2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3503-7770-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IROS58592.2024.10802522

RoboCop: A Robust Zero-Day Cyber-Physical Attack Detection Framework for Robots

Upinder Kaur[1], Z. Berkay Celik[2], and Richard M. Voyles[3]

***Abstract*— Zero-day vulnerabilities pose a significant challenge to robot cyber-physical systems (CPS). Attackers can exploit software vulnerabilities in widely-used robotics software, such as the Robot Operating System (ROS), to manipulate robot behavior, compromising both safety and operational effectiveness. The hidden nature of these vulnerabilities requires strong defense mechanisms to guarantee the safety and dependability of robotic systems. In this paper, we introduce RoboCop, a cyber-physical attack detection framework designed to protect robots from zero-day threats. RoboCop leverages static software features in the pre-execution analysis along with runtime state monitoring to identify attack patterns and deviations that signal attacks, thus ensuring the robot's operational integrity. We evaluated RoboCop on the F1-tenth autonomous car platform. It achieves a 93% detection accuracy against a variety of zero-day attacks targeting sensors, actuators, and controller logic. Importantly, in on-robot deployments, it identifies attacks in less than 7 seconds with a 12% computational overhead.**

I. Introduction

Robot cyber-physical systems (Robot CPS) integrate computational algorithms with physical agents to perform complex tasks in the real world, transforming industries with applications such as self-driving cars [1], robot assistants [2], and delivery drones [3]. However, as Robot CPS become increasingly prevalent in everyday life, they face cyber-physical attacks.

Cyber-physical attacks refer to malicious actions that exploit vulnerabilities in both the cyber (software, networks) and physical (sensors, actuators) components of a robotic system [4]. The attack goal is to disrupt or manipulate the robot's behavior, potentially leading to safety hazards, privacy breaches, or operational failures. For example, attackers have remotely manipulated the messages from the Robot Operating System (ROS) to cause malfunctions in surgical robots, compromising patient safety [5]. Similarly, drones have also been compromised to illegally capture videos of critical sites, jeopardizing national security and confidentiality [6].

Despite recent considerable advances in vulnerability discovery and security enforcement, e.g., [7], [8], securing robot CPS presents unique challenges that remain unaddressed. The tight coupling of computational and physical elements creates a more complex and multifaceted attack surface [9]–[11]. One particularly dangerous threat is posed by zero-day vulnerabilities that are not yet publicly known or discovered. In the context of Robot CPS, these vulnerabilities are especially dangerous, as they exploit the complex interactions between software and hardware, allowing attackers to gain control before defenses can be deployed. For example, a strategically placed piece of tape tricked an autonomous car into exceeding speed limits [12]. The widespread use of common platforms, such as ROS, exacerbates the risk by making multiple systems vulnerable to the same exploit [13].

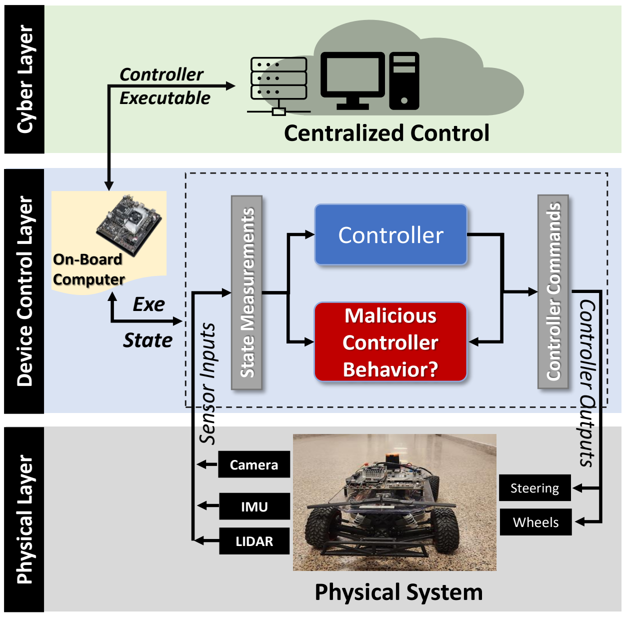

Fig. 1. RoboCop validates the overall robot behavior to identify attacks launched through the cyber or physical layers.

