).

No changes to GECO were required, i.e., no additional function templates. We simply present GECO the training data and it learns a state-space model for this process. GECO raises an alert if the data of the test dataset deviates too much from the trained state-space model (cf. Sec. 4.5). The hyperparameters

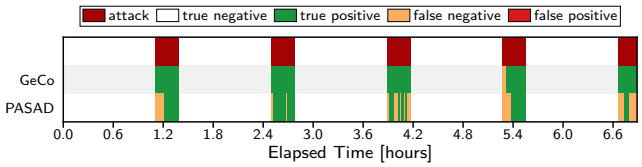


Figure 6: GECO generalizes well to a new chemical domain.

Table 3: Compared to a knowledge-based IIDS leveraging invariant rules, GECO's performance is not inferior on SWaT.

SWaT dataset	Prec.	Rec.	F1	eTaP	eTaR	eTaF1	FPA	Scen.
GECO	94.8	79.0	86.2	83.1	60.7	70.2	4	86.1
Knowledge-based	92.1	69.2	79.0	75.7	28.7	41.6	3	52.8

were set to default values (cf. Tab. 6).

As shown in Fig. 6, GECO detects all five attacks instantaneously without any false positives. We achieve a F1 score of 96.0 and an eTaF1 score of 98.0. In contrast, PASAD detects the attacks later. Given that no manual effort was required to apply GECO to this scenario, this experiment nicely shows its ability to generalize to new ICS domains.

6.1.3 Comparison to an Experts' Knowledge-based IIDS

GECO compares highly competitively to data-driven IIDSs. However, given that GECO targets to supersede the hard manual work of experts, it is equally essential that this high level of automation does not results in a penalty of detection capabilities compared to manually created knowledge-based IIDSs. In contrast to data-driven IIDSs, where implementations and detailed evaluation results are readily available [63], obtaining similar input for a comparison with knowledge-based IIDSs is challenging. Either related work does not state any metrics [3,4,29], evaluates their IIDSs in practical experiments [18,51], or considers selected subparts of datasets [8,9].

To compare GECO to knowledge-based IIDSs, we consider publications that propose expert-created invariants (boolean equations) that must always be satisfied for the SWaT dataset. Knowing, for example, that `FIT101` measures the inflow after the valve `MV101` (cf. Fig. 1), we can formulate the invariant `MV101` open$\Rightarrow$`FIT101`$> x$. We identified a total of six publications that provide manually crafted rules for SWaT [3,26,48,49,57] and a list of invariants by the SWaT dataset creators [2]. We collected and implemented 63 invariants, of which we omitted 16 due to a high number of false alerts and 14 because they were duplicated. In total, this leaves us with 33 invariants (listed in Tab. 8), which we combined into a knowledge-based IIDS (implementation available in artifacts). These invariants do not require training and the IIDS raises an alert whenever any invariant is violated.

We measure the performance of the knowledge-based IIDS on the SWaT dataset and compare it to GECO in Tab. 3 observing similar performance for both approaches. While the knowledge-based IIDS detects fewer attacks (Scen.), it yields one less false positive. Overall, when experts are being replaced with GECO, we obtain a detection performance that is on par with labor-intensively created knowledge-based IIDSs.

6.2 Model and Alert Comprehensibility

Besides detection performance, the comprehensibility of emitted alerts are equally crucial in ICS applications [25]. First, a transparent IIDS model simplifies supervision and prevents unintentional overfitting, as the operator can verify whether the fitted model actually resembles the physical process. Second, an IIDS has to provide a degree of transparency that allows ICS operators to comprehend why the IIDS raised an alert [61]. Yet, effectively understanding and validating alerts in retrospect can be cumbersome for machine learning models [28]. Comprehensibility also eases incident response as the cause for the fault becomes apparent. We analyze GECO's comprehensibility on a theoretical level in Sec. 6.2.1 and by conducting a preliminary user study in Sec. 6.2.2.

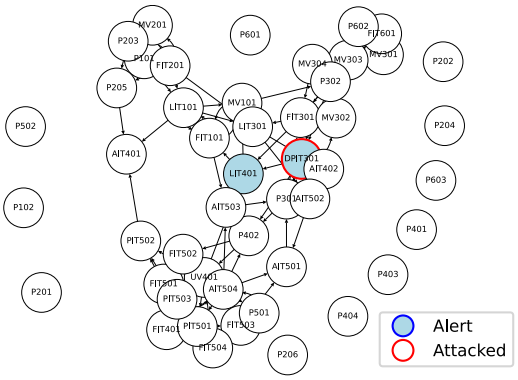
6.2.1 Localizing and Understanding GECO's Alerts

Model comprehensibility has already been discussed earlier in Sec. 4.4. There, the IIDS discovered equations that accurately reflect the datasets' physical process. Thereby, operators with knowledge about the concrete ICS can verify the trained model and correct it if necessary.

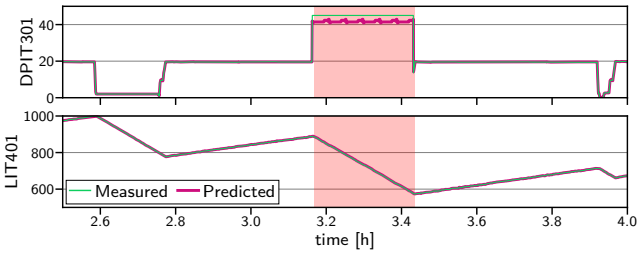
Now, we focus more on understanding the alerts. As our methodology, we show how an operator could trace the cause of an alert along with one exemplary attack. We consider attack number eight of the SWaT dataset (cf. red attack for SWaT in Fig. 5 at 3.1h). Here, the attacker targets a pressure meter `DPIT301`, setting it to a higher value.

The first step in investigating an alert is determining which part of the ICS is under attack. GECO makes this simple since we model each process value with exactly one equation. Thus, our intuition is that the attacked location can be narrowed down by the equation(s) that diverge. E.g., an attack modifying the `LIT101` value likely violates the corresponding equation (cf. Eq. 3), as demonstrated in Sec. 4.6, or an equation that depends on `LIT101`.

We now validate this intuition along with an example depicted in Fig. 7. We start by drawing the dependencies between the equations in Fig. 7a. The nodes represent the different process values. We draw an edge between two nodes x1 and x2, if the state-transition function of x1 depends on x2. E.g., the node for `LIT101` is connected to `FIT101` and `FIT201` (cf. Eq. 3). The graph is arranged by the Fruchterman-Reingold force-directed algorithm, drawing related variables closer together. Nodes not connected to the graph are not dependent on other values. We also mark where the attack took place (red border), here `DPIT301`. We highlight the process values that deviate from their predications and lead to alert



(a) GECO's alerts enable inferring where an attack took place as nearby dependent process values indicate the presence of an attack in that region. Here, we show attack 8 of SWaT. Videos covering the complete timespan of all datasets are available on Zenodo.



(b) Having identified the potential attack points, a closer investigation of the predicted and measured behavior gives further insights into the attack. Here, DPIT301 behaves differently than predicted.

Figure 7: GECO enables operators to analyze attacks quickly without requiring a profound machine learning background. Here, the attack increased the value of DPIT301 to a fixed value above 40 [31], beyond the standard operating limit.

by GECO (blue circle). From this dependency graph, we can derive that the attack likely targeted `LIT401` or `DPIT301` as both predictions deviated from the observed behavior. This narrows the investigation to a small part of the ICS.

