Developing a Strong CPS Defender: An Evolutionary Approach

Qingyuan Hu[†], Christopher M. Poskitt[‡], Jun Sun[‡], Yuqi Chen[*†]
[†]ShanghaiTech University, China, {huqy2022, chenyq}@shanghaitech.edu.cn;
[‡]Singapore Management University, Singapore, {cposkitt, junsun}@smu.edu.sg

***Abstract*—Cyber-physical systems (CPSs) are used extensively in critical infrastructure, underscoring the need for anomaly detection systems that are able to catch even the most motivated attackers. Traditional anomaly detection techniques typically do 'one-off' training on datasets crafted by experts or generated by fuzzers, potentially limiting their ability to generalize to unseen and more subtle attack strategies. Stopping at this point misses a key opportunity: a defender can actively challenge the attacker to find more nuanced attacks, which in turn can lead to more effective detection capabilities. Building on this concept, we propose *Evo-Defender*, an evolutionary framework that iteratively strengthens CPS defenses through a dynamic attacker-defender interaction. *Evo-Defender* includes a smart attacker that employs guided fuzzing to explore diverse, non-redundant attack strategies, while the self-evolving defender uses incremental learning to adapt to new attack patterns. We implement *Evo-Defender* on two realistic CPS testbeds: the Tennessee Eastman process and a Robotic Arm Assembly Workstation, injecting over 600 attack scenarios. In end-to-end attack detection experiments, *Evo-Defender* achieves up to 2.7× higher performance than state-of-the-art baselines on unseen scenarios, while utilizing training data more efficiently for faster and more robust detection.**

***Index Terms*—Cyber-physical systems; benchmark generation; incremental learning; defense strategies**

I. Introduction

Cyber-physical systems (CPSs), which tightly integrate cyber and physical components, are widely deployed in critical domains such as water treatment, electricity distribution, and pharmaceutical manufacturing. High-profile incidents—including the Stuxnet-induced centrifuge failure at Natanz [1] and the Industroyer-triggered grid collapse in Kyiv [2]—underscore how CPS vulnerabilities can translate into physical-world catastrophes. The inherent complexity and legacy nature of many CPSs creates security blind spots that traditional IT defenses struggle to address, making it challenging to build robust protection mechanisms. Consequently, anomaly detection has become a widely relied-upon strategy, particularly for legacy systems. While mathematically modeling CPS behavior for anomaly detection is often challenging and time-consuming, the abundance of sensor and actuator data has led to the widespread adoption of machine learning algorithms to develop data-driven, time-series anomaly detectors [3]–[9].

For data-driven approaches, detection performance heavily depends on the quality and diversity of the training dataset. Unfortunately, these datasets typically only capture benign operating conditions, limiting their utility for detecting real-world attacks. Collecting data from scenarios involving active attacks or abnormal behavior—such as those in [10], [11]—requires significant time, effort, and domain expertise. Even then, such datasets can carry inherent biases. Furthermore, generalizing these methods across diverse CPSs—particularly legacy systems—remains highly impractical due to each system's unique processes, mechanisms, and complexities. This creates a fundamental paradox: defenders not only need vast amounts of high-quality data to achieve robust detection, but also require substantial computational and storage resources to process and retrain on such data. However, in practice, they often lack both, making effective and adaptive defense especially challenging.

To address this challenge, prior work has applied automated testing and fuzzing techniques to systematically explore the behavior of CPSs [12], [13]. These methods can efficiently generate test cases that drive the system into hazardous states, making them useful for evaluating existing anomaly detection mechanisms. However, they are less effective for producing the clean, diverse data required to train robust anomaly detectors. These techniques typically rely on a machine learning model trained on sensor and actuator logs or network traffic to guide the automated discovery of diverse attacks targeting different sensors or actuators. Their key limitation lies in their singular focus on achieving attack goals, without consideration for co-developing defensive strategies. Consequently, while these approaches may partially address data scarcity, they often neglect data quality and generate redundant or noisy data. This undermines their usefulness for building robust defenders, which require significant computational resources.

To address these challenges, we propose a novel framework, *Evo-Defender*, designed to automatically construct a robust CPS attack detection system, particularly for scenarios where the defender has access to limited data due to constrained computing power and memory. Our framework consists of two main components: the Spear and the Shield, as illustrated in Figure 1. In the first component, the objective is to extract data from the CPS through a diverse range of attacks, capturing various abnormal states the system may encounter. To mitigate the challenges posed by the massive search space and data homogenization, we employ a machine learning model to guide the Spear, ensuring it avoids redundant attack strategies while achieving multiple attack objectives. By exposing the CPS to unsafe states, this process reveals critical attack behaviors,

* Corresponding author.

thereby strengthening the Shield. In the second component, we implement an incremental learning approach tailored to the time-series data generated by the Spear. This method addresses both the scarcity of abnormal CPS state data and the resource constraints by supporting dynamic model updates without retraining from scratch or storing vast historical logs. This enables efficient knowledge accumulation, rapid adaptation, and reduces issues with redundant or low-quality data scenarios typical of CPS environments. First, we filter out misclassified data from the latest CPS logs using the current defender, which is typically a machine learning-based anomaly detector initialized randomly. Then, a multi-module incremental learning strategy—incorporating exemplars, imbalanced learning, and continual backpropagation—leverages the misclassified data to enhance the Shield's performance. This approach minimizes the need for extensive data storage and retraining while rapidly integrating newly discovered threat patterns into the defense, eliminating the need to wait for a complete attack dataset before improving detection capabilities.

We evaluated the effectiveness of our solution using two realistic CPS testbeds: the Tennessee Eastman process (TE) and the Robotic Arm Assembly Workstation (RAAW). TE is a widely used chemical process simulation environment developed by Matlab Simulink [14], whereas RAAW is a real-world robotic arm platform. We deployed *Evo-Defender* on these platforms, injected 336 and 300 attacks respectively, and recorded the entire process in detail for the training and evaluation. For TE, the injected attacks targeted high reactor pressure, high temperatures, and abnormal tank liquid levels, whereas for RAAW, attacks attempted to halt CPS operations and cause a sudden pause in movement. With incremental training, *Evo-Defender* achieves 92.03% and 89.59% accuracy on two platform test sets, improving upon the best baselines by 23.38% and 29.07%, respectively. In end-to-end tests, the model correctly identifies 89 of 103 and 38 of 44 attacks on their respective test sets. Compared to the best baseline, the TE platform's false positive rate drops to one-third, while the RAAW platform's detection rate is nearly tripled. Finally, we conducted an ablation study on *Evo-Defender* to assess each module's contribution across various incremental learning module configurations. The results indicated that our method requires fewer evolution rounds than other learning schemes and utilizes data more efficiently, leading to faster improvements in defender's performance.

In summary, our work contributes the following:

- **Automated Attack Generation**: We develop a smart attacker that efficiently uncovers varied attack trajectories, generating a comprehensive dataset of over 600 attack scenarios on two real CPS platforms.
- **Continual Learning Defender**: We propose the first online scheme for incrementally updating the defense model, with the goal of improving existing deep learning-based anomaly detection methods when faced with incorrect incoming prediction data. In end-to-end testing scenario, our defender outperformed the baseline by at least 2.7 times in accuracy. In terms of data consumption, our approach used 69.6% less data and delivered higher accuracy than the baseline.
- **Robust Detection Capability**: We implement our approach on two CPS platforms, demonstrating its ability to detect a wide range of unsafe states and validating the effectiveness of continual evolution in addressing newly emerging threats.
- **Data Availability**: We provide the results at [15].

The remainder of this paper is organized as follows: Section II introduces the background of our two CPS testbeds and problem definition. Section III details the architecture of our *Evo-Defender* framework and a case study demonstrating the deployment of our approach on a realistic CPS platform. Section IV provides experimental evaluation across accuracy and robustness metrics. Section V reviews related work in adversarial machine learning for CPS. The paper concludes with final remarks in Section VI.

II. Background and Problem Definition

In this section, we provide an overview of the two CPS platforms—TE and RAAW—which serve as the testbeds for evaluating our approach. We then present a problem definition followed by a summary of the threat model considered in this work.

TE Testbed*:* The Tennessee Eastman (TE) process is a computational model that simulates a chemical plant with five main units: a reactor, condenser, compressor, separator, and stripper. It is widely used for designing and testing control algorithms, with a well-documented architecture and implementation [14], [16]. As shown in Figure 2, the process involves reactants (A, C, D, E) converted into two products (G, H), along with an inert and a byproduct, after passing through the main units. Fung's work [17] provides modified TE code for sensor and actuator manipulations implemented in Matlab Simulink, making it an ideal testbed for our modification and testing. We built upon their work to enable more flexible manipulations, including 25 sensors, 9 actuators, and 17 PID controller configurations.

The TE design includes defining safe operational states through specific constraints that are vital for equipment protection and safety. Any violation of these constraints triggers an automatic system shutdown. Downs and Fogel [16] outline specific operational constraints for each process unit. For example, if the Reactor Pressure indicator detects a pressure exceeding 3000 kPa, the system will execute its interlock strategy to shut down the process.

RAAW Testbed*:* RAAW is a fully automated platform featuring an ABB robotic arm and a Siemens S7-1200 PLC, creating a two-node distributed CPS capable of executing a wide range of assembly tasks. The PLC uses 29 digital inputs and 22 digital outputs to communicate with sensors, actuators, and the robotic arm, facilitating the exchange of states and signals. It interacts with components like warehouse sensors, conveyor belt sensors, gripper position sensors, and safety door sensors. In this study, we assign it a palletizing task: stacking six plastic rectangular prisms into a pyramid-shaped structure.