Current security frameworks for Robot CPS primarily focus on known attack vectors, such as sensor spoofing, and often address detection and mitigation after an attack has occurred [11], [14]–[18]. Although these post-attack strategies are valuable, they are insufficient to anticipate and defend against zero-day vulnerabilities, which cannot be mitigated using traditional defense methods [5], [19]–[21]. Furthermore, many frameworks do not fully consider the hybrid nature of robotics systems, where the interplay between software and hardware creates novel attack vectors not typically found in conventional IT systems.

In this paper, we focus on zero-day cyber-physical attacks targeting robot CPS. We map the extensive attack surface, identifying vulnerabilities arising from manipulations in the

*Kaur and Voyles acknowledge support from USDA 2018-67007-28439 and NSF CNS-1450342, while Celik acknowledges support from NSF CNS-2144645 and IIS-2229876 in the fulfillment of this work.

[1] Upinder Kaur is with the Agricultural and Biological Engineering Department, Purdue University, West Lafayette, Indiana, USA. kauru@purdue.edu
[2] Z. Berkay Celik is with the Department of Computer Science, Purdue University, West Lafayette, Indiana, USA. zcelik@purdue.edu
[3] Richard M. Voyles is with the School of Engineering Technology, Purdue Polytechnic Institute, Purdue University, West Lafayette, Indiana, USA. rvoyles@purdue.edu

code that govern sensors, actuators, and controllers. By analyzing known weaknesses within ROS and widely used robotic software stacks [22], [23], we propose a novel detection framework RoboCop, aimed at identifying and mitigating zero-day cyber-physical threats, particularly those exploiting unauthorized code injections to compromise critical components. RoboCop is a two-stage approach to protect robotic systems. In the pre-execution stage, the framework scrutinizes the robot's software stack, using zero-shot learning to detect anomalies indicative of potential zero-day exploits. During runtime, RoboCop employs a reinforcement learning-based monitoring system that continually assesses the robot's behavior in real time, ensuring swift detection and mitigation of attacks bypassing initial defenses.

In this paper, we make the following contributions.

- We introduce RoboCop, a framework that combines pre-execution evaluation and real-time hardware-aware monitoring for the comprehensive detection of zero-day cyber-physical attacks in robots. This allows RoboCop to identify such attacks before execution and to continuously monitor for anomalies during operation.
- We introduce a pre-execution evaluation model built on zero-shot learning. This model establishes a latent feature space to identify unseen cyber-physical attacks, preventing them from gaining control of the robot.
- RoboCop also comprises a reinforcement learning-based online monitoring. This approach enables real-time detection of malicious intent across short and long horizons by continuously monitoring static and dynamic robot behaviors.
- We validated the effectiveness of RoboCop on a robot vehicle using simulation and real-world tests. Additionally, we evaluated its latency and computational overhead to ensure suitability for resource-constrained hardware.

II. Threat Model and Attack Vectors

The architecture of robot CPS includes a cyber layer where users transmit commands and executables to the system and a controller layer that integrates these commands and processes sensor inputs from the physical layer to generate control outputs. These outputs, in turn, actuate the mechanical components in the physical layer, as depicted in Fig. 1. In this section, we present a taxonomy of attack vectors relevant to robotic CPS, highlighting the multifaceted nature of potential threats that affect the system.

A. Attack Vectors for Robots

Robot systems, by their very nature, are a collection of sensors, actuators, and control units, each presenting unique attack surfaces. Sensors, which serve as the eyes of the system, can be deceived by spoofing or jamming [16], [24], [25], leading to erroneous perception of data. Actuators translate computational commands into physical action and can be hijacked to execute unintended movements or operations. At the core, the controller, whether a simple PID controller or an advanced neural network, interprets sensor data to issue commands to actuators. Compromising this central decision-making unit can have cascading effects, affecting the robot's functionality and mission [14], [26], [27]. We broadly classify these attacks within the context of robot CPS as follows:

- *Sensor Attacks:* These involve the manipulation of sensor outputs to feed false data to the controller, thereby skewing its perception of the environment or the robot's state. Attackers might employ techniques, from simple noise injection to sophisticated adversarial signal spoofing, to manipulate sensor readings [28]. For instance, malicious code can obfuscate the true measurements and inject randomized noise to cause the robot to misbehave. Stealthy attacks also ensure that the true value z_k of any sensor k is altered within the accepted threshold Z_T to avoid easy detection.
- *Controller Attacks:* The controller is the decision-making component of a robot. Attacks targeting controller logic could recalibrate control algorithms, alter command execution, or disrupt the robot's intended behavior, potentially leading to detrimental operational outcomes. For example, in the case of a PID controller, the output is calculated as a function of the tracked error and the current state, as follows:

$$u(t) = K_p e(t) + K_i \int_0^t e(t')dt' + K_d \frac{de(t)}{dt} \quad (1)$$

where K_p, K_i, K_d are the proportional, integral, and derivative gains, $e(t)$ is the current error of the system and $u(t)$ is the control command. The heading is calculated as a correction to the controller error. Alteration of any of these parameters will lead to erratic behavior, which may even lead to physical crashes.
- *Actuator Attacks:* By tampering with the command signals to the actuators, adversaries can orchestrate unauthorized actions or induce mechanical failures. This challenges the system's ability to execute controlled movements or tasks. Therefore, instead of the true actuation y_t, at any instant t given inputs x_t, the actuators receive input $\hat{y}_t$. For a general robot vehicle, these can be of two types: velocity attacks and steering attacks. Velocity attacks may cause the robot to go faster or slower than intended, while steering attacks may cause it to have swerving motion; both may result in collisions.

We highlight that these attacks are often designed to be stealthy. In addition, as robot code becomes increasingly available as shared open-source software, the risk of crafting attacks targeting widely used robotic platforms, such as robotic arms and even surgical robots, increases significantly.

B. Threat Model

We consider an attacker who has a deep understanding of the target robot's software and hardware, including its controller software, sensor suite, and actuator configurations. This knowledge enables the attacker to identify and exploit zero-day vulnerabilities. The prevalence of open-source software in robotics makes this a realistic threat model. The

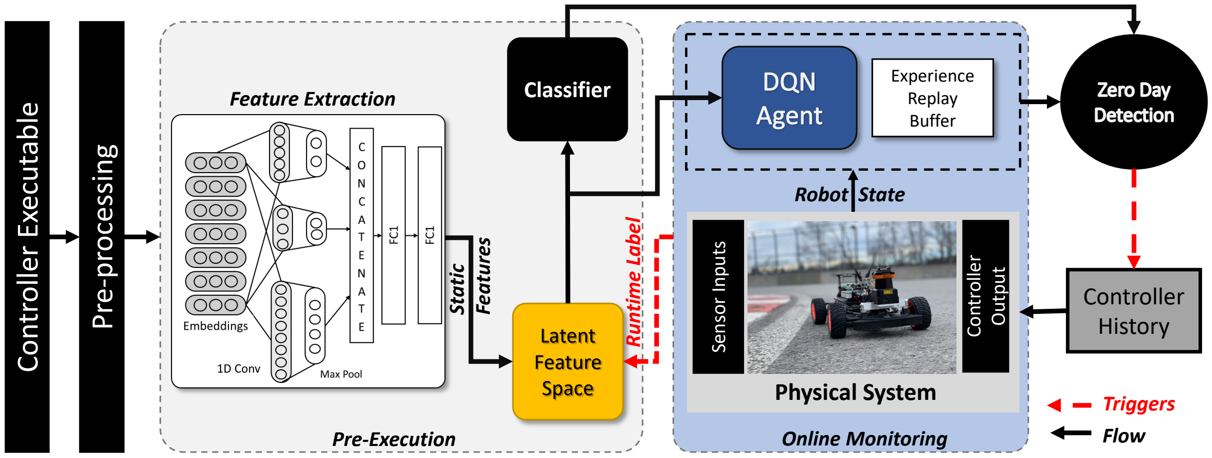

Fig. 2. The RoboCop architecture illustrating the pipeline from pre-execution to online monitoring for zero-day attack detection.

attacker's objective is to compromise the robot's operation by injecting malicious code that can manipulate sensor data, disrupt controller logic, and hijack actuators. The attacker's in-depth knowledge of the system allows them to craft attacks that exploit the complex interactions between the cyber and physical components of the robot, making them particularly challenging to identify and mitigate. The evaluation of RoboCop in Section IV focuses on its ability to defend against such sophisticated zero-day cyber-physical attacks.