This methodology is just the first step to locating an attack. Next, we consider the measured and predicted values for `DPIT301` and `LIT401` in Fig. 7b. For `DPIT301`, we observe a clear distinction between its predicted (red) and measured behavior (green) during the attack. First, the predictions are much noisier than prior to or after the attack, which hints at a misbehaving system. Secondly, the predictions presume a lower value for `DPIT301` than actually measured. Indeed, this coincides with the attack description, which explains that the attacker sets `DPIT301` to a higher value than normal (> 40). The alerting `LIT401` level meter then belongs to the tank after the pressure measurements and is thus also in close proximity to the attack. Overall, an ICS operator is able to correctly locate and comprehend an ongoing attack with GECO without requiring profound machine learning knowledge.

Fig. 8 shows that this methodology also works for other

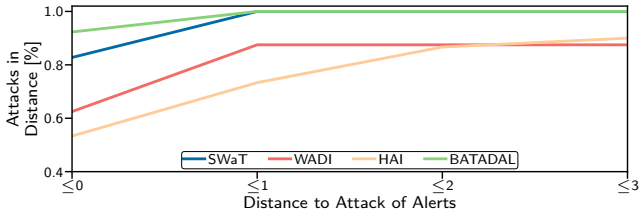


Figure 8: GECO indicates most attacks in short distances to the formulas that break according to the dependency graph, cf. Fig. 7a. Note that we omit unconnected values here, cf. P102 in Fig. 7a, as their distance to other nodes is infinity.

attacks. GECO's alerts are, on average, over all datasets and attacks 0.5 hops away from the attack point and often identical to the target. Thus, GECO facilitates comprehensibility as the alert usually coincides with the attacked process value.

Another aspect visible in Fig. 7a is that most process values form a connected graph. Thus, more complicated or stealthy attacks modifying multiple process values at the same time are likely to be picked up by more than one equation. The SWaT, WADI, and HAI datasets already contain such multi-point attacks that GECO successfully detects. Lastly, since the graph is mostly connected, an attack would need to spoof nearly all process values at the same time and correctly to each other's dependencies to stay hidden. This is a clear benefit of GECO in contrast to knowledge-based IIDSs of related work often modeling the ICS just partially (cf. Sec. 3).

Altogether, GECO's model and alerts are comprehensible as they closely correlate to the ICS's underlying physical process. In contrast, works such as Seq2SeqNN provide a single suspicion score for the entire process derived from a neuronal network's decision, and the comprehensibility of machine learning IIDSs is generally limited [28].

6.2.2 ICS Engineers' View on GeCo

GECO promises comprehensible and actionable alerts. To support our theoretical analyses, we perform qualitative interviews with experts from various industrial domains, similar to Fung et al. [28]. We aim to assess the comprehensibility of GECO's alarms, i.e., to what extent attacks are understood, and resulting actionability from ICS engineers' point of view.

Study Design. At the start of each structured interview, we asked the participant to name the industry domain they are working in and to describe their role. Afterward, we presented the basic industrial process from Fig. 1, which was chosen as the process can be understood rather quickly. Participants were then introduced to the context-enriched GECO alarms. During the interviews, participants were tasked to analyze four scenarios from the SWaT dataset, three attacks constrained to the investigated subprocess (3, 21, 30) and one false alarm. A separate attack (33) is used to introduce participants to GECO. Participants were presented with the resulting alarms

Table 4: Backgrounds of interview participants.

Participant	Industry Domain	Role
P1	Energy	Testbed Engineer
P2	IT Security	Consulting
P3	Engine Manufacturing	IT/OT Interface Engineer
P4	Maritime Manufacturing	IT/OT Software Developer
P5	Energy	OT Network Planning
P6	Automotive	Secure Product Development

of GECO in randomized order and tasked with explaining the attack and how they would react. We compared their answers to the actual attack descriptions of SWaT [31]. At the end of each interview, we asked participants about their general satisfaction with the context information provided by GECO, especially the identification of relevant sensors. Our study design follows the guidelines of our institutional ethics committee and is exempt from ethics review (cf. Ethics Considerations). On average, interviews lasted 35 minutes.

Participants. We contacted potential participants with experience in OT environments from our professional and personal networks. In total, we recruited 6 participants from various industry domains, as listed in Tab. 4. Interviews were conducted virtually and in person. The participants were not compensated for their participation.

Results. We start by assessing the comprehensibility of GECO's alerts. Overall, the participants were able to identify the origin of the attacks with high accuracy as only one of the true positives of 18 scenarios was misinterpreted. The value-specific alerts of GECO with low detection delays were key to narrow down the search (P1, P2, P5, P6). This coincides with our theoretical analysis that GECO identifies those values directly involved in or influenced by the attack (cf. Fig. 7a). The false positive was correctly identified by three participants (P1, P5, P6) while two others stated that they can not find an attack (P2, P4). Later, P2 acknowledged that "there will always be false positives... [But] if you are operating such plants, you will likely know whether a certain behavior is expected and could identify this as false positive".

All participants derived suitable mitigation measures or test procedures for further investigation of all attacks, including checking the real water levels or turning pumps or valves on and off to validate that the water levels are tracked as expected. We can thus conclude that GECO generates actionable alerts.

Regarding the quantity of the presented data, the selection of the relevant process values by GECO was perceived as helpful as there was "not too much information and [information] was cut away where not necessary"–P3. The way how the information was presented and pre-selected by GECO helps "increase the reaction speed" since it would otherwise "take longer to obtain an overview of [all] the data"–P5.

Overall, our interviews emphasize GECO's advantages. Three participants even proactively asked whether we are willing to test GECO in their OT environments, highlighting the demand for comprehensible intrusion detection for ICSs.

Table 5: Computational performance demanded by GECO for training and live detection. Training times are normalized to a system with 128 CPU threads.

Dataset (Length, n, k)	Tests Eq. 6	Training Total	Training One Fit	Live
SWaT (495k, 48, 3)	1.77M	15.1h	3.92s	0.2ms
WADI (784k, 122, 3)	73.87M	46.9d	7.02s	0.1ms
HAI (216k, 72, 3)	8.97M	1.3d	1.64s	0.9ms
BATADAL (8k, 42, 3)	1.04M	3.6m	0.03s	1.7ms

6.3 Computational Performance and Training

We are furthermore interested in the computational complexity of GECO, which encompasses two dimensions. The first dimension concerns the live detection phase. On a server equipped with two AMD EPYC 7452 CPUs with 64 cores and 236G RAM, GECO classifies state snapshots often in a fraction of 1ms, including the data-parsing overhead of the IPAL framework, as shown in Tab. 5. Thus, GECO can keep up with real-time critical applications.

The second dimension concerns the training phase. We begin with a theoretical analysis of the search space for the state-space model. For an ICS with n process values, t function templates with lengths of up to k, the number of combinations in the search space is given by (cf. pseudocode in Appx. A.1):

$$\texttt{combinations} := t \cdot n \cdot \sum_{i=0}^{k} \frac{n!}{(n-i)! \cdot i!} \tag{6}$$

We see that adding new function templates (t) impacts the number of combinations linearly, whereas the facultative impact of n on the complexity is potentially concerning. The other hyperparameters (scale and drift factor) do not affect the computational performance. To analyze practicality, we measured the training time on each dataset, as shown in Tab. 5.

Training times vary drastically between 3.6m for BATADAL and 46.9d for WADI. Except for WADI, this is comparable to complex IIDSs from related work, such as Invariant or Seq2SeqNN. Seq2SeqNN was trained on six GPUs [36], highlighting the immense hardware requirements. Due to a different architecture, Seq2SeqNN's two hours of training time are difficult to compare to our measurements. On the contrary, training Invariant on the same hardware takes on average 9.5h for SWaT, while GECO requires 15.1h.