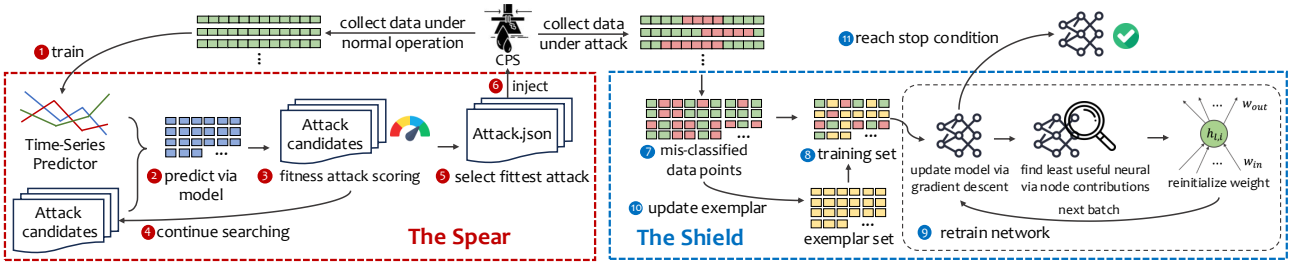


Fig. 1: Overview of our approach

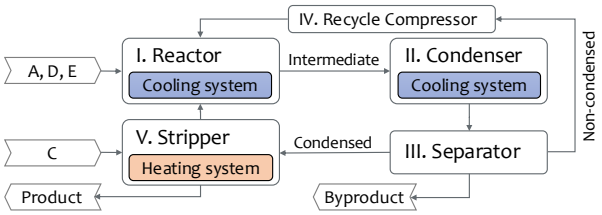


Fig. 2: The Tennessee Eastman Challenge Process. A, C, D, and E are the are the four raw materials participating in the reaction

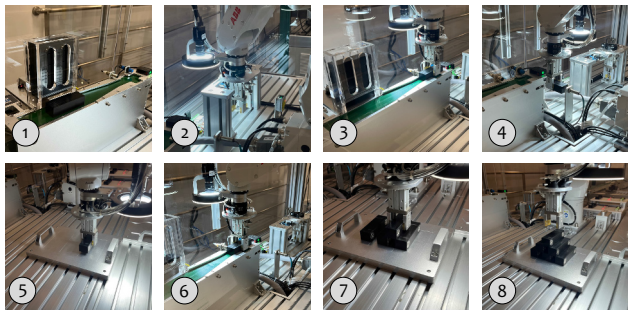


Fig. 3: The workflow of a robotic arm in palletizing tasks

As shown in Figure 3.1, the process begins with a plastic material being pushed from the warehouse to specific locations via a conveyor belt. Simultaneously, the robotic arm retrieves its gripper (3.2). The conveyor belt transports the pallet to the designated location, where the robotic arm grabs the material (3.3). The robotic arm uses sensors to ensure that it has successfully captured the material (3.4). After grasping the material, the robotic arm places it on the platform (3.5). This action is repeated as the robotic arm retrieves additional materials (3.6-7). Finally, it positions the last material to complete the pyramid formation (3.8). Through six rounds of PLC-triggered grasping and placement, six plastic materials are built into a pyramid.

The PLC stores 127 variables in fixed memory locations, similar to programming variables, to facilitate palletizing and management of the alarm system. Similar to TE, RAAW has predefined safety limits. For example, if a safety door is open, an alarm triggers, setting a Boolean variable to true, which activates the alarm procedure.

Problem Definition: A CPS is a decentralized reactive system, interacting with its physical environment via inputs and outputs in an ongoing manner. Consider the TE testbed as an example. Figure 4 illustrates a cascading control structure

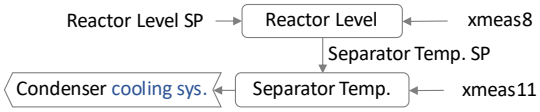


Fig. 4: Example of a cascading controller in TE

designed to maintain the reactor's liquid level. Rather than controlling the reactor level directly, the system achieves regulation by manipulating downstream components. Specifically, a master control loop continuously monitors the liquid level via sensor *xmeas8* and generates a Separator Temperature Setpoint by combining the measured value with a predefined Reactor Level Setpoint. A slave control loop then controls the separator temperature by processing the setpoint from the master loop and the temperature signal from sensor *xmeas11*, thereby managing the condenser's cooling system. The TE system comprises numerous such interdependent controllers, all working toward a common operational objective. This complexity introduces significant system analysis challenges, especially when attempting to reason about behavior under adversarial conditions.

Given the tightly coupled and domain-specific nature of CPSs, both attacking and defending them demand not only cybersecurity expertise but also a deep understanding of the underlying physical processes. To reduce reliance on such domain knowledge and to lower the associated costs across diverse CPS platforms, we draw inspiration from prior work on fuzz testing [18] to uncover attacks that induce abnormal system states by generating malicious network packets or control commands. However, due to the inherent nature of fuzzing, the generated data often exhibit low quality, class imbalance, and high redundancy. Moreover, real-world CPS deployments face challenges such as sensor drift [19], [20] in physical processes, which cause ongoing shifts in data distribution. These factors result in dynamically evolving data characteristics, which limit the effectiveness of static, batch-trained anomaly detectors. Therefore, incremental learning is essential in our problem setting, as it enables continual model adaptation to both the heterogeneous fuzzing-generated samples and real-world changes, ultimately improving the robustness and coverage of the anomaly detector.

Threat Model: As the objective of the Spear is to generate data that can be used to enhance and evaluate *Evo-Defender*, we first clarify the assumptions made regarding the CPS and the nature of potential attacks. In our approach, the Spear is designed to emulate real-world adversaries who target control

components and the communication channels between sensors and actuators, thereby enabling network-based attacks.

We assume that adversaries possess certain privileges within the CPS, enabling them to access and modify actuator commands, sensor readings, and system configurations. Additionally, attackers are assumed to have access to ground-truth sensor data, allowing them to observe the effects of their manipulations. Our threat model accommodates a spectrum of adversaries, ranging from highly capable attackers—who can simultaneously compromise multiple components—to more constrained ones with limited access (e.g., only a subset of sensors). While this model may appear to empower the attacker, our objective is not to simulate a likely real-world compromise, but rather to construct a diverse and systematic framework for evaluating the resilience of defensive mechanisms. By enabling the simulation of a wide range of adversarial behaviors, this flexible framework facilitates comprehensive testing and contributes to the development of more robust defensive strategies within *Evo-Defender*.

III. Approach and Implementation

The overall objective of our solution is to quickly develop a robust defender without depending on existing datasets or requiring extensive computing resources. To achieve this goal, we propose a practical, generalizable framework that leverages the Spear to continuously reinforce the Shield. Our approach enables the creation of a powerful defender with minimal data while maintaining high accuracy in identifying real-world attacks.

Our approach mainly consists of two components, the Spear and the Shield, as illustrated in Figure 1. In the Spear, we train a predictive model using data from a CPS historian, which includes sensor readings, actuator values, and configurations. This guides the Spear in identifying and injecting diverse effective attacks into the CPS to observe its operational state under attack. Subsequently, we apply incremental learning techniques using these operational states to enhance the defender's capabilities, allowing it to learn the latest attack characteristics and update its defense strategies. Finally, through iterative execution of these steps, we develop a robust defender and validate its performance with real testing datasets.

In the following, we outline the general implementation of these broad steps and illustrate their application using the TE testbed as an example, with the same approach also applicable to the RAAW platform. The corresponding code is provided in the supplementary materials [15].

A. Pre-Requisites

The Spear stage requires a model capable of predicting the effects of manipulating sensor and actuator inputs. This is a standard prerequisite for CPS-based testing (e.g., [12], [13]), and we elaborate on the steps below.

Data Collection*:* Training an effective prediction model requires a dataset from the target CPS that captures the relationships between actuator values, sensor readings, controller configurations, and future system states. This data should be recorded at regular intervals and encompass a range of operational scenarios to ensure comprehensive coverage. The dataset can be obtained from the system historian or collected passively by monitoring SCADA [21] communications or network traffic, provided the CPS operates without external interference. To further enrich the data and expose additional system behaviors, we may also introduce controlled variations in configurations or actuator settings during runtime.

Implementation for TE. We utilized the TE process in MATLAB Simulink to achieve this goal, incorporating both data recording and real-time configuration adjustments. Each operational cycle was simulated over 12 hours, during which we recorded data points every 0.5 seconds. To increase data variability, we introduced several manually crafted attacks after 5 hours of runtime, such as altering a PID controller configuration to a random value, and continued system operation until a timeout or shutdown occurred. After each run, we reset the system for the next data collection round. In total, we recorded 30 data entries, each containing between 10,000 and 24,000 sampling points, including sensor readings, actuator values, and configuration parameters. The dataset was randomly partitioned, with 80% allocated for training and the remaining 20% reserved for model validation.

Training a Prediction Model*:* In this step, we use the collected raw data to train a model that guides the Spear to identify effective manipulations. Before training, we convert the well-organized dataset into a series of vectors with a fixed format suitable for processing by a learning algorithm, such as a neural network. In this work, we assume that the status of the CPS is described by time series data at discrete time points, represented by three kinds of variables: sensor readings, actuator values, and configurations. These values can be either discrete or continuous. Based on this assumption, a possible form is $\langle \mathbf{A}_{0:n}, \mathbf{S}_{0:n}, \mathbf{C}_{0:n} \rangle$, where the $\mathbf{A}_t$ $\mathbf{S}_t$ and $\mathbf{C}_t$ signify the sets of values for actuators, sensors, and system variables, respectively, at time point t. Given that this vector includes both continuous and discrete values, an appropriate data normalization method must be applied after data collection. Once the feature vectors are defined, we proceed to train a supervised machine learning model to predict future features. It is important to note that while the model should ideally be accurate with respect to observed data, there is flexibility in the required level of accuracy. Since the attacker seeks to identify new attacks for which no prior data exists, the model does not need to be perfectly accurate for all unseen scenarios. Instead, the model is expected to capture the correct trends, enabling it to guide the identification of effective manipulations. Furthermore, models can be refined over time to address these blind spots, as demonstrated in related works such as [13].