III. The RoboCop Framework

A. *Overview of* RoboCop

The RoboCop framework detects cyber-physical attacks in two stages: (1) *Pre-Execution:* Using a zero-shot learning-based classifier on the raw controller software to detect known and unknown attacks, and (2) *Online Monitoring:* which uses a deep reinforcement learning agent to monitor robot behavior during run time to stop malicious attacks, as shown in Fig. 2. Together, this system helps prioritize safe operation by evaluating the immediate and long-term consequences of the state and actions of the robot.

B. *Pre-Execution*

In the pre-execution stage, we parse the controller executable using a multilayer neural network to extract features. The neural network acts as a zero-shot classifier, extracting features from known classes in a given dataset and encoding them into a latent feature space. Using a center-loss-based model, we encoded the similarity map onto this latent feature map. For unknown classes, the model identifies their distance from known classes for classification.

Feature Extraction. To extract key features, each input binary executable is converted into a string of hexadecimal values. As shown in Fig. 2, the network consists of an embedding layer followed by three parallel one-dimensional convolutional neural networks (1D CNN) and max pooling layers. The string is encoded using an embedding layer with a vocabulary of 257 (0 to 256 hex representation). This encoding ensures a fixed-length representation, allowing finer-grained learning and the capture of intrinsic semantic relationships.

The refined inputs are then parsed through parallel 1D CNN layers. These layers, with different filter sizes, capture various hierarchies of local mappings. The parallel architectures of CNNs optimize for speed and outperform serial structures. Finally, the pooled and concatenated layers create the latent feature space. Mathematically, the output of this component can be formulated as:

$$X_{c_j} = f_m^{s_j}(\sigma_1(f_c^{k_j}(e^{d_w}(x_i)))), 1 \leq j \leq 3, \quad (2)$$

$$\mathcal{C} = [X_{c_1}, X_{c_2}, X_{c_3}], \quad (3)$$

$$\mathcal{X}_d = \sigma_3(w_2^T \sigma_2(w_1^T(\mathcal{C}) + b_1) + b_2) \quad (4)$$

where, e^d and $f_c^{k,w}$ represent the embedding layer with input dimension d_w and convolutional layers j with kernel size k, respectively.

The output of the convolutional layers is grouped using the maximum pooling layers f_m^s with stride s to generate a set of features X_{c_j}, where $1 \leq j \leq 3$ for the three parallel layers. The output of the parallel max pooling layers is concatenated as $\mathcal{C}$, which is then input into the two dense layers with weight w_1, w_2 and biases b_1, b_2. Each layer has its activation function $\sigma(.)$. During training, the network is trained with the categorical cross-entropy loss $\mathcal{L}_{ce}$ for the known set of attack classes and center loss $\mathcal{L}_c$ to increase the distance among these classes in the latent space, defined as:

$$\mathcal{L}_{ce} = \frac{-1}{s} \sum_{i=1}^{s} y_i log(\mathcal{F}(X_d(x_i))) \quad (5)$$

$$\mathcal{L}_c = \frac{1}{2} \sum_{i=1}^{s} \max(\|x_{y_i}^* - c_{y_i}\|_2^2, 1) \quad (6)$$

$$\mathcal{L}_T = \mathcal{L}_{ce} + \lambda_c * \mathcal{L}_c \quad (7)$$

where s is the batch size, c is the center of the y^{th} class, where $y_i \in k$. While y_i is the given label, the classifier label is derived by $\mathcal{F}(X_d(x_i))$, where x_i is the executable input controller. The total loss $\mathcal{L}_T$ is the sum of the categorical cross-entropy loss and the center loss, where λ_c is a hyperparameter used to balance the influence of the center loss.