WADI, as the largest dataset with the most process values, represents an upper limit for GECO's feasibility. Alleviating this situation, training has to be done only once and must not be repeated for hyperparameter tuning since this step is performed afterward (cf. Appx. A.1). Furthermore, it is possible to reduce the search area by restricting the search to components that are in close proximity to each other.

In the following, we analyze the impact of the amount of training data on *(1)* the training time and *(2)* the accuracy of the learned model. Therefore, we gradually increase the

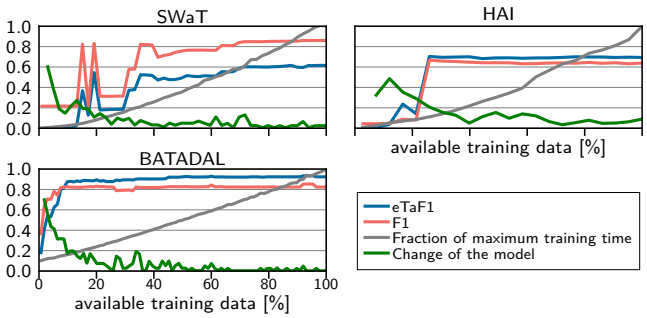


Figure 9: Decreasing the training data of GECO has little impact on the detection performance, but significantly reduces the training time. The change of the learned state-space model can help estimating when the training can be stopped.

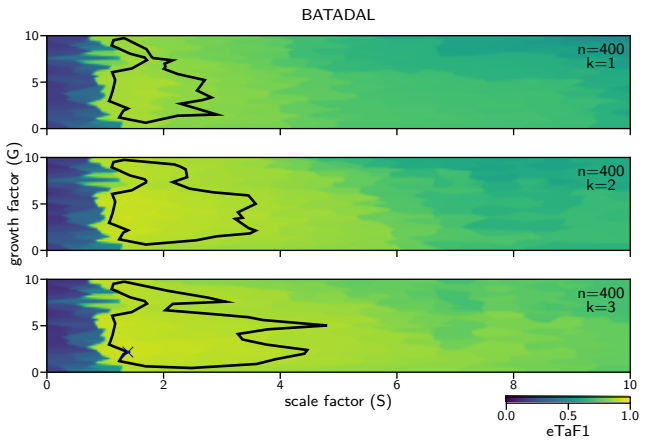


Figure 10: Finding suitable hyperparameters for the GECO promises to be an easy task since the approach shows good performance over wide areas. The hyperparameters used for our evaluation are marked by the × and the black line indicates the area within 5% eTaF1 of the maximum.

amount of training data accessible to GECO while recording the training time. We then evaluate the learned model against the entire test dataset. We performed this analysis for all datasets except for WADI due to its high training demands (cf. Tab. 5). GECO usually keeps its good detection performance even if the training data is reduced drastically (cf. Fig. 9). E.g., for BATADAL, the training time increases nearly linearly with the amount of training data such that with 10% of the training data, about 84% of the training time can be saved. Meanwhile, detection performance is only reduced by 0.002 (0.048) in F1 (or eTaF1). We see similar potential to reduce training time without impacting detection performance for SWaT and HAI.

While this property relaxes the issue of prolonged training, operators still need to understand when to stop training. First, we can infer from Fig. 9 that having more training data does not diminish detection performance. For more insights, we now access how the learned state-space models converge. To this end, we calculate the relative difference between two trained models as detailed in Appx. A.2. This methodology is motivated by scree plots used by PASAD [12] or loss curves from machine learning. When considering the change of the model over the amount of training data (cf. green lines in Fig. 9), we observe that saturation is reached if enough training data is provided. E.g., according to the green curve, GECO has finished training on SWaT after about 75% of the training data is used. This coincides with the moment when the F1 and eTaF1 curves (blue and red) stabilize. This metric thus provides a good estimate on GECO's training progress.

In Appx. A.4, we evaluate the influence of the training data's quality on detection performance. GECO yields good results even if the training data is of worse quality.

7 Hyperparameter Selection and Discussion

GECO is a potent IIDS with minimal required expert knowledge through maximal automation while providing good comprehensibility. Yet selecting hyperparameters, a common topic in machine learning [50], could complicate its usefulness in practice. Thus, Sec. 7.1 examines the ease of finding hyperparameters for GECO. Finally, Sec. 7.2 discusses further aspects for improvements and whether GECO has achieved a middle ground between data-driven and expert systems.

7.1 Influence of Hyperparameters

Finding hyperparameters in novel settings is a known issue in deploying anomaly detectors [7, 50]. Looking toward deploying GECO, an IIDS that requires complex hyperparameters to be tuned, risks yielding inferior performance in practice [27, 60–62], especially since ICS operators are not necessarily machine learning experts. While for established approaches such as Random Forests advice on how to tune their hyperparameters exists [50], this is not the case for a novel IIDS such as GECO. Thus, we aim to understand the influence of GECO's hyperparameters. We tackle this issue by i) limiting the number of hyperparameters per design (cf. Sec. 5.2), ii) transparently assessing how hyperparameters affect GECO's performance, and iii) by providing guidelines on how to choose hyperparameters and explain their effect to prevent exhaustive search in the following. In comparison, related work often disregards this topic entirely [15, 16, 32, 40, 44, 64].

We assess the influence ofGECO's hyperparameters as proposed by Wolsing et al. [62]. We extend our existing evaluation by examining GECO under randomly sampled configurations over all three hyperparameters, namely the growth factor G, the scale factor S, and the maximal function length k. For the two factors G and S, we examined a range between 0 and 10, and for the maximum function length k, we examined values from 1 to 3. Here, we consider the BATADAL dataset (the other datasets are depicted in Appx. A.3).

For BATADAL (cf. Fig. 10), there are large regions with "stable" performance, i.e., where the influence of the hyperparameters is minimal. Only if the scale factor is chosen too small or too high, approximately below one or above four, we observe diminishing performance. A higher function length seems beneficial for increasing the maximum performance since the regions marked with black lines, indicating the area within 5% of the maximum, enlarge from $k = 1$ to $k = 3$. On this dataset, little trends regarding the growth factor are visible. SWaT and WADI show a similar behavior, but for HAI a higher scale factor and function length are beneficial. In contrast, it was shown that some related work performs only optimally if multiple hyperparameters depending on each other are set adequately [27, 60, 62]. As we demonstrated, operators of GECO likely have to spend little time and resources on tuning these hyperparameters and can rather invest the time to validate and optimize the trained model.

Besides this analysis, we advise on how to parameterize GECO based on our experience. First, we recommend fixing $k = 3$ (if computationally feasible cf. Sec. 6.3) as it provides a good trade-off between training duration and detection performance. Once a first model has been trained, we recommend analyzing whether process variables exist for which GECO cannot find suitable equations. These can often be identified by false-positives and could either be initially ignored or tried to improve by adding more suitable function templates (cf. Sec. 4.4). Only then do we recommend fine-tuning the other two parameters. Modifying the scale factor (S) affects the sensibility of the alert threshold. Here, a higher value raises this threshold and leads to fewer alerts. From our hyperparameter analysis (cf. Fig. 10 and Fig. 11) we recommend initially setting S to about 1.5. The growth factor (G) can affect the length of alerts and smaller values shorten alerts if the CUSUM values decay slowly. Finding an appropriate G can be more complicated as we observed variance across our datasets (cf. Tab. 6). Yet, its impact on detection performance is usually smaller than that of S.