Implementation for TE. We assume that the Spear manipulates CPS through the network, and the future state of TE depends on its previous operational state. Therefore, the input format is $\langle f_0, f_1, ..., f_t \rangle$, where f_t represents the feature at time point t, $f_t = \langle s_{0,t}, ..., s_{n,t}, a_{0,t}, ..., a_{m,t}, c_{0,t}, ..., c_{l,t} \rangle$ for each time point t. As our evaluation of attack success is based on violations of the TE system's operational constraints, we focus

Algorithm 1: Attack Vector Generation with Optional Genetic Algorithm

input : Vector of current CPS status v_0, prediction model M_p, max feature number n_m, population size p, fitness function f
parameter: Boolean flag $useGA$ to enable genetic algorithm, number of generations g (if $useGA$ is true)
output : Attack vector v_a

1 Let $V_s := \langle\rangle$; `// Initialize sequence`
2 **while** *Size of* $V_s < p$ **do**
3 Construct an attack vector v from v_0 by randomly selecting and manipulating $n \leq n_m$ features;
4 $s_p := M_p(v)$;
5 $V_s := V_s \cup \langle v_p, f(s_p)\rangle$;
`// Apply genetic algorithm if required`
6 **if** $useGA$ **then**
7 **for** $gen \leftarrow 1$ **to** g **do**
8 $V_s :=$ $V_s \cup$ CrossoverAndMutate(V_s) ;
9 $V_s :=$ Select a subset from V_s using *Roulette Wheel Selection* based on v's fitness value $f(M_p(v))$, where $v \in V_s$;
10 Select an attack vector v_a from *V* using *Roulette Wheel Selection* with corresponding fitness values $f(s_p)$;
11 **return** v_a;

Algorithm 2: Coverage-guided Fitness Function

input : Candidate attack embeddings $E = \langle e_1, e_2, ..., e_n\rangle$, past attack embeddings $F = \langle f_1, f_2, ..., f_m\rangle$, feature extractor φ from predictor
output: Rating for candidate attacks D'

1 $D \leftarrow \emptyset$
2 **for** *each attack* e_i *in* E **do**
3 $d_i = \frac{1}{m}\sum_{j=1\sim m}(\texttt{l2-norm}(e_i, f_j))$
4 $D = D \cup d_i$
5 d_{max}, d_{min} = $\texttt{max}(D)$, $\texttt{min}(D)$
6 normalize ds in D with d_{max} and d_{min} using **Min-Max Scaling** to get D'

our prediction on a subset of critical sensor readings directly tied to these constraints—specifically, reactor pressure, reactor level, reactor temperature, separator level, and stripper level.

For model selection, we employ a Long Short-Term Memory (LSTM) network due to its effectiveness in handling sequential and multivariate data, as well as its capability to capture complex nonlinear dependencies—characteristics essential for modeling CPS dynamics. Our implementation utilizes a straightforward LSTM architecture followed by a linear output layer.

B. The Spear

In the Spear, our objective is to systematically find attack vectors that the existing anomaly detection model cannot catch. To achieve it, building on Chen et al.'s work [12], [13], we use random search and genetic algorithms to generate meaningful attack vectors, as illustrated in Algorithm 1. As outlined in the threat model, we assume that the Spear can arbitrarily manipulate components within the CPS through network-based attacks (e.g., man-in-the-middle attacks). We represent a sequence of such attacks using a floating-point vector, where each element corresponds to a modification applied to a specific variable within the CPS under test.

Unlike previous work that focuses solely on finding more attacks, we place greater emphasis on attack diversity to maximize the coverage of the attack surface. Existing CPS testing approaches typically pursue a single objective, limiting exploration and making it difficult to uncover more system vulnerabilities. Our design encourages the Spear to generate diverse attacks, using a fitness function to evaluate the similarity between new and historical attacks. For example, if most existing attacks have caused an increase in the reactor tank's liquid level, subsequent attacks should aim to trigger other system behaviors in the CPS, with higher scores assigned to more distinct effects. In addition, we focus on generating attacks that significantly deviate the system from its normal operational state; the fitness function also considers actual sensor readings, assigns higher scores to attacks causing greater deviations from the normal sensor value ranges.

By integrating these two design considerations—behavioral diversity and deviation from normalcy—the fitness function effectively steers the search process toward the generation of diverse and impactful attack vectors. Once formally defined, the fitness function assigns a fitness value to each attack vector, reflecting its threat level (Algo 1, Line 5). To refine these attack vectors, we can apply heuristic search strategies such as random search (Algo 1, Lines 2–5) or genetic algorithms (Algo 1, Lines 6–9) to explore the space of sensor values, actuator states, and system configurations. After generating a set of attack candidates, we use Roulette Wheel Selection [22] to choose the attack. Each candidate's selection probability is proportional to its fitness, calculated as $f_i / \sum_{j=1}^{n} f_j$, where f_i is the fitness of candidate i. We generate a random number between 0 and the total fitness sum, then iterate through the candidates, accumulating their fitness values until the sum exceeds the random number. The candidate at this point is selected as the attack vector. This approach ensures both the diversity and effectiveness of the generated attack vectors in evaluating the security and robustness of the CPS.

Once the attack vectors are generated, we inject them into the CPS to evaluate the performance of the defense mechanism, specifically targeting manipulations that can drive certain system properties into unsafe states without triggering alarms from the anomaly detector. Whether or not data is logged depends on the response of the defender. If the defender fails to detect the injected attack, or erroneously raises an alarm when the CPS is operating normally (i.e., a false positive), we record the corresponding data along with its generated label for that period. If the system continues to operate normally until a predefined time limit is reached, we consider the attack unsuccessful, and the resulting data is labeled as normal. After each injection, the system is reset to normal operation, and this process repeats until the defender meets a predefined stopping condition (e.g., when the defender consistently detects attacks exceeding a specified threshold multiple times).

Implementation for TE. In TE, all sensors, actuators, and configuration parameters are represented as floating-point

values, allowing attacks—defined as sequences of control commands—to be encoded as float vectors. Each element in an attack vector corresponds to a command that alters a specific configuration. For example, an attack vector $\langle -2.0, 0.5, \ldots, 0.0 \rangle$ represents a series of commands aimed at causing a reactor overflow. The first element changes the reactor temperature controller's proportional gain (Kc) from -8 to 8, the second updates the integral gain (Ti) parameter from 0.125 to 0.1875, and so on, with 0.0 indicating no change to a configuration variable. In this case, the attack changed the proportional gain from -8 to +8, causing the controller to issue heating commands rather than cooling commands as the temperature rose. This led to a rapid temperature increase, intense reactor vaporization, and a pressure surge that triggered the high-pressure alarm. For manipulating sensors and actuators, we follow the method described by Fung et al. [17] to apply diverse attack approaches.

The fitness function in TE consists of 2 parts. The first is a coverage-guided fitness function (see Algorithm 2), which encourages the exploration of diverse vulnerabilities rather than focusing solely on the easiest ones to find. In our LSTM-based predictor, the process typically begins with the output of an embedding, which is then passed through a fully connected layer to generate the final prediction. We save the embeddings from the historian attack vectors to create an embedding set (Algo 2, Lines 2-4), which is then used to evaluate each candidate during each round of search process.

In the second part of the fitness function, we define it as follows:

$$f(v_s) = \begin{cases} 1 - \frac{\min((v_s - L_s), (H_s - v_s))}{H_s - L_s}, & \text{if } v_s \in [L_s, H_s] \\ 1 + \frac{\min(|v_s - L_s|, |v_s - H_s|)}{H_s - L_s}, & \text{otherwise} \end{cases}$$

where v_s is the value of the current sensor s, L_s is its lower threshold, and H_s is the upper threshold, and $H_s - L_s$ represents the safety range of the sensor. The final fitness value for an attack, calculated by summing the two fitness components described above, measures how close sensor values are to safety limits, guiding the search for both effective and unexplored attack vectors.

In our study, we configure the random search algorithm with a population size of 100. For the genetic algorithm, we set the population size to 100, and used crossover and mutation to generate 20 offspring. These parameters enable the Spear to efficiently determine an attack vector within 10 seconds.

C. The Shield

The Shield can be divided into two stages: first, collecting misclassified data, and then using it for incremental learning to strengthen the defender.

Collect misclassified data*:* In this component, as illustrated in Figure 1, incoming data traces—potentially affected by attacks—are evaluated by the current defender, which may initially be a randomly initialized machine learning classifier. If the defender produces an incorrect judgment, the misclassified data points within the trace are identified and filtered. These points are then incorporated into the exemplar set to retrain the defender using incremental learning techniques.

In addition, it is worth noting that sensor drift [19], [20]—a gradual, linear increase in sensor output—is a common issue in real-valued sensor nodes. If this drift exceeds the tolerance threshold of the CPS system and results in abnormal behavior, the Shield should be able to detect it. Such conditions can be simulated using the attack vector generation strategy described in the previous section. Conversely, if the drift remains within the system's tolerance range, the Shield should refrain from triggering alarms to prevent unnecessary disruption to CPS operations. To simulate sensor drift, a slight offset can be encoded into an attack vector and injected during CPS initialization.

To gather data points for reinforcing the Shield, we first employ the current model to assess incoming traces. This evaluation uses an anomaly detector applied to the entire trace via a sliding window approach. The Shield may exhibit the following types of erroneous judgments:

- Prematurely raising an alarm before the attack is injected,
- Failing to raise an alarm after a valid attack is injected but before system shutdown (false negative),
- Incorrectly raising an alarm following an ineffective attack before the predefined time limit (false positive).

These misclassified instances are extracted and subsequently incorporated into the retraining process to incrementally enhance the Shield's performance over time.

Implementation for TE. When the Spear generates an attack that is misclassified by the anomaly detector, we record the corresponding data log and transform it into a feature vector formatted as $\langle f_0, f_1, \ldots, f_t \rangle$. Here, $f_t = \langle a_{0,t}, \ldots, a_{n,t}, s_{0,t}, \ldots, s_{m,t} \rangle$ denotes the actuator and sensor values at time t, while l_t indicates whether the CPS state at that time is normal or abnormal. To simulate sensor drift, we inject attacks by targeting up to 10 sensors or actuators per injection during the fuzzing process. Offsets are randomly assigned to these targets, with magnitudes constrained to no more than four times their standard deviation under normal operating conditions. This constraint ensures that the simulated drift remains within the TE system's operational tolerance.

The current anomaly detector is then used to analyze the entire trace using a sliding window approach. If the detector produces incorrect judgments, the corresponding traces are saved for use in improving the model. In total, we injected and generated 223 attacks and associated data entries for enhancing the detector. An additional 113 attacks were used to evaluate the performance improvements. A detailed analysis of this dataset will be provided in the evaluation section.