During *test phase*, we extract the characteristic vector $(\mathcal{X}_d(x_t))$. We calculate the Mahalanobis distance $d_M(.,.)$ between this feature vector and the known class centers in the latent feature space, defined as:

$$d_M(\mathcal{X}_d(x_t), c_k) = \sqrt{(\mathcal{X}_d(x_t) - c_k)A_k^{-1}(\mathcal{X}_d(x_t) - c_k)^T} \tag{8}$$

where A is the feature transformation matrix for each class k in the latent feature space. To distinguish between known and unknown classes, we use a threshold Θ, which is the minimum distance between the centers of known classes. If the distance for any test feature $d_t < \Theta$, then it is a known class; otherwise, it is an unknown class. The classifier $\mathcal{F}$ assigns a label to the input with probability p. If this probability is below a threshold Θ_2, the online monitoring investigates the robot behavior. The controller is allowed to run if the binary is deemed benign or stopped if it is deemed malicious.

C. Online Monitoring

In online monitoring, the goal is to detect signs of malicious behavior as early as possible to prevent catastrophic deviations, such as crashes. We formulate this as a Markov Decision Process (MDP) with state (s_t), action (a_t), and reward (r_t) in a given environment at a given time t. The state includes the robot sensor inputs (IMU, LiDAR, etc.), the actuator commands (velocity, steering angle), and the controller errors operating at the sampling frequency of the controller block. This formulation allows us to model the trade-off between the urgency of correct decision-making while also ensuring continuity of operation.

The online monitoring stage operates on the robot computer in parallel to the main robot control. At any time t, it decides whether the robot behaves expectantly or maliciously, choosing whether to CONTINUE $(a_t = 0)$ or STOP $(a_t = 1)$. The STOP action triggers a reset of the controller block to the last known good state, stored in the history buffer. Thus, the system has a sparse action space in a discrete observation space. The model learns a policy $\pi(a_t|s_t)$ that learns the appropriate action for a given state of the robot. Formally, this is defined as

$$\pi_t(a_t|s_t) = \arg\max_{a^i} Q'(s_t, a_t^i|\theta_t), \text{where} i = \{0, 1\} \tag{9}$$

here, i signifies the action, and $Q(.)$ is the action-value function. The rewards are determined based on two factors: 1) assessing the behavior of the robot as the correct label and 2) making the decision as fast as possible to prevent catastrophic events such as crashes. Therefore, the reward is scaled over time, defined as

$$r_t(s_t, a_t^i) = \begin{cases} R(1 - \beta\frac{t}{T}), & \text{if } y_i = L \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

where, L is the true label and y_i is label chosen by the agent. The maximum possible reward R is scaled by a factor β within the time horizon T to force the agent to make faster decisions.

Algorithm 1: The RoboCop Pre-Execution Training

```
1  Input: $\mathcal{D}_{train}$, $M\{Em, Conv(W_c), d(W), \sigma, \mathcal{L}_{ce}, \mathcal{L}_c\}$
2  Output: $M(w)$
3  for $epoch \leq Epochs$ do
4      $N_b \leftarrow D_{train}$;
5      for $n(x, y)$ in $N_b$ do
6          $y_p \leftarrow Mx$;
7          $g_w \leftarrow \mathcal{L}_{ce}(y_p, y)$;
8          $e_w \leftarrow \mathcal{L}_c(y_p, y)$;
9          Store $g_w, e_w$;
10     end
11     Update $w \leftarrow w + \alpha \times BackProp(w, g_w)$;
12     Update $c_i$
13     Store $w, c_i$;
14 end
```

Algorithm 2: The RoboCop Online Monitoring

```
1  Input: Agent $A$, actions $[a_s, a_c]$
2  Output: $\pi_t(a_t \| s_t)$
3  for $epoch$ do
4      for $e \in episodes$ do
5          $s_t \leftarrow envt$;
6          $a_t \in Q(s_t, a)$;
7          $A \leftarrow ra_t$; $s_{t+1} \leftarrow \hat{s}_t$;
8          Update $Q(s, a) \leftarrow Q + r(s_t, a_t) + \gamma Q'(\hat{s}_t, \hat{a}_t \| \theta_t)$;
9          Store $H_{buff} \leftarrow [Q, s, a]$;
10     end
11 end
```