7.2 Future Improvements and Discussion

After demonstrating the capabilities of GECO as a generalizable and comprehensible IIDS, we now discuss to which extent we achieved a middle ground between required expert knowledge and automation. Furthermore, we identify potential for further improving this trade-off.

One potential for optimization, especially for larger ICSs with many process values, results from the current brute-force approach to learn the correlations (cf. Sec. 6.3). Here, leveraging experts could improve this situation, e.g., by guiding the search with domain knowledge. For example, informing GECO which process values are related can significantly reduce the search space. Nonetheless, reducing the search space through human input risks missing essential but non-intuitive dependencies between the process values.

Designing additional function templates is another avenue for improvements that only linearly impacts training time (cf. Eq. 6). GECO supports new function templates to more accurately describe complex physical processes (cf. Sec. 4.4). Still, GECO does not need to accurately describe all physical phenomenons, as (linear) approximations often suffice for intrusion detection, as demonstrated in various domains: water treatment (SWaT), water distribution (WADI and BATADAL), electrical (HAI), and chemical. Overall, GECO achieves better detection performance than many existing IIDSs, while it offers experts options for further improvements.

GECO thus shifts the optimization possibilities back towards the user of the IIDS, i.e., the operators of an ICS. In contrast, IIDSs from the machine learning domain try to improve results with technical solutions, often increasing complexity, such as varying neural network architectures [21, 38, 39] or stacking multiple classifiers [52]. While these improve detection performance on paper, they move further away from a usable and comprehensible IIDS. Overall, GECO enables ICS operators to flexibly decide how much expert-knowledge or automation is necessary to set up the IIDS instead of solely relying on either expert knowledge or automation, thereby achieving a new kind of middle ground.

8 Conclusion

In light of numerous cyberattacks against ICSs and critical infrastructures [11, 45], the pressure to secure them has gained increasing attention both by cybersecurity research [30, 42] as well as legislation [1]. This work concerns protecting industrial facilities by focusing on the second line of defense in the form of effective intrusion detection as a means to alert operators as soon as an attack takes place.

Existing works in this area can be divided into data-driven IIDSs, denying operators control over model training and impeding the understanding of alerts, and IIDSs that require significant manual effort by domain experts to construct the underlying models. This work aims to find a middle ground between these two extremes, combining the best of both worlds.

To this end, we propose GECO – a generalizable and comprehensible IIDS that automatically infers system knowledge with minimal human input. Using a set of generic state-space model templates, GECO automatically learns a system model of the ICS. Thereby, we remove the need for manual model design, as was required in previous works. At the same time, the underlying model remains comprehensible for ICS operators.

Besides showing a detection performance on par with state-of-the-art related work, GECO is able to retain high generalizability across various use cases while also being easy to comprehend in its alert decision process by human operators. Consequently, GECO paves the way towards giving the users of an IIDS control back over their tools to protect valuable physical processes in ICS and critical infrastructure.

Acknowledgments

We would like to express our gratitude to all experts who participated in our interviews on short notice. Moreover, we sincerely thank our shepherd and reviewers for their feedback on improving this work as well as Jan Bauer and Martin Serror for their help in preparing the expert interviews. Partially funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC-2023 Internet of Production – 390621612. This work is part of the project MUM2 and was partially funded by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) with contract number 03SX543B managed by the Project Management Jülich (PTJ).

Ethics Considerations

Discussing the ethical concerns among the authors, we could not identify any significant reasons for not conducting this work. The topic of intrusion detection is a defensive security mechanism that can be installed optionally and along existing or new deployments. Thus we do not put additional strain on companies to implement this technology. Moreover, publishing (new) IDSs is a common and ongoing topic in research. Nonetheless, by publishing details about and the source code of defense strategies, adversaries might learn about the detection capabilities of their target and thus adapt their attack strategies, e.g., to remain stealthy. However, we deem that leaving ICSs unprotected is more dangerous than publishing the underlying detection methodologies especially since security through obscurity or secrecy is not preferable in our opinion. Our analyses were conducted with the help of publicly available datasets. Thus, we do not disclose additional knowledge about ICSs that is not already known to the public. One group of stakeholders where our research might have a negative impact on are the experts and ICS operators in charge of deploying or working with IDSs. Since we claim with GeCo that their workload can be reduced through the automation of model generation, it could potentially obviate the need for such experts, in the worst case could lead them to being unemployed. However, as system experts are a valuable and scarce resource for companies, they might also get more time to focus on other essential tasks to improve security further. Likewise, as discussed in the introduction, small companies that currently have no expertise might obtain access to IDS methodologies through GeCo, which, in return, yields an overall benefit for the society through improved cybersecurity for our vulnerable critical infrastructures.

Next, we discuss ethical considerations w.r.t. the user study conducted in Sec. 6.2.2. Prior to getting in touch with any participants, we carefully followed the guidelines of our institutional ethics committee in designing our study. More precisely, we ensure that participation is voluntary, the results are stored anonymously, and we inform the participants about the expected methodology and the data collected in advance. During the (virtual) interviews, we only present pictures of the alerting behavior of GeCo (no violence, abuse, or questionable content). Based on this study design and the guidelines of our institutional ethics committee, our study is exempted from institutional ethics committee review.

Open Science

The artifacts of our research are available at https://zenodo.org/records/15120036. To make a valuable contribution to open science, we made the following considerations: First, we implement and evaluate GeCo with the help of the open-source IPAL IDS framework [63] intending to unify and assist research in this domain. Second, to bootstrap further research about GeCo, we distribute the precise hyperparameters and trained models for the analyzed datasets in this publication. Thereby, the training process can be avoided if researchers want to use GeCo for other evaluations. Lastly, we publish the alerts, i.e., the datasets classified by GeCo into benign and anomalous, as another artifact. This was proposed by Lamberts et al. [42] as a solution to mitigate misconceptions in evaluations, such as different definitions of metrics.

References

[1] Directive (EU) 2022/2555 of the european parliament and of the council of 14 december 2022 on measures for a high common level of cybersecurity across the union, amending regulation (EU) no 910/2014 and directive (EU) 2018/1972, and repealing directive (EU) 2016/1148 (NIS 2 Directive). https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32022L2555. Accessed: 2024-08-29.

[2] Invariants using Design-centri approach. https://itrust.sutd.edu.sg/wp-content/uploads/2017/08/comparison.pdf. Accessed: 2025-01-06.

[3] Sridhar Adepu and Aditya Mathur. Distributed Detection of Single-Stage Multipoint Cyber Attacks in a Water Treatment Plant. In *ASIACCS*, 2016.

[4] Sridhar Adepu and Aditya Mathur. Using Process Invariants to Detect Cyber Attacks on a Water Treatment System. In *ICT Systems Security and Privacy Protection*, 2016.

[5] Ekta Aggarwal et al. CORGIDS: A Correlation-Based Generic Intrusion Detection System. In *CPS-SPC*, 2018.

[6] Chuadhry Ahmed et al. WADI: A Water Distribution Testbed for Research in the Design of Secure Cyber Physical Systems. In *CySWATER*, 2017.

[7] Chuadhry Mujeeb Ahmed et al. Challenges in Machine Learning Based Approaches for Real-Time Anomaly Detection in Industrial Control Systems. In *CPSS*, 2020.

[8] Chuadhry Mujeeb Ahmed et al. NoisePrint: Attack Detection Using Sensor and Process Noise Fingerprint in Cyber Physical Systems. In *ASIACCS*, 2018.