Incremental learning*:* In this step, the Shield leverages the misclassified samples collected previously to further refine its model and enhance its defensive capabilities. The rationale is that the detection model should be promptly updated whenever misclassified samples are identified and reported. However, the complete training data is not always available, and retraining the model solely on recent data is not viable. To address this challenge, we adopt an incremental learning approach. This approach enables the model to incorporate new defense patterns based on recent inputs while retaining knowledge of

earlier strategies. Nonetheless, there are three major challenges in applying incremental learning to refine the detection model: catastrophic forgetting, data imbalance, and plasticity loss. In the following, we describe our strategies to address each of these challenges.

The first challenge is *catastrophic forgetting*, a phenomenon in which a model's performance on previously learned tasks deteriorates when it is trained on new data using conventional methods. In our setting, this arises because the Shield lacks access to the full historical dataset and must be updated solely using newly reported samples. In particular, as the Spear continuously generates data, training with all historical data to prevent catastrophic forgetting is computationally impractical and could delay timely updates. Thus, the Shield must balance efficiency and performance. To address this issue, *replay* is a widely adopted and intuitive strategy that emulates human cognition [23], [24] by revisiting a subset of previously encountered exemplars. In our approach, we utilize the classifier as a feature extractor by capturing the outputs from its final fully connected layer to guide exemplar selection. The goal is to ensure that the selected exemplars remain representative of the current state of the classifier.

We begin by initializing the exemplar set as empty. Prior to each training round, newly selected samples are added to the existing exemplar set, and we compute the mean of their extracted features, denoted as $\bar{\varphi}$. Next, we sort the data points $\{d_i\}$ according to the distance between their corresponding feature representations $\{\varphi_i\}$ and the mean feature $\bar{\varphi}$. A representative subset is then formed by downsampling from this sorted list to serve as the exemplar set for the upcoming round. This procedure enables efficient identification of representative samples, thereby enhancing the model's ability to retain knowledge across sequential tasks.

The second challenge is *data imbalance*, caused by a disproportionate number of normal versus abnormal samples. This problem often occurs due to the characteristics of the Spear and similar tools, which tend to produce many more normal samples, making effective model training more difficult. To mitigate this issue, we incorporate a balance penalty term into the Binary Cross-Entropy (BCE) loss, defined as:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \frac{\lambda}{B} \cdot |C_{\text{normal}} - C_{\text{abnormal}}|$$

Here, $\mathcal{L}_{\text{BCE}}$ denotes the standard binary cross-entropy loss, and B is the batch size. The terms C_{normal} and C_{abnormal} represent the number of correct predictions for the normal and abnormal classes, respectively. The balance factor λ adjusts the strength of the penalty, encouraging the model to maintain balanced performance across both classes.

The last challenge in incremental learning is the *loss of plasticity*, which refers to the model's declining ability to adapt to new information over time. During extended training, we observed that the model initially exhibited steady performance improvements. However, in later stages, its performance deteriorated significantly, indicating a loss of plasticity—i.e., the model's ability to adapt to new information diminished over time. This phenomenon suggests that standard deep learning methods struggle to maintain adaptability under prolonged training. To preserve the model's plasticity, we adopt a strategy known as *continual backpropagation*, a machine learning technique designed to sustain a network's adaptability during incremental learning. Theoretical studies [25] have shown that the plasticity of neural networks tends to decline throughout the course of stochastic gradient descent (SGD), as certain neurons become inactive and cease contributing meaningfully to the learning process.

Algorithm 3: Continual Backpropagation for Training a l-layer DNN

input : Replacement rate ρ, decay rate η, l-layer DNN defender D
output: new defender D'

1 Initialize utilities $\mathbf{u}_{i,j} = 0$ for each neuron $n_{ij} \in D$. Initialize replace flag $\mathbf{c}_j = 0$ for each layer j in D.
2 **for** *each data point* $\boldsymbol{x}$ **do**
3 Pass $\boldsymbol{x}$ through the network for forward propagation.
4 **for** *each layer* i *in* D **do**
5 **for** *each* $neuron_{i,j}$ *in layer* j **do**
6 $\mathbf{u}_{i,j} = \eta \times \mathbf{u}_{i,j} + (1-\eta) \times |h_{i,j}|$; `// h_{i,j} is the value of neuron j in layer i`
7 Find the number of inactive neurons: $\boldsymbol{n}_{\text{inactive}}$ in layer j that $h_{i,j} == 0$.
8 $\mathbf{c}_j = \mathbf{c}_j + \boldsymbol{n}_{\text{inactive}} \times \rho$
9 **if** $\boldsymbol{c}_j > 1$ **then**
10 Find the $neuron_{r,j}$ in layer j with the smallest utility $\mathbf{u}_{r,j}$.
11 Reinitialize $neuron_{r,j}$'s input weights to random numbers.
12 Reinitialize $neuron_{r,j}$'s output weights to 0.
13 $\mathbf{u}_{r,j} = 0$
14 $\mathbf{c}_j = \mathbf{c}_j - 1$

The key idea of continual backpropagation is to identify and reinitialize such inactive neurons dynamically during training, thereby restoring the network's plasticity. Algorithm 3 outlines the steps of this procedure. In each training iteration, we estimate each neuron's contribution to the network by evaluating its activation state and updating a corresponding utility value (Algo 3, line 6). This utility is accumulated from the neuron's hidden outputs: if a neuron consistently produces low activation values, its outputs are unlikely to influence subsequent layers, indicating low contribution to the model's predictions.

After multiple updates, neurons with significantly lower contribution scores $\mathbf{u}_{i,j}$ are identified as *inactive*. The decay rate η is used to compute the running average of these contributions, ensuring stability in measurement over time. Meanwhile, we monitor the number of inactive neurons (Algo 3, line 8), which serves as a trigger for the neuron replacement mechanism. The replacement rate ρ—typically set to a small value—controls the frequency of neuron resets, such that only a single neuron is replaced after hundreds of updates. When the replacement is triggered, the neuron with the lowest utility is selected and its input and output weights are reinitialized. In particular, the output weights are reset to zero to prevent

immediate downstream influence.

This approach enables the continual monitoring of network activity and the adaptive replacement of underperforming units. By injecting controlled randomness and non-gradient-based updates, we maintain the model's flexibility and responsiveness to new data. Importantly, the low replacement rate ensures that the overall network architecture remains stable, while the neuron configuration is gradually refined over time.

Implementation for TE. We begin by randomly initializing a binary Multilayer Perceptron (MLP) as the detector and setting the exemplar set to empty. In each evolution round, the detector attempts to detect attacks. If detection fails, the misclassified data is collected and forwarded to the Shield for refinement. We then construct a training set by combining the misclassified samples with the existing exemplar set, and retrain the Shield using the loss function $\mathcal{L}$ alongside continual backpropagation, incorporating an early stopping criterion to obtain the updated detector.

Following training, the updated detector is employed as a feature extractor to select a new exemplar set from the current training data. This evolution process is repeated until the detector consistently achieves detection performance above a predefined threshold.

IV. Evaluation

We evaluate the effectiveness and performance of our method for constructing an anomaly detector using the TE and RAAW testbeds (Section II) by addressing the following research questions:

- **RQ1 (Attack Discovery)**: How comprehensive is the Spear in discovering potential attack vectors?
- **RQ2 (Attack Detection)**: Can we build a robust Shield capable of accurately detecting attacks?
- **RQ3 (Parameter Sensitivity)**: Is *Evo-Defender* robust to different anomaly detector widths and sliding window sizes?
- **RQ4 (Ablation Study)**: What is the contribution of each module in the Shield evolution system to its overall performance?

RQ1: Attack Discovery

The first RQ aims to assess the Spear's ability to discover real and meaningful attacks, thereby ensuring that the collected data can effectively enhance the anomaly detector and support reliable evaluation. In this experiment, we record all historical data generated during the attack discovery phase for both the TE and RAAW testbeds. Our goal is to establish a foundational understanding of the Spear's performance. Since the test data used in the subsequent RQs is generated by the Spear, this RQ serves to validate its effectiveness and ensures a fair and meaningful evaluation of the anomaly detector.

Furthermore, to analyze the diversity of the attacks discovered by the Spear, we examine the various system states encountered by the CPS under test. We employ t-Distributed Stochastic Neighbor Embedding (t-SNE), a dimensionality reduction technique that preserves pairwise distances while projecting high-dimensional data into a lower-dimensional space, thereby facilitating a clearer interpretation of the system's behavioral patterns.

TABLE I: Attack Results on Training and Testing Sets for TE and RAAW

Dataset	Train	Test
TE		
Total Number	223	113
Attack Successful	205 (91.93%)	103 (91.15%)
Attack Fail	17 (7.62%)	10 (8.85%)
Single	71 (31.84%)	40 (35.40%)
Double	79 (35.43%)	41 (36.28%)
Triple	47 (21.08%)	19 (16.81%)
Quadruple	8 (3.59%)	3 (2.65%)
Quintuple	0 (0.00%)	0 (0.00%)
Reactor Pressure	110 (49.33%)	49 (43.36%)
Reactor Level	74 (33.18%)	33 (29.20%)
Reactor Temperature	0 (0.00%)	0 (0.00%)
Separator Level	110 (49.33%)	62 (54.87%)
Stripper Level	108(48.43%)	47(41.59%)
RAAW		
Total Number	200	100
Attack Fail	33 (16.50%)	21 (21.00%)
Instant Alarm	69 (34.50%)	35 (35.00%)
Attack Successful	98 (49.00%)	44 (44.00%)

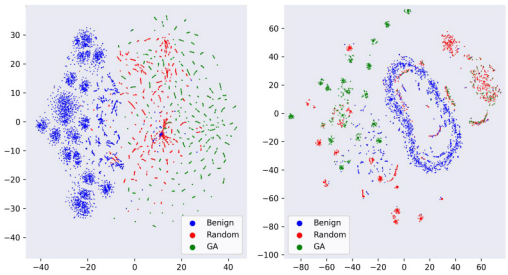


Fig. 5: Diversity Analysis of System States Using t-SNE. **Our approach pushes CPS into diverse states, uncovering complex and subtle attack vectors.**

Setup. In our study, we conducted different attack scenario injections on the TE and RAAW platforms, dividing them into training and evaluation datasets. Each set of attack vectors was generated independently to ensure a diverse testing environment.