[9] Chuadhry Mujeeb Ahmed et al. Process Skew: Fingerprinting the Process for Anomaly Detection in Industrial Control Systems. In *WiSec*, 2020.

[10] Bushra A. Alahmadi et al. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. In *USENIX Security*, 2022.

[11] Tejasvi Alladi et al. Industrial Control Systems: Cyberattack trends and countermeasures. *Computer Communications*, 155, 2020.

[12] Wissam Aoudi et al. Truth Will Out: Departure-Based Process-Level Detection of Stealthy Attacks on Control Systems. In *CCS*, 2018.

[13] Giovanni Apruzzese et al. SoK: Pragmatic Assessment of Machine Learning for Network Intrusion Detection. In *IEEE EuroS&P*, 2023.

[14] Andreas Bathelt et al. Revision of the Tennessee Eastman Process Model. *IFAC-PapersOnLine*, 48(8):309–314, 2015.

[15] Alvaro A. Cárdenas et al. Attacks Against Process Control Systems: Risk Assessment, Detection, and Response. In *ASIACCS*, 2011.

[16] John Henry Castellanos and Jianying Zhou. A Modular Hybrid Learning Approach for Black-Box Security Testing of CPS. In *ACNS*, 2019.

[17] Chi-Tsong Chen. *Linear System Theory and Design*. Saunders college publishing, 1984.

[18] Hongjun Choi et al. Detecting Attacks Against Robotic Vehicles: A Control Invariant Approach. In *CCS*, 2018.

[19] Mauro Conti et al. A Survey on Industrial Control System Testbeds and Datasets for Security Research. *IEEE Communications Surveys & Tutorials*, 23(4), 2021.

[20] Markus Dahlmanns et al. Missed Opportunities: Measuring the Untapped TLS Support in the Industrial Internet of Things. In *ASIACCS*, 2022.

[21] Ailin Deng and Bryan Hooi. Graph Neural Network-Based Anomaly Detection in Multivariate Time Series. In *AAAI*, 2021.

[22] Derui Ding et al. A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275, 2018.

[23] Joseph J. DiStefano et al. *Schaum's Outline of Feedback and Control Systems*. McGraw-Hill, 1976.

[24] Alessandro Erba and Nils Ole Tippenhauer. Assessing Model-free Anomaly Detection in Industrial Control Systems Against Generic Concealment Attacks. In *ACSAC*, 2022.

[25] Sandro Etalle. From Intrusion Detection to Software Design. In *ESORICS*, 2017.

[26] Cheng Feng et al. A Systematic Framework to Generate Invariants for Anomaly Detection in Industrial Control Systems. In *NDSS*, 2019.

[27] Clement Fung et al. Perspectives from a Comprehensive Evaluation of Reconstruction-based Anomaly Detection in Industrial Control Systems. In *ESORICS*, 2022.

[28] Clement Fung et al. Attributions for ML-based ICS Anomaly Detection: From Theory to Practice. In *NDSS*, 2024.

[29] Hamid Reza Ghaeini et al. State-Aware Anomaly Detection for Industrial Control Systems. In *SAC*, 2018.

[30] Jairo Giraldo et al. A Survey of Physics-Based Attack Detection in Cyber-Physical Systems. *ACM Computing Surveys*, 51(4), 2018.

[31] Jonathan Goh et al. A Dataset to Support Research in the Design of Secure Water Treatment Systems. In *CRITIS*, 2016.

[32] Zhongyuan Hau and Emil C. Lupu. Exploiting Correlations to Detect False Data Injections in Low-Density Wireless Sensor Networks. In *CPSS*, 2019.

[33] Won-Seok Hwang et al. "Do You Know Existing Accuracy Metrics Overrate Time-Series Anomaly Detections?". In *SAC*, 2022.

[34] J. Inoue et al. Anomaly Detection for a Water Treatment System Using Unsupervised Machine Learning. In *ICDMW*, 2017.

[35] Rolf Isermann and Marco Münchhof. *Identification of Dynamic Systems: An Introduction with Applications*, volume 85. Springer, 2011.

[36] Jonguk Kim et al. Anomaly Detection for Industrial Control Systems Using Sequence-to-Sequence Neural Networks. In *CyberICPS*, 2020.

[37] Eric D Knapp and Joel Thomas Langill. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2014.

[38] Moshe Kravchik and Asaf Shabtai. Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. In *CPS-SPC*, 2018.

[39] Moshe Kravchik and Asaf Shabtai. Efficient Cyber Attack Detection in Industrial Control Systems Using Lightweight Neural Networks and PCA. *IEEE TDSC*, 19(4), 2021.

[40] Marina Krotofil et al. The Process Matters: Ensuring Data Veracity in Cyber-Physical Systems. In *ASIACCS*, 2015.

[41] Dominik Kus et al. A False Sense of Security? Revisiting the State of Machine Learning-Based Industrial Intrusion Detection. In *CPSS*, 2022.

[42] Olav Lamberts et al. SoK: Evaluations in Industrial Intrusion Detection Research. *Journal of Systems Research*, 3(1), 2023.

[43] Maxime Lanvin et al. Towards Understanding Alerts raised by Unsupervised Network Intrusion Detection Systems. In *RAID*, 2023.

[44] Qin Lin et al. TABOR: A Graphical Model-based Approach for Anomaly Detection in Industrial Control Systems. In *ASIACCS*, 2018.

[45] Sean Lyngaas. Russia-linked hacking group suspected of carrying out cyberattack on Texas water facility, cybersecurity firm says. *CNN*, April. 17, 2024.

[46] Yilin Mo and Bruno Sinopoli. Secure control against replay attacks. In *47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, 2009.

[47] Neil Ortiz et al. From Power to Water: Dissecting SCADA Networks Across Different Critical Infrastructures. In *PAM*, 2024.

[48] Koyena Pal et al. Effectiveness of Association Rules Mining for Invariants Generation in Cyber-Physical Systems. In *HASE*, 2017.

[49] Venkata Reddy Palleti et al. A mechanistic fault detection and isolation approach using kalman filter to improve the security of cyber physical systems. *Journal of Process Control*, 68, 2018.

[50] Philipp Probst et al. Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *Journal of Machine Learning Research*, 20(53), 2019.

[51] Raul Quinonez et al. SAVIOR: Securing Autonomous Vehicles with Robust Physical Invariants. In *USENIX Security*, 2020.

[52] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4), 2018.

[53] Rahul Anand Sharma et al. Lumen: A Framework for Developing and Evaluating ML-Based IoT Network Anomaly Detection. In *CoNEXT*, 2022.

[54] Hyeok Shin et al. HAI 1.0: HIL-based Augmented ICS Security Dataset. In *CSET*, 2020.

[55] Riccardo Taormina et al. Battle of the Attack Detection Algorithms: Disclosing Cyber Attacks on Water Distribution Networks. *Journal of Water Resources Planning and Management*, 144(8), 2018.

[56] Rafael Uetz et al. You Cannot Escape Me: Detecting Evasions of SIEM Rules in Enterprise Networks. In *USENIX Security*, 2024.

[57] Muhammad Azmi Umer et al. Generating invariants using design and data-centric approaches for distributed attack detection. *IJCIP*, 28:100341, 2020.

[58] David Urbina et al. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *CCS*, 2016.

[59] Jingyi Wang et al. Towards 'verifying' a water treatment system. In *Formal Methods*, 2018.

[60] Hilde J.P. Weerts et al. Importance of Tuning Hyperparameters of Machine Learning Algorithms. *arXiv preprint arXiv:2007.07588*, 2020.

[61] Konrad Wolsing et al. Can Industrial Intrusion Detection Be SIMPLE? In *ESORICS*, 2022.

[62] Konrad Wolsing et al. Deployment Challenges of Industrial Intrusion Detection Systems. In *CyberICPS*, 2024.

[63] Konrad Wolsing et al. IPAL: Breaking up Silos of Protocol-dependent and Domain-specific Industrial Intrusion Detection Systems. In *RAID*, 2022.

[64] Zeyu Yang et al. PLC-Sleuth: Detecting and Localizing PLC Intrusions Using Control Invariants. In *RAID*, 2020.

A Appendix

A.1 Pseudocode of GeCo

The following pseudocode summarizes the implementation of GeCo whereas the full implementation is available at https://zenodo.org/records/15120036.

```
def train(x, y):
  for target in process_values:
    for func in function_templates:
      for length = 0 To max_function_length:
        forall subsets of process_values with length:
          # Fit function to first 80% of the data
          xt = x[:80%], yt = y[:80%]
          parameter = fit(xt, yt, func, target, subset)

          # Test if new fit is the current best fit
          errors = func(x, parameter) - y
          if MSE(error) > current best model for target:
            continue

          # Calculate drift and threshold
          drift = mean(errors) + stddev(errors)
          cusum = threshold = 0
          for error in errors:
            cusum = max(cusum + abs(error) - drift, 0)
            threshold = max(threshold, cusum)

def test(state):
  for function in functions:
    diff = cur_state - function(prev_state, parameter)

    # Update CUSUM and limit infinite growth
    cusum = max(cusum + abs(diff) - drift)
    cusum = min(cusum, threshold + drift * growth_factor)

    if cusum >= threshold * scale_factor
      raise alert
  prev_state = state
```

A.2 Pseudocode for Model Difference

To estimate when the training of GECO can be stopped, we calculate the relative change in the learned state-space model from one training step to another. The following pseudocode shows how we calculate this difference for two model:

```
def diff(a, b):
  diffs = []
  for target in process_values:
    if a and b use different function_templates: #1
      diffs.append(1)
    elif a and b's process variables differ: #2
      diffs.append(1)
    else: #3
      diffs.append(abs(
        (MSE(a) - MSE(b)) / max(MSE(a), MSE(b))
      ))

  return mean(diffs)
```

We calculate a score for each function learned by the model. If the models either use different function templates (cf. #1) or depend on different process values for the prediction (cf. #2), we attribute the highest difference (1). Otherwise (cf. #3) we calculate the relative difference between the mean squared error (MSE). The MSE is the difference between the training data and the model's prediction and indicates whether one fit is better than another. The difference can be between 0 and 1, with 1 denoting a maximal change between two models.

A.3 Hyperparameters and Analysis

In Tab. 6, we list the precise hyperparameters for each dataset used to conduct our evaluation. Beyond the results discussed in Sec. 7.1 regarding the influence of hyperparameters, we also conducted the same analysis for the other datasets as shown in Fig. 11. Again, the × indicates the hyperparameters we analyzed for this specific dataset. The black line indicates the areas within 5% of the maximum achievable performance.

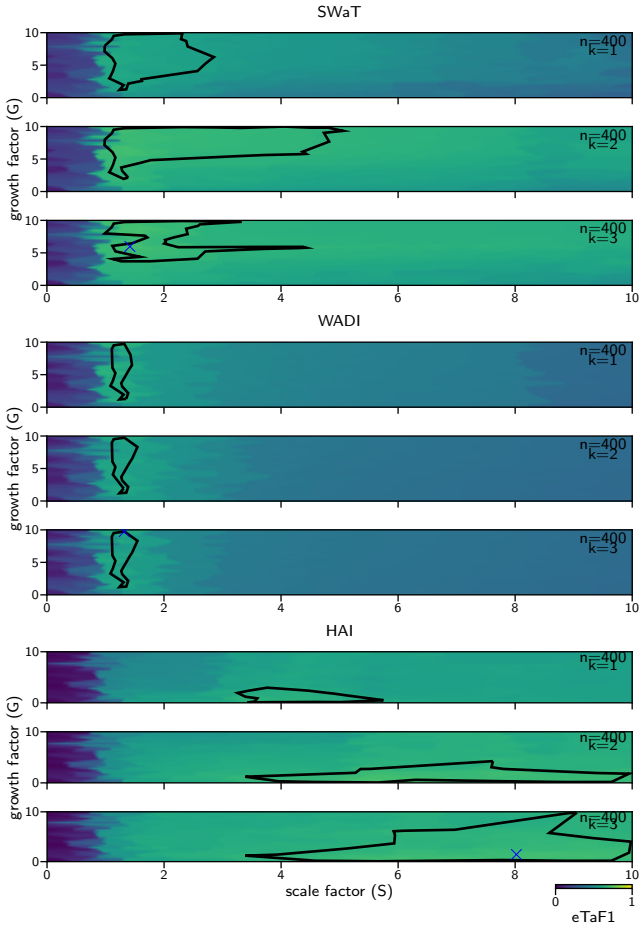


Figure 11: As discussed for BATADAL in Sec. 7.1, the hyperparameters of GECO on the other datasets are likewise stable, featuring large areas without sudden changes.

A.4 Tainted Training Data

The training data contained in the datasets may be superficially clean. In contrast, for real deployments, it is hard to guarantee clean data since, e.g., it could contain artifacts from faults or even unnoticed attacks. Moreover, due to changes in the network or variations of the process, the training might need to be repeated from time to time. Thus, we want to understand how GECO performs if the training data is tainted, i.e., some attack data is contained within the training data.

We adopt a similar methodology as proposed by Uetz et al. [56]. Instead of training GECO solely on the dedicated (benign) training set, we additionally taint the training data with some of the evaluation data containing attacks. We

Table 6: Hyperparameters selected for each dataset.

Dataset	Function Length k	Scale Factor S	Drift Factor G
SWaT	3	1.42	5.98
WADI	3	1.32	9.74
HAI	3	8.02	1.44
BATADAL	3	1.39	2.16
TEP	3	1.0	1.0

Dataset	Prec.	Rec.	F1	eTaP	eTaR	eTaF1	FPA	Scen.
SWaT	−0.00	−0.02	−0.01	−0.01	−0.04	−0.03	−10	+0.03
WADI	−0.00	−0.03	−0.04	−0.00	−0.01	−0.02	+0	+0.00
HAI	−0.00	−0.13	−0.09	+0.01	−0.10	−0.06	−18	−0.12
BATADAL	−0.00	−0.03	−0.02	−0.00	−0.01	−0.01	+0	+0.00

Table 7: GECO finds a good performing model even if the training data is tainted, i.e., contains two attacks in this case. The values measure the difference between a model trained without attacks and one with two attacks in training. A negative number indicates a performance loss due to the attacks in percent points. Note that a positive number for Scen. is better.

split each evaluation dataset after the second attack. The first split, containing two attacks, is added to the training while the remaining attacks are used for evaluation as before. Tab. 7 compares ˙GECO's detection performance being trained on the intended benign dataset part (baseline) and the tainted data. As we are left with two fewer attacks for the evaluation, we re-evaluated the baseline also on the same evaluation data missing those two attacks to keep the results comparable.

Overall, tainted training data has a comparable effect on all scenarios. The F1-scores decrease on average by 4.0 percent points, and eTaF1 decreases by 3.0 percent points. However, we cannot draw the expected conclusion that GECO's performance worsens on tainted data, as the number of FPAs decreases for SWaT and HAI. E.g., we observe 18 fewer FPA on HAI. Meanwhile, GECO only detects marginally fewer attacks in HAI while the other datasets see no difference or even experience a slight increasement (SWaT). We suspect that these results stem from potential overfitting in a too clean dataset, yet can not validate these speculations.