Specifically, for the TE platform, attack injections were initiated after 5 hours of system operation, with a maximum allowable runtime of 12 hours. Runs shorter than 5 hours were excluded from our analysis to ensure the system reached a stable state and to eliminate cases where a TE shutdown was caused by aggressive sensor drift before the attack. Attacks that resulted in a system shutdown between the 5 and 12-hour mark were classified as effective, while those that reached the maximum runtime without triggering a shutdown were deemed ineffective. To assess the safety constraints of the TE platform, we identified five critical sensors linked to shutdown events: reactor pressure, reactor level, reactor temperature, separator level, and stripper level. Any readings from these sensors that exceeded predefined safety thresholds resulted in an immediate system shutdown.

In the RAAW experiment, we injected attacks during the

system's operation, resulting in three potential outcomes. It is worth noting that for the robotic arm, it is straightforward to reset to the normal condition. The first scenario involved an ineffective attack, allowing the system to complete the stacking process normally. The second scenario featured an immediately detected effective attack, causing an instant pause, such as a spoofed access control sensor triggering an internal alarm. The third scenario involved an undetected effective attack that disrupted the system's workflow, persisting until the runtime exceeded the predefined operational limit without completing the stacking task. Different attack strategies may lead to the same sequence of visited states. To explore the diversity of abnormal states identified by the Spear, we sampled the training data to assess the states visited by the CPS under test. We ensured that the sampling length matched the anomaly detector's time interval. Each sampling point represents the CPS state over a specific time interval, formatted as a two-dimensional tensor.

Result. Our experiments on the TE and RAAW platforms demonstrate the effectiveness of the Spear's ability to execute successful attacks in the absence of any anomaly detector (Tables I). Specifically, on the TE platform, we collected 223 attacks for the training set and 113 for the test set, achieving an attack success rate exceeding 90%. Notably, more than half of these successful attacks caused one or two sensors to enter abnormal states, with attacks triggering two sensors more frequently than those affecting only one. An analysis of the trigger patterns revealed that all sensors were activated multiple times, except for the *Reactor Temperature*, which did not directly trigger any anomalies. Nevertheless, we identified 14 traces across the dataset in which the reactor temperature increased; in each case, the rise indirectly caused other sensors to exhibit abnormal behavior. This phenomenon occurs because even a slight temperature increase in the TE system can propagate and impact other subsystems, violating safety constraints before the temperature itself crosses the abnormal threshold.

In the RAAW dataset, we injected 200 attacks into the training set and 100 into the test set, with over three-quarters of the attack vectors resulting in successful attacks. Among these, more than half evaded detection by the built-in anomaly detector and disrupted normal operations. For instance, one successful attack set the variable `BLDC_01_Auto_FWD_Start` to 1, causing the conveyor motor to remain continuously active. This led to a misalignment of plastic blocks being pushed from the hopper onto the conveyor, ultimately resulting in material jams that halted the production process.

Figure 5 presents the t-SNE analysis results for data collected from both platforms. On the TE platform, a clear separation is observed between blue (normal) points and green and red (abnormal) points. Samples generated using the GA successfully explored novel regions of the state space that were not covered by the random algorithm. Samples from the same time series tend to form compact clusters, with abnormal samples within a cluster often exhibiting linear distributions. This is primarily due to two factors: the relatively small number of malicious samples necessitated the inclusion of all abnormal points, and the dominance of floating-point features in the TE system resulted in smooth variations across samples.

Similarly, on the RAAW platform, normal and abnormal states are distinctly separated, and samples from the same sequence cluster together. However, due to the prevalence of discrete features in the RAAW dataset, transitions between states exhibit less continuity. Jumps in discrete variables reduce the similarity between consecutive samples, leading to more fragmented and independent clusters.

Overall, attacks generated by the random algorithm tend to trigger states close to the nominal operating conditions, while those generated by the GA algorithm explore a broader and more diverse range of scenarios. These results highlight the differences in how attack algorithms traverse the system's state space and underscore the superior effectiveness of the GA algorithm in uncovering complex and subtle attack vectors.

RQ2: Attack Detection

For this RQ, our goal is to assess the ability of our defender to handle real world attacks mentioned in **RQ1**. For comparison, we select prior CPS anomaly detection methods [3], [26]–[28] as baselines. We evaluate our defender's performance using two complementary methods: *sample classification* and *end-to-end testing*.

Setup. In the sample classification setting, we divide the test data from RQ1 into individual samples, each of which is a window of data consisting of consecutive actuator values and sensor readings, representing the smallest input unit for the defender. Specifically, we segment 103 test logs on the TE platform into 1,144,305 samples of length 150 time points, and 44 logs on the RAAW platform into 17,559 samples of length 50 time points. We assess Shield's performance in terms of accuracy, precision, recall, and F1 score on these samples, thereby evaluating its capability to detect localized temporal anomalies.

In end-to-end testing, we evaluate Shield's attack detection capability using continuous logs corresponding to individual attack injections. Specifically, for the TE dataset, we divide each log into multiple segments, each consisting of 50 time points. A log is labeled as under attack only if there are at least 100 consecutive segments that are detected as anomalous. For the RAAW dataset, we set the segment length to 40 time points and consider a log to be under attack if at least 10 consecutive segments are flagged. These thresholds are based on observed attack patterns across various systems. Their selection will be addressed in the following RQs.

This evaluation assesses the defender's ability to identify previously encountered attacks (memorization) as well as novel, unseen attack instances (generalization). Baseline methods are evaluated on the same datasets for comparative analysis.

Together, these evaluation strategies provide a comprehensive assessment of the defender's classification ability on discrete samples and its real-world effectiveness on continuous operational data, validating both its detection robustness and its stability against sustained attacks.

TABLE II: Extended End-to-End Detection Results on TE and RAAW with Baseline Comparisons. **For TE, our approach achieves more robust performance, whereas the baseline has higher false positives due to over-sensitivity. For RAAW, our method greatly outperforms the baselines and shows strong generalization.**

Defender	Seen Scenario		Unseen Scenario	
	Detection	False Alarm	Detection	False Alarm
TE				
Evo-MLP	97.1% (199/205)	10.2% (21/205)	86.4% (89/103)	6.8% (7/103)
CNN [26]	100% (205/205)	35.1% (72/205)	100.0% (103/103)	36.9% (38/103)
GRU [27]	100% (205/205)	35.1% (72/205)	100.0% (103/103)	34.9% (36/103)
LSTM [3], [28]	100% (205/205)	34.6% (71/205)	100.0% (103/103)	34.9% (36/103)
RAAW				
Evo-MLP	93.7% (89/95)	7.4% (7/95)	86.4% (38/44)	0.0% (0/44)
CNN [26]	11.6% (11/95)	0.0% (0/95)	18.2% (8/44)	0.0% (0/44)
GRU [27]	9.5% (9/95)	0.0% (0/95)	11.4% (5/44)	0.0% (0/44)
LSTM [3], [28]	36.8% (35/95)	0.0% (0/95)	31.8% (14/44)	0.0% (0/44)

TABLE III: Sample Classification Performance on TE and RAAW. **In general, our approach outperform baselines in most metrics.**

Defender	Accuracy	Precision	Recall	F1
TE				
Evo-MLP	**92.03%**	**84.52%**	36.36%	**50.84%**
CNN [26]	68.65%	26.09%	**97.21%**	41.14%
GRU [27]	68.68%	26.09%	97.02%	41.12%
LSTM [3], [28]	68.71%	26.06%	96.73%	41.06%
RAAW				
Evo-MLP	**89.59%**	**100.00%**	**75.42%**	**85.99%**
CNN [26]	60.52%	100.00%	7.02%	13.12%
GRU [27]	60.61%	100.00%	7.22%	13.47%
LSTM [3], [28]	60.70%	100.00%	7.44%	13.84%

Result. Table III presents the sample classification performance of various detection models on the TE and RAAW datasets. Across both datasets, our method consistently outperforms all baselines. On the TE dataset, Evo-MLP achieves the highest accuracy (92.03%), precision (84.52%), and F1 score (50.84%), demonstrating a significant improvement over previous methods by yielding fewer false positives and achieving a better balance between false positives and missed detections. This highlights its overall effectiveness in accurately detecting relevant samples. While deep learning baselines (CNN, GRU, LSTM) achieve recall rates exceeding 99%, further analysis reveals their vulnerability to sensor drift. This results in increased false alarm rates, thereby reducing their overall accuracy and precision compared to our approach.

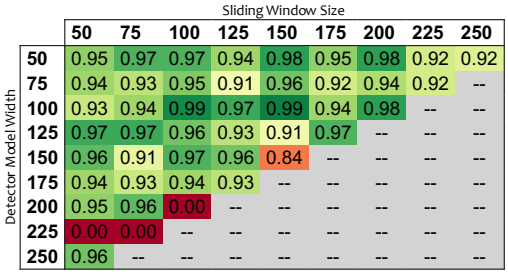
For the RAAW dataset, Evo-MLP again achieves the best overall performance, with an accuracy of 89.59%, precision of 100.00%, recall of 75.42%, and an F1 score of 85.99%. Although the CNN, GRU, and LSTM baselines attain perfect precision, their lower recall and accuracy compared to Evo-MLP indicate a more conservative behavior, leading them to miss certain attack instances.

It is also noteworthy that the baselines demonstrate stronger recall on the TE platform, while exhibiting much higher precision on the RAAW platform. This discrepancy can be attributed to the characteristics of the variables in each testbed. In the TE testbed, many variables are continuous, such as temperature readings, which can range from tens to hundreds of degrees. In contrast, most variables in the RAAW testbed are discrete, representing operational states (e.g., the value of a laser detector is either 0 or 1). As a result, when calculating the difference between predicted and actual values, the prediction model in TE is more likely to produce larger values, potentially exceeding the detection threshold, leading to higher false alarm rates. On the other hand, in the RAAW testbed, abnormal states may manifest in only a few key variables, making it difficult for the loss function to capture the anomaly effectively. This, in turn, increases the likelihood of missed attacks.

Table II summarizes the end-to-end detection performance of various defender models on both platforms. For seen attacks on the TE dataset, Evo-MLP achieved a 97.1% detection rate with a moderate 10.2% false alarm rate. Predictor-based defenders (CNN/GRU/LSTM) detected all attacks (100% detection success) but suffered from high false alarm rates, ranging from 34.6% to 35.1%. On unseen attacks in the TE dataset, Evo-MLP maintained a strong detection rate of 86.4% with a low false alarm rate of 6.8%, while predictor-based defenders continued to achieve perfect detection but triggered high false alarms (35%). These results are consistent with the sample classification performance. They also highlight that predict-based approaches are overly sensitive when applied to chemical systems with numerous continuous sensors.

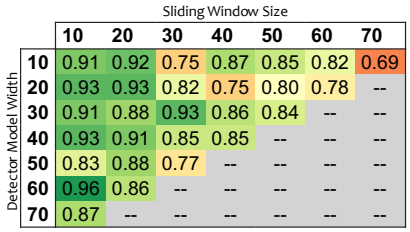
For the RAAW dataset, Evo-MLP achieved 93.7% detection success against known attacks with only 7.4% false alarms. Predictor-based defenders struggled, with CNN achieving only 11.6% success and LSTM performing the best at 36.8%. For unseen RAAW attacks, Evo-MLP continued to perform strongly, achieving 86.4% detection with no false alarms, highlighting its superior generalization capability.

These results show that Evo-MLP maintains strong detection performance and low false alarm rates across datasets and attack scenarios—crucial for CPS operations. In contrast, predictor-based detectors consistently fail, either missing attacks or generating false alarms due to fundamental weaknesses in handling attack patterns and CPS characteristics. This is also evidenced by their low recall in sample classification experiments, highlighting their high false negatives and limited



	50	75	100	125	150	175	200	225	250
50	0.95	0.97	0.97	0.94	0.98	0.95	0.98	0.92	0.92
75	0.94	0.93	0.95	0.91	0.96	0.92	0.94	0.92	--
100	0.93	0.94	0.99	0.97	0.99	0.94	0.98	--	--
125	0.97	0.97	0.96	0.93	0.91	0.97	--	--	--
150	0.96	0.91	0.97	0.96	0.84	--	--	--	--
175	0.94	0.93	0.94	0.93	--	--	--	--	--
200	0.95	0.96	0.00	--	--	--	--	--	--
225	0.00	0.00	--	--	--	--	--	--	--
250	0.96	--	--	--	--	--	--	--	--

(a) TE



	10	20	30	40	50	60	70
10	0.91	0.92	0.75	0.87	0.85	0.82	0.69
20	0.93	0.93	0.82	0.75	0.80	0.78	--
30	0.91	0.88	0.93	0.86	0.84	--	--
40	0.93	0.91	0.85	0.85	--	--	--
50	0.83	0.88	0.77	--	--	--	--
60	0.96	0.86	--	--	--	--	--
70	0.87	--	--	--	--	--	--

(b) RAAW

Fig. 6: Analysis of Anomaly Detector Width and Sliding Window Size. The vertical axis shows the detector model width, while the horizontal axis shows the sliding window size. Performance is displayed as both numeric values and color, with green indicating higher accuracy and red indicating lower accuracy. **Our method allows flexible selection of anomaly detector width and sliding window size, with most settings yielding good performance.**

effectiveness in detecting attacks.

Overall, the comprehensive evaluation shows that our approach outperforms predictor-based baselines in both sample classification and end-to-end detection, consistently achieving high detection rates with low false alarm rates across diverse CPS datasets and attack scenarios. Its strong performance against both known and novel attacks highlights its adaptability and reliability for real-world deployment.

RQ3: Parameter Sensitivity

For this RQ, we aim to show that the widths of the anomaly detector and the sliding window size do not significantly impact the detection performance. These parameters are set based on empirical observations of attack patterns in different systems and experimental results. For the TE dataset, the minimum number of samples from attack injection to system shutdown is 346, so the sum of the anomaly detector's window width and the sliding window size should not exceed this value. In the RAAW dataset, about 480 samples are generated during all stacking operations, with each plastic block producing an average of 80 time units of data, thus requiring a smaller window size.

Setup. This experiment was conducted on both platforms, with parameter choices tailored to each platform's characteristics. For the TE dataset, the width of the anomaly detector model and the sliding window size each ranged from 50 to 250, while for RAAW, both parameters were set between 10 and 70. Since there is a theoretical upper limit for the sum of these two parameters (as described earlier), parameter combinations in the lower right corner of Figures 6a-6b were not tested. To evaluate the detection capabilities and memory of each defender, we performed performance assessments on the validation set using the same approach as in RQ2.

Result. The results in Figure 6 show that our method is robust to most settings. On the TE dataset, detection accuracy exceeded 0.9 for most configurations, with the highest accuracy observed when the anomaly detector width was 100 and the sliding window size was 100 or 150. Certain parameter combinations (e.g., 200-100, 225-50, 225-75) consistently failed to converge during training, as confirmed by repeated experiments and log analysis. This suggests these settings compromised data quality, hindering effective defender evolution. As these cases approach the system's theoretical limits, occasional poor performance is expected and does not affect the overall robustness of our method. On the RAAW dataset, most defender configurations achieved performance above 0.8, with the best results when the width was 60 and the sliding window size was 10. We also found that larger window sizes in RAAW tended to result in slightly lower detection performance.

Overall, comprehensive evaluation indicates that the detection performance of our method is insensitive to the width of the anomaly detector and the sliding window size. This demonstrates that *Evo-Defender* maintains stable performance across different platforms and a variety of parameter configurations, highlighting its strong generalizability.

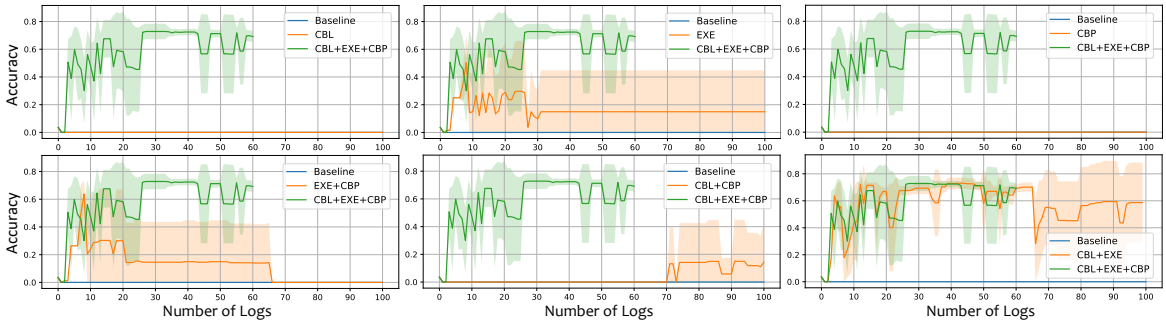
RQ4: Ablation Study

This research question aims to evaluate the impact of each module in the defender evolution system. Recall that our defender evolution comprises three main modules: class balance loss, exemplar data, and continual backpropagation.

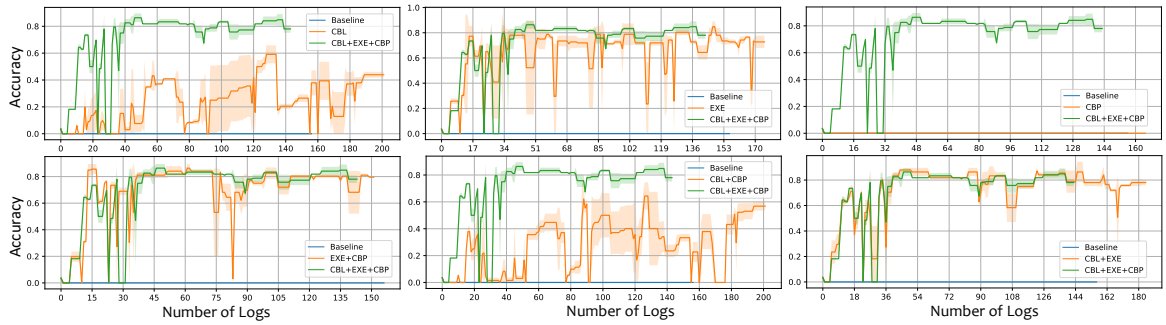
- **CBL** (Class Balance Loss): Balances the defender's learning preference between positive and negative labels within a single round. This is particularly useful when incoming data is unevenly distributed between the two classes, thereby mitigating data imbalance issues.
- **EXE** (Exemplar): Using exemplars, the defender can effectively consolidate knowledge gained from historical data, reinforcing the memory of previously learned information.
- **CBP** (Continual Backpropagation): Incorporates continual backpropagation into the preceding modules, enhancing the defender's adaptability and stabilizing the training process. This also helps reach the predefined stopping condition more quickly.

To systematically evaluate the contributions of the three modules in the Shield, we conducted ablation experiments on both the TE and RAAW platforms. To achieve this, we separately removed different modules of each stage and then compared their results.

Setup. Each platform evaluates eight configurations: (1) Baseline (no modules), (2-4) Single module (Baseline + Exem-



(a) TE



(b) RAAW

Fig. 7: The effect of each module on (a) TE and (b) RAAW. **Overall, all module contributes positively to the evolution process, allowing the defender to reach equal or better performance with less data.**

plar / CBL / CBP), (5-7) Pairwise combinations, and (8) Full module. The evolutionary process terminates when stopping conditions are met or data traces are exhausted. Unlike RQ2, we use a less stringent stopping condition to highlight the performance differences between modules in terms of data consumption. We perform five independent runs for each configuration, standardize sequence lengths by truncating or padding with the final result, discard the highest and lowest values, and calculate the mean and standard deviation of the success detection rate. Length standardization reflects the data needed for each configuration to meet the same stopping criterion, minimizing random variation between runs rather than merely truncating to the shortest length.

Result. For the TE platform, as shown in Figure 7a, the full-module configuration (green line) demonstrates optimal convergence, satisfying stopping criteria in 68 data logs, which is 69.6% less than baseline (224 $\rightarrow$ 68). While the baseline configuration suffers from catastrophic forgetting, it consistently shows lower accuracy throughout. Among the single-module variants, EXE shows initial performance improvement, but as training progresses, it fails to learn from new data and its performance declines significantly in later stages. Other single-module configurations perform similarly to the baseline throughout evolution. In pairwise combinations, CBL+EXE temporarily exceeds the full module's peak accuracy but typically needs more data to meet the same stopping criteria. The accuracy variation of the EXE+CBP combination is similar to that of the EXE single module, while CBL+CBP shows no measurable improvement.