We conclude that the overall performance of GECO is not significantly worsened by tainted training data. Thus, we expect GECO to work decently outside the lab, as its training does not require to be perfectly clean of any anomalies.

Table 8: The 33 consolidated invariants from 7 sources [2, 3, 26, 48, 49, 57] that we combine into a knowledge-based IIDS.

ID	Invariants
1	LIT101<490 ⇒ P101 off ∨ P102 off
2	LIT301<790 ⇒ P101 on ∨ P102 on
3	LIT301>1010 ⇒ P101 off ∨ P102 off
4	LIT301<790 ⇒ MV201 open
5	LIT301>1010 ⇒ MV201 closed ∧ P101 off
6	FIT201<2 ⇒ P201 off ∧ P202 off ∧ P204 off ∧ P206 off
7	AIT201>260 ⇒ P201 off ∧ P202 off
8	AIT503>280 ⇒ P201 off ∨ P202 off
9	AIT202<695 ⇒ P203 off ∨ P204 off
10	AIT203>500 ⇒ P205 off ∨ P206 off
11	AIT402>330 ⇒ P205 off ∨ P206 off
12	LIT301<785 ⇒ P301 off ∨ P302 off
13	LIT401>900 ⇒ P301 off ∨ P302 off
14	LIT401<300 ⇒ P301 on ∨ P302 on
15	P401 on ∨ P402 on ⇒ FIT401>0.1
16	FIT401<1 ⇒ UV401 off
17	P401 off ⇒ UV401 on
18	UV401 off ⇒ P501 off
19	UV401 on ⇒ P501 on
20	FIT401<1 ⇒ P501 off
21	LIT101>810 ⇒ P601 on
22	LIT101<490 ⇒ MV101 open ∧ FIT101>0.1
23	LIT101>810 ⇒ MV101 closed ∧ FIT101≤1
24	P101 on ⇒ MV201 open
25	LIT301>1015 ⇒ MV201 closed ∧ P101 off
26	LIT301<785 ⇒ P301 on
27	FIT301>0.1 ⇒ P301 off
28	LIT401<290 ⇒ P301 on ∨ MV302 open
29	FIT401<1 ⇒ P301 off
30	MV101 open ⇒ FIT101>0.1
31	FIT401<1 ⇒ P301 off
32	LIT101<490 ⇒ FIT101>0
33	LIT301<790 ∧ LIT101>810 ∧ MV201 open ⇒ P101 on

Figure 12: In our visual analysis of GECO in Sec. 6.1, we depicted a partial view on the alerting behavior due to better visibility. This figure shows the entire alerting behavior of GECO and related work for all analyzed datasets. The areas marked with dotted lines correspond to the regions shown before in Fig. 5. An enlarged version of this figure is uploaded to Zenodo.

USENIX Security '25 Artifact Appendix: GeCos Replacing Experts: Generalizable and Comprehensible Industrial Intrusion Detection

Konrad Wolsing[*,‡] Eric Wagner[*,‡] Luisa Lux[*,‡] Klaus Wehrle[‡] Martin Henze[‡,*]

[*]*Fraunhofer FKIE* [‡]*RWTH Aachen University*

A Artifact Appendix

A.1 Abstract

Protecting Industrial Control Systems (ICSs) against cyber-attacks is crucial to counter escalating threats to critical infrastructure. To this end, Industrial Intrusion Detection Systems (IIDSs) provide an easily retrofittable approach to uncover attacks quickly and before they can cause significant damage. Current research focuses either on maximizing automation, usually through heavy use of machine learning, or on expert systems that rely on detailed knowledge of the monitored systems. While the former hinders the interpretability of alarms, the latter is impractical in real deployments due to excessive manual work for each individual deployment.

To bridge the gap between maximizing automation and leveraging expert knowledge, we introduce GeCo, a novel IIDS based on automatically derived comprehensible models of benign system behavior. GeCo leverages state-space models mined from historical process data to minimize manual effort for operators while maintaining high detection performance and generalizability across diverse industrial domains. Our evaluation against state-of-the-art IIDSs and datasets demonstrates GeCo's superior performance while remaining comprehensible and performing on par with rule-based IIDS based on the manual effort of experts.

This artifact provides the implementation of GeCo and describes how to reproduce our main results regarding the detection performance of GeCo, as well as our comparison to related work from Section 6.1 of the publication.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our artifact does not contain content that threatens security or privacy as we publish the code and the evaluation tools for a defensive intrusion detection system (GeCo). GeCo is evaluated purely on datasets of ICS process data and thus our evaluation process does not require malware. Due to privacy concerns, we do not make the results of our user study from Sec. 6.2.2 available.

A.2.2 How to access

Our artifact is available at https://zenodo.org/records/15120036. Besides the files serving as an online appendix for our publication, *code-and-results.zip* contains the files for this artifact. It includes the code of the proposed IIDS (GeCo) and the configuration files used for its evaluation. A permanently maintained version of GeCo is available at the IPAL repository https://github.com/fkie-cad/ipal_ids_framework/tree/master/ids/geco, yet it is not required for reproducing the results from our publication.

A.2.3 Hardware dependencies

GeCo does not require specific hardware architectures. Still, the hardware requirements (CPU cores and memory) depend on the dataset and impact the runtime performance. To use and evaluate GeCo with pre-trained models, a single CPU core with a minimum of about 2Gb RAM has proven sufficient. However, to retrain the models of GeCo, more memory must and more CPU cores should be spared.

The smaller datasets (BATADAL and TEP) can be trained in under one hour with 8 CPU cores and at least 2Gb of memory. The training of the SWaT, WADI, and HAI models is much more demanding (cf. Section 6.3 and Table 5 in our paper). Regarding memory, they require at least 16GB for SWaT and HAI and 64GB for WADI to load the datasets. The number of CPU cores impacts the training time. Note that depending on the chosen number of CPU cores, additional memory is required to enable parallel processing.

A.2.4 Software dependencies

GeCo and the evaluation tools are a collection of Python scripts that have been tested with Python 3.12. Other versions are expected to work as well. The implementation has been tested on Linux and macOS operating systems.

The implementation of GeCo is based on the open-source IPAL IDS framework[1] and uses its provided datasets and evaluation tools. Except for the datasets (cf. the following A.2.5), all necessary software is provided within the artifact.

[1] IPAL – https://github.com/fkie-cad/ipal

List of used software versions:

- https://github.com/fkie-cad/ipal_ids_framework v1.5.2
- https://github.com/fkie-cad/ipal_evaluate v1.2.9
- https://github.com/fkie-cad/ipal_datasets v1.3.7

A.2.5 Benchmarks

Note that we cannot publicly provide the dataset files (*datasets/*) for legal reasons. Still, the datasets can be obtained by the publishers and transcribed into the IPAL format following the instructions here (cf. *datasets/README.md*). These steps are necessary to reproduce any of our results:

1. Clone the https://github.com/fkie-cad/ipal_datasets repository.
2. Install the dependencies listed in the *requirements.txt* file.
3. Select a dataset, e.g., SWaT, for this example.
4. Obtain the raw dataset from the publishers. Links to the dataset publishers are listed in the *ipal_datasets/README.md* table. The precise version and files used in our paper are listed below.
5. Place the dataset files in the respective folder. For example, *ipal_datasets/SWaT/raw/[dataset files]*.
6. Execute the *transcribe.py/sh* script for the chosen dataset.
7. Copy the generated files and replace this artifact's placeholders *attack.json* and *ipal/** files.

List of used files and dataset versions:

- SWaT: Download all three XLSX files from the *SWaT.A1 & A2_Dec 2015/Physical* folder.
- WADI: Download all CSV files from the *WADI.A2_19 Nov 2019* folder.
- HAI: Download all **csv.gz* from the https://github.com/icsdataset/hai/tree/master/hai-21.03 repository.
- BATADAL: Download all three CSV files from https://www.batadal.net/data.html (*BATADAL_dataset04.csv*, *BATADAL_dataset03.csv*, and *BATADAL_test_dataset.csv*).
- TEP: Download all files from the https://github.com/mikeliturbe/pasad/tree/master/data repository.

A.2.6 Description of files and content

The artifact consists of various files and folders, some relevant for reproducing our results while others serve as additional material for our paper. In the following, we describe the structure of the artifact:

- *code/*: This folder contains the software necessary for conducting our evaluation, based on the IPAL IDS Framework (cf. A.2.4), and the implementation of GeCo (*code/ipal-ids-framework/ids/GeCo*).
- *config/*: For each dataset, the **.json* files define the hyperparameters for GeCo. Based on these configurations and the training data, GeCo learns a model stored within the respective **.model* files.
- *datasets/*: We evaluated GeCo on five datasets stored in this folder (cf. A.2.5 on how to prepare the datasets).
- *knowledge-based/*: This folder contains the artifact to reproduce our results of Sec. 6.1.3.
- *output/*: Once the datasets are labeled by GeCo, we store the IDSs output here.
- *related-work/*: This folder contains the alerts of related work (SIMPLE, TABOR, Invariant, Seq2SeqNN, PASAD), which we use as a comparison in Table 2. These results were reproduced with the help of the IPAL IDS framework. The **.state.gz* files contain the labeled datasets for each IDS, and the **.json* files the respective metrics.
- *results*: Based on the labeled dataset from *output/*, the **.json* files contain the calculated performance metrics for GeCo and the **.pdf* files depict the alerts of GeCo (black) compared to the dataset's attacks (red).
- *videos/*: This folder contains renderings of the dependency graph from Figure 7a for all datasets, which are complementary material for our publication.

A.3 Set-up

A.3.1 Preparation

Please prepare the dataset files as described in A.2.5. For each dataset, all the dummy files in the respective dataset folders (*datasets/SWaT,WADI,HAI,BATADAL/**) have to be replaced.

A.3.2 Installation

Execute the following commands to install the required software for GeCo:

```
python3 -m venv venv
source ./venv/bin/activate
pip3 install igraph==0.10.4
pip3 install code/ipal-ids-framework/
pip3 install code/ipal-evaluate/
```

More information can be found in the *README.md* file.

A.3.3 Basic Test

Execute the following command to apply GeCo to one of the selected datasets.

```
./run-ids.py {SWaT,WADI,HAI,BATADAL,TEP}
```

To test whether the installation was successful, select "n" in the *run-ids.py* script to skip the sometimes long training process. For testing the training, use one of the datasets for which training is fast, such as BATADAL or TEP.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): GeCo shows competitive detection performance, which is often better than related work. This is proven by the evaluation (E1) described in Section 6.1.1, whose results are reported in Table 2, Table 6, and Figure 12.

(C2): GeCo generalizes well to a new domain of the chemical Tennessee Eastman Process (TEP). This claim is supported by the evaluation (E2) described in Section 6.1.2, whose results are reported in Figure 6.

(C3): GeCo yields a detection performance that is on par with labor-intensively created knowledge-based IIDSs. This claim is assessed in evaluation (E3) described in Section 6.1.3 and Table 3.

(C4): GeCo's alerts are comprehensible as they closely correlate to the ICS's underlying physical process. This experiment was designed and conducted in Section 6.2.1 and the results are shown in Figure 7 and Figure 8.

A.4.2 Experiments

How to, preparation, and execution

To conduct the evaluation, execute the *run-ids.py [dataset]* script, providing one of the five datasets (SWaT, WADI, HAI, BATADAL, and TEP) as the argument one by one. If the goal is to reproduce our results, select "n" at the beginning of the *run-ids.sh* script. That way, GeCo uses the provided pretrained models from the *config/* folder. These steps suffice to reproduce the results of our experiments (E1–E4). [5 human-minutes, 10 compute-minutes, 1GB disk space]

Optionally, the detection models of GeCo can also be retrained on the training data instead of using the pretrained models. To this end, select "y" at the beginning of the *run-ids.sh* script. Since the training phase can be extensive depending on the dataset, it is advisable to leverage an appropriate amount of CPU cores for parallelization (cf. Section 6.3). The number of cores for training can be configured in the configuration files under *config/{SWaT,WADI,HAI,BATADAL,TEP}.json* with the *cpus* option (default is 7). We recommend training the BATADAL or TEP model first, as they are the least computationally demanding datasets [5 human-minutes, 1-2 compute-hours, 1GB disk space]. SWaT, HAI [5 human-minutes, 24 compute-hours] and WADI [5 human-minutes, 1+ compute-month] demand significantly more time for training even with 64 CPU cores and 236GB of RAM. Yet, this step is not mandatory for the success of the following experiments.

Results

(E1): *Detection Performance* [10 human-minutes]: GeCo's detection performance is measured by comparing its alerts to the datasets labels. The results can be found in the *results/* folder for each dataset, where the **.json* files list different detection performance metrics and the **.pdf* files visualize the alerts in relation to the attacks. To visualize and aggregate these results, execute the respective scripts (*table-2.py*, *fig-12.py*, and *table-6.py*) to obtain the results shown in our publication. These scripts either render an image or generate the LaTeX code for the respective figures and tables.

(E2): *Generalizability* [5 human-minutes]: We show that GeCo successfully transfers to a new industrial domain by applying GeCo to the TEP dataset. The detection performance stated in Section 6.1.2 (96.0 in F1 and 98.0 in eTaF1) can be validated with the *results/TEP.json* file. Additionally, the *fig-6.py* script renders GeCo's alerts compared to the PASAD IIDS.

(E3): *Expert Knowledge-based IIDS* [5 human-minutes]: We implemented an IIDS that leverages a collection of Invariants as listed in Table 8. Execute the following script located under *knowledge-based/run.sh* to apply this IIDS to the SWaT dataset. The results are stored in the *output.json* and *output.pdf* files. To reproduce Table 3 of our paper, execute the *table-3.py* script.

(E4): *Alerts Localization and Understandability* [5 human-minutes]: We analyze GeCo's detection model to infer which process values are responsible for the alerts by analyzing the dependency between the learned state-space models by means of a dependency graph in Figure 7a. The dependency graph can be recreated with the respective *fig-7a.py* script. Note that the graph is arranged differently with each execution of the script. Lastly, the *fig-8.py* script generates the results for Figure 8.

A.5 Notes on Reusability

If the goal is to examine GeCo's detection performance in more detail on the datasets evaluated so far, please refer to the files in the *output/* directory, as they contain the raw output of GeCo for each dataset. For other activities, such as applying GeCo to different datasets, we recommend working with the implementation provided by the IPAL IDS framework[1]. To this end, we direct the reader to the tutorial and documentation about IPAL[2]. For guidance on how to configure the hyperparameters of GeCo, please refer to Section 7.1 of our publication.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2025/.

[2] IPAL Tutorial – https://github.com/fkie-cad/IPAL/blob/main/tutorial/Tutorial.md