For the RAAW platform, as shown in Figure 7b, the full-module configuration maintains its advantage with 26.28% faster convergence than the baseline (115 vs. 156 points). Behavioral patterns differ significantly from TE: EXE alone achieves the closest performance to the full system, though it trails by 8 percentage points and suffers more performance fluctuations, such as a sharp accuracy drop in accuracy between rounds 60 and 61. Single-module CBL shows reduced effectiveness compared to its performance on TE, while CBP demonstrates platform-specific limitations. The pairwise module combinations demonstrate superior effectiveness compared to individual modules on the RAAW platform. In RAAW's composite module experiments, the combined schemes outperformed single-module approaches. Specifically, while EXE+CBP and CBL+EXE achieved accuracy comparable to the full-module configuration, while the full module achieved the stop condition with less data and fluctuations.

Overall, all three modules provided essential support for the defender's evolution process at varying levels. On TE, CBL offered the greatest performance boost, while combining EXE and CBP notably enhanced stability. On RAAW, the EXE module delivered the most significant performance gain, with CBL and CBP adding stability.

Threats to Validity

There are some threats to the validity of our results. First, our method relies on predictive models, so the quality of the

model may affect the effectiveness of the attack. However, we emphasize that the goal of this approach is to build robust defenders fast and stable, rather than focus on the behavior of smart attackers as in previous test suite generation work [13]. We focus on reconstructing attack scenarios rather than simulating complex strategies.

Second, due to the lack of attack benchmarks, we use independently generated attacks to evaluate the defender. These generated attacks do not necessarily cover all possible tampering by intelligent attackers targeting different aspects of the system, so the results may not apply to certain 0-day attacks. However, our efficient and automated reconstruction of attack scenarios, as well as the robustness demonstrated under continuous adversarial attacks, strengthens our confidence in the approach.

V. Related Work

In this section, we highlight some related work addressing the broader themes of this paper: ensuring the integrity of CPSs and incremental learning.

Several recent research studies have focused on detecting and preventing CPS attacks. Popular solutions include anomaly detection, analyzing logs of running states to identify suspicious events or abnormal behaviors [3], [4], [6]–[9], [29]–[38]; fingerprinting, where sensors are monitored for spoofing by analyzing time and frequency domain features from sensor and process noise [39]–[43]; invariant-based checks, which continuously monitor conditions across processes and components [44]–[53]. The main difference between these works and ours is the lack of adequate and continuous feedback from CPS testing and insufficient understanding of the underlying processes and control operations (e.g., for designing physical invariants); as a result, physics-based anomaly detectors are typically static and cannot dynamically adapt or improve in response to evolving threats. In contrast, our approach not only enables dynamic self-improvement through incremental learning, but also significantly reduces the need for domain-specific knowledge and deployment effort. While causality-guided testing [54] employs formal methods to discover new attacks, Evo-Defender identifies attack vectors mainly using a coverage-based fitness function. We believe these methods can enhance our defender evolution approach; for instance, advanced machine learning models for anomaly detection might augment our incremental defender construction.

The strengths and weaknesses of different countermeasures has been the focus of various studies. Erba and Tippenhauer [55] spoof sensor values (e.g. using precomputed patterns) and are able to evade three black box anomaly detectors published at top security conferences. Urbinaetal. [56] evaluated several attack detection mechanisms in a comprehensive review, concluding that many of them are not limiting the impact of stealthy attacks (i.e. from attackers who have knowledge about the system's defences), and suggest ways of mitigating this. Our defender evolution solution tackles these issues by integrating a smart attacker and employing incremental learning to steadily improve the defender's robustness. Cárdenas et al. [57] propose a general framework for assessing attack detection mechanisms, but in contrast to the previous works, focus on the business cases between different solutions. For example, they consider the cost-benefit trade-offs and attack threats associated with different methods, e.g. centralised vs. distributed.

As a testbed dedicated for cyber-security research, many different countermeasures have been developed for TE itself. These include anomaly detectors, typically trained on the manual crafted datsets or public datasets [58], [59] using deep learning techniques, e.g. [60]–[62]. Du [62] implemented fingerprinting systems based on sensor and process noise for detecting stochastic faults. Aoudi [4] determines anomalies by compressing time-series signals into a low-dimensional subspace and calculating the distance between new inputs and the subspace centroid as an anomaly score to determine anomalies.

The application of incremental learning is a vibrant area of research [63], [64], but targets are typically focus on the class-incremental learning and image classification tasks. To highlight a few examples: Yu et al. [65] introduced a novel Knowledge Refreshing and Consolidation (KRC) framework to address catastrophic forgetting in lifelong learning, enhancing performance on both old and new tasks through bi-directional knowledge transfer and dynamic memory interaction; Buzzega et al. [66] combines rehearsal with knowledge distillation to mitigate catastrophic forgetting in incremental image classification tasks; These examples focus on image classification, whereas Evo-Defender trains models to classify time series data from sensor readings. While some studies, like Qiao's work [67] on class-incremental learning for time series, utilize incremental learning methods on public datasets, they do not address real-world dynamic systems.

VI. Conclusions

In this paper, we introduced *Evo-Defender*, an evolutionary framework that strengthens anomaly detection in cyber-physical systems by pairing a smart attacker with a continually adapting defender. Through extensive evaluations on both simulated and real-world CPS testbeds, we demonstrated that *Evo-Defender* achieves superior detection performance, requiring less data and fewer evolution rounds than traditional approaches. Our ablation studies further highlight the critical roles played by continual backpropagation, exemplar replay, and class balance techniques in stabilizing and enhancing defender performance. By simulating diverse attack behaviors and incrementally evolving the defender, *Evo-Defender* offers a practical and scalable solution for securing complex, data-scarce CPS environments.

Acknowledgment

We sincerely appreciate the anonymous reviewers for their valuable feedback, which significantly enhanced this paper. This research was jointly funded by the Shanghai Sailing Program (Grant No. 23YF1427500), the NSFC Program (Grant No. 62302304), and the ShanghaiTech Startup Funding.

REFERENCES

[1] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[2] R. Lee, J. Slowik, B. Miller, A. Cherepanov, and R. Lipovsky, "Industroyer/crashoverride: Zero things cool about a threat group targeting the power grid," pp. 1–68, 2017.

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

[4] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 817–831. [Online]. Available: https://doi.org/10.1145/3243734.3243781

[5] Z. He, A. Raghavan, S. Chai, and R. Lee, "Detecting zero-day controller hijacking attacks on the power-grid with enhanced deep learning," 06 2018.

[6] M. Kravchik and A. Shabtai, "Detecting cyber attacks in industrial control systems using convolutional neural networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 72–83. [Online]. Available: https://doi.org/10.1145/3264888.3264896

[7] Q. Lin, S. Adepu, S. Verwer, and A. Mathur, "Tabor: A graphical model-based approach for anomaly detection in industrial control systems," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 525–536. [Online]. Available: https://doi.org/10.1145/3196494.3196546

[8] V. Narayanan and R. B. Bobba, "Learning based anomaly detection for industrial arm applications," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 13–23. [Online]. Available: https://doi.org/10.1145/3264888.3264894

[9] P. Schneider and K. Böttinger, "High-performance unsupervised anomaly detection for cyber-physical system networks," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–12. [Online]. Available: https://doi.org/10.1145/3264888.3264890

[10] "iTrust Labs: Datasets," https://itrust.sutd.edu.sg/itrust-labs_datasets/, 2025, accessed: May 2025.

[11] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Critical Information Infrastructures Security*, G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, Eds. Cham: Springer International Publishing, 2017, pp. 88–99.

[12] Y. Chen, C. M. Poskitt, J. Sun, S. Adepu, and F. Zhang, "Learning-guided network fuzzing for testing cyber-physical system defences." Nov 2019. [Online]. Available: http://dx.doi.org/10.1109/ase.2019.00093

[13] Y. Chen, B. Xuan, C. M. Poskitt, J. Sun, and F. Zhang, "Active fuzzing for testing and securing cyber-physical systems," Jul 2020. [Online]. Available: http://dx.doi.org/10.1145/3395363.3397376

[14] N. Lawrence Ricker, "Decentralized control of the Tennessee Eastman challenge process," *Journal of Process Control*, vol. 6, no. 4, pp. 205–221, 1996. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0959152496000315

[15] "evo-defender-artifacts," https://github.com/StoodCoronet/evo-defender, 2025, gitHub repository, accessed August 13, 2025.

[16] P. Lyman and C. Georgakis, "Plant-wide control of the Tennessee Eastman problem," *Computers & Chemical Engineering*, vol. 19, no. 3, pp. 321–331, 1995. [Online]. Available: https://www.sciencedirect.com/science/article/pii/009813549400057U

[17] C. Fung, E. Zeng, and L. Bauer, "Attributions for ML-based ICS anomaly detection: From theory to practice," in *Proceedings of the 31st Network and Distributed System Security Symposium*. Internet Society, 2024.

[18] A. Takanen, J. D. Demott, and C. Miller, *Fuzzing for Software Security Testing and Quality Assurance*, 2nd ed. USA: Artech House, Inc., 2018.

[19] S. U. Jan, U. Saeed, and I. Koo, "Machine learning for detecting drift fault of sensors in cyber-physical systems," in *2020 17th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, 2020, pp. 389–394.

[20] S. Munirathinam, "Drift detection analytics for IoT sensors," *Procedia Computer Science*, vol. 180, pp. 903–912, 2021, proceedings of the 2nd International Conference on Industry 4.0 and Smart Manufacturing (ISM 2020). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050921003951

[21] D. Bailey and E. Wright, *Practical SCADA for industry*. Elsevier, 2003.

[22] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[23] M. A. Wilson and B. L. McNaughton, "Reactivation of hippocampal ensemble memories during sleep," *Science*, vol. 265, no. 5172, pp. 676–679, Jul. 1994.

[24] A. Tambini and L. Davachi, "Persistence of hippocampal multivoxel patterns into postencoding rest is related to memory," *Proceedings of the National Academy of Sciences*, vol. 110, no. 48, pp. 19 591–19 596, 2013. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1308499110

[25] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton, "Loss of plasticity in deep continual learning," *Nature*, vol. 632, no. 8026, pp. 768–774, August 2024. [Online]. Available: https://doi.org/10.1038/s41586-024-07711-7

[26] C. Fung, S. Srinarasi, K. Lucas, H. B. Phee, and L. Bauer, "Perspectives from a comprehensive evaluation of reconstruction-based anomaly detection in industrial control systems," in *ESORICS 2022: 27th European Symposium on Research in Computer Security*, Sep. 2022.

[27] M. Kravchik and A. Shabtai, "Efficient cyber attacks detection in industrial control systems using lightweight neural networks," *ArXiv*, vol. abs/1907.01216, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:195776402

[28] G. Zizzo, C. Hankin, S. Maffeis, and K. Jones, "Intrusion detection for industrial control systems: Evaluation analysis and adversarial attacks," *ArXiv*, vol. abs/1911.04278, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:207852621

[29] S. Adepu, F. Brasser, L. Garcia, M. Rodler, L. Davi, A.-R. Sadeghi, and S. Zonouz, "Control behavior integrity for distributed cyber-physical systems," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, 2020, pp. 30–40.

[30] E. Aggarwal, M. Karimibiuki, K. Pattabiraman, and A. Ivanov, "CORGIDS: A correlation-based generic intrusion detection system," in *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy*, ser. CPS-SPC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 24–35. [Online]. Available: https://doi.org/10.1145/3264888.3264893

[31] M. Macas and C. Wu, "An unsupervised framework for anomaly detection in a water treatment system," in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, 2019, pp. 1298–1305.

[32] L. Cheng, K. Tian, and D. D. Yao, "Orpheus: Enforcing cyber-physical execution semantics to defend against data-oriented attacks," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, ser. ACSAC '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 315–326. [Online]. Available: https://doi.org/10.1145/3134600.3134640

[33] T. K. Das, S. Adepu, and J. Zhou, "Anomaly detection in industrial control systems using logical analysis of data," *Computers & Security*, vol. 96, p. 101935, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404820302121

[34] Y. Harada, Y. Yamagata, O. Mizuno, and E.-H. Choi, "Log-based anomaly detection of CPS using a statistical method," in *2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP)*, 2017, pp. 1–6.

[35] Z. He, A. Raghavan, G. Hu, S. Chai, and R. Lee, "Power-grid controller anomaly detection with enhanced temporal deep learning," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 160–167.

[36] J. Kim, J.-H. Yun, and H. C. Kim, "Anomaly detection for industrial control systems using sequence-to-sequence neural networks," in *Computer Security*, S. Katsikas, F. Cuppens, N. Cuppens, C. Lambrinoudakis,

C. Kalloniatis, J. Mylopoulos, A. Antón, S. Gritzalis, F. Pallas, J. Pohle, A. Sasse, W. Meng, S. Furnell, and J. Garcia-Alfaro, Eds. Cham: Springer International Publishing, 2020, pp. 3–18.

[37] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011, pp. 2195–2201.

[38] T. Schmidt, F. Hauer, and A. Pretschner, "Automated anomaly detection in CPS log files," in *Computer Safety, Reliability, and Security*, A. Casimiro, F. Ortmeier, F. Bitsch, and P. Ferreira, Eds. Cham: Springer International Publishing, 2020, pp. 179–194.

[39] C. M. Ahmed, A. P. Mathur, and M. Ochoa, "NoiSense Print: Detecting data integrity attacks on sensor measurements using hardware-based fingerprints," *ACM Trans. Priv. Secur.*, vol. 24, no. 1, Sep. 2020. [Online]. Available: https://doi.org/10.1145/3410447

[40] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems," 01 2016.

[41] Q. Gu, D. Formby, S. Ji, H. Cam, and R. Beyah, "Fingerprinting for cyber-physical system security: Device physics matters too," *IEEE Security & Privacy*, vol. 16, no. 5, pp. 49–59, 2018.

[42] M. Kneib and C. Huth, "Scission: Signal characteristic-based sender identification and intrusion detection in automotive networks," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 787–800. [Online]. Available: https://doi.org/10.1145/3243734.3243751

[43] K. Yang, Q. Li, X. Lin, X. Chen, and L. Sun, "iFinger: Intrusion detection in industrial control systems via register-based fingerprinting," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 5, pp. 955–967, 2020.

[44] S. Adepu and A. Mathur, "Distributed detection of single-stage multipoint cyber attacks in a water treatment plant," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, ser. ASIA CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 449–460. [Online]. Available: https://doi.org/10.1145/2897845.2897855

[45] A. Sridhar and M. Aditya, "Using process invariants to detect cyber attacks on a water treatment system," in *ICT Systems Security and Privacy Protection*, J.-H. Hoepman and S. Katzenbeisser, Eds. Cham: Springer International Publishing, 2016, pp. 91–104.

[46] S. Adepu and A. Mathur, "Distributed attack detection in a water treatment plant: Method and case study," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 1, pp. 86–99, 2021.

[47] A. A. Cárdenas, S. Amin, Z.-S. Lin, Y.-L. Huang, C.-Y. Huang, and S. Sastry, "Attacks against process control systems: risk assessment, detection, and response," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 355–366. [Online]. Available: https://doi.org/10.1145/1966913.1966959

[48] Y. Chen, C. M. Poskitt, and J. Sun, "Towards learning and verifying invariants of cyber-physical systems by code mutation," in *FM 2016: Formal Methods*, J. Fitzgerald, C. Heitmeyer, S. Gnesi, and A. Philippou, Eds. Cham: Springer International Publishing, 2016, pp. 155–163.

[49] Y. Chen, C. Poskitt, and J. Sun, "Learning from mutants: Using code mutation to learn and monitor invariants of a cyber-physical system," in *2018 IEEE Symposium on Security and Privacy (SP)*, 2018, pp. 648–660.

[50] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 801–816. [Online]. Available: https://doi.org/10.1145/3243734.3243752

[51] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Comput. Surv.*, vol. 51, no. 4, Jul. 2018. [Online]. Available: https://doi.org/10.1145/3203245

[52] M. A. Umer, A. Mathur, K. N. Junejo, and S. Adepu, "Generating invariants using design and data-centric approaches for distributed attack detection," *International Journal of Critical Infrastructure Protection*, vol. 28, p. 100341, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548220300056

[53] C. H. Yoong, V. R. Palleti, R. R. Maiti, A. Silva, and C. M. Poskitt, "Deriving invariant checkers for critical infrastructure using axiomatic design principles," *Cybersecurity*, vol. 4, no. 1, p. 6, April 2021. [Online]. Available: https://doi.org/10.1186/s42400-021-00069-7

[54] C. M. Poskitt, Y. Chen, J. Sun, and Y. Jiang, "Finding causally different tests for an industrial control system," in *Proceedings of the 45th International Conference on Software Engineering*, ser. ICSE '23. IEEE Press, 2023, p. 2578–2590. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00215

[55] A. Erba and N. O. Tippenhauer, "No need to know physics: Resilience of process-based model-free anomaly detection for industrial control systems," *CoRR*, vol. abs/2012.03586, 2020.

[56] D. I. Urbina, J. A. Giraldo, A. A. Cardenas, N. O. Tippenhauer, J. Valente, M. Faisal, J. Ruths, R. Candell, and H. Sandberg, "Limiting the impact of stealthy attacks on industrial control systems," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1092–1105. [Online]. Available: https://doi.org/10.1145/2976749.2978388

[57] A. A. Cárdenas, R. Berthier, R. B. Bobba, J. H. Huh, J. G. Jetcheva, D. Grochocki, and W. H. Sanders, "A framework for evaluating intrusion detection architectures in advanced metering infrastructures," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 906–915, 2014.

[58] Kaspersky, "Tennessee Eastman process with cyber-attacks dataset," https://box.kaspersky.com/, 2015, accessed: 2025-04-14.

[59] T. Rieth *et al.*, "Additional Tennessee Eastman process simulation data for anomaly detection evaluation," Harvard Dataverse, 2017, data referenced in Rieth et al. (2017). Issues and Advances in Anomaly Detection Evaluation for Joint Human-Automated Systems. Presented at Applied Human Factors and Ergonomics 2017. Sponsored by the Office of Naval Research under contract N00014-15-C-5003. Public domain data. [Online]. Available: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6C3JR1

[60] X. Yang and D. Feng, "Generative adversarial network based anomaly detection on the benchmark Tennessee Eastman process," in *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, 2019, pp. 644–648.

[61] P. Filonov, F. Kitashov, and A. Lavrentyev, "RNN-based early cyber-attack detection for the Tennessee Eastman process," 09 2017.

[62] I. Lomov, M. Lyubimov, I. Makarov, and L. E. Zhukov, "Fault detection in Tennessee Eastman process with temporal deep learning models," *Journal of Industrial Information Integration*, vol. 23, p. 100216, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2452414X21000145

[63] M. Masana, X. Liu, B. Twardowski, M. Menta, A. D. Bagdanov, and J. van de Weijer, "Class-incremental learning: Survey and performance evaluation on image classification," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 5, p. 5513–5533, May 2023. [Online]. Available: https://doi.org/10.1109/TPAMI.2022.3213473

[64] D.-W. Zhou, Q.-W. Wang, Z.-H. Qi, H.-J. Ye, D.-C. Zhan, and Z. Liu, "Class-incremental learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 12, p. 9851–9873, Dec. 2024. [Online]. Available: https://doi.org/10.1109/TPAMI.2024.3429383

[65] C. Yu, Y. Shi, Z. Liu, S. Gao, and J. Wang, "Lifelong person re-identification via knowledge refreshing and consolidation," in *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023. [Online]. Available: https://doi.org/10.1609/aaai.v37i3.25436

[66] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[67] Z. Qiao, Q. Pham, Z. Cao, H. H. Le, P. N. Suganthan, X. Jiang, and S. Ramasamy, "Class-incremental learning for time series: Benchmark and evaluation," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, ser. KDD '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 5613–5624. [Online]. Available: https://doi.org/10.1145/3637528.3671581