The agent uses a double Q learning algorithm to approximate the action value function. The Q-function is updated after each batch sample as,

$$Q(s_t, a_t^i|\theta_t) = r(s_t, a_t^i) + \gamma Q'(s_t, a_t^i|\theta_t) \tag{11}$$

here, R is the reward obtained, γ is the discount factor, and T is the time horizon. The scaled reward encourages the agent to take the STOP action when a malicious label is detected and to minimize the time it takes to detect anomalies. RL with time-scaled rewards is inherently suited for such sequential decision-making processes, as it evaluates the long-term consequences of actions, allowing proactive and preventive responses to threats.

IV. Experiments and Results

To evaluate the effectiveness of RoboCop, we conducted a series of experiments that focused on both simulation and real world environments. These experiments were designed to test its resilience against a wide range of cyber-physical attacks, specifically targeting zero-day vulnerabilities on the F1-tenth autonomous car platform [29]. Our objective is to validate the detection capabilities of RoboCop in various attack scenarios, benchmark its performance against existing

TABLE I

THE PERFORMANCE RESULTS

Model	Globalized Evaluation				Zero Day Evaluation			
	Accuracy	Precision	F1 Score	FNR	Sensor	Controller	Velocity	Steering
MLP [20]	0.70	0.71	0.71	0.3	0.77	0.78	0.63	0.56
RF [20]	0.68	0.65	0.68	0.28	0.66	0.68	0.63	0.69
SeqCNN [20]	0.76	0.79	0.75	0.27	0.79	0.81	0.74	0.66
LightGBM	0.79	0.81	0.78	0.23	0.84	0.78	0.81	0.78
RoboMal [20]	0.85	0.87	0.87	0.17	0.91	0.82	0.8	0.71
RoboCop (PE)	**0.87**	**0.89**	**0.87**	**0.24**	**0.85**	**0.91**	**0.89**	**0.83**
RoboCop (PE+OM)	**0.94**	**0.95**	**0.93**	**0.02**	**0.93**	**0.92**	**0.97**	**0.93**

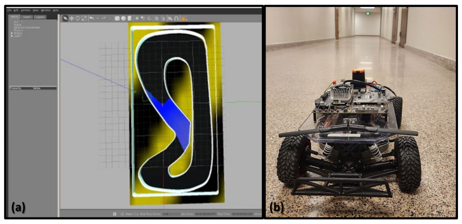

Fig. 3. (a) The simulator and (b) robot vehicle used as a sim-to-real validation pipeline for the ROBOCOP framework.

models, and investigate the impact of online monitoring on robot performance.

A. Data Collection and Implementation

The effective evaluation of the ROBOCOP framework requires testing it against a wide array of attacks that span the entire spectrum of potential cyber-physical threats. To achieve this, we use the RoboMal dataset [20], a comprehensive collection of 452 benign and malicious binary executables designed to simulate cyber-physical attacks on robotic systems. This dataset, uniquely focused on robotic malware, includes examples of sensor, actuator, and controller logic attacks, making it an ideal basis for evaluating ROBOCOP's detection algorithms. Notably, RoboMal is a publicly available dataset specifically designed for cyber-physical and malware attacks targeting robots, built upon the F1-tenth robot car code base. The dataset provides binaries and configurations for both benign and attack scenarios, which we replicated using a simulator, as illustrated in Fig. 3(a), for a wall-following task along a track. We augmented the dataset by replicating attacks through generated code files and established a simulator-to-real robot evaluation pipeline using a robot vehicle, as shown in Fig. 3(b), to enable physical robot validation.

In our simulations, we executed a range of cyber-physical attacks including sensor, velocity, steering, and controller manipulations. During sensor attacks, we intentionally introduced a random bias in the input code for sensor data, distorting the sensor's output from its true measurements. For controller attacks, we altered parameters such as limits, gain values, and processing steps to compromise the controller logic. Actuator attacks were simulated by modifying the parameters that influence velocity and steering, such as actuator limits and scaling factors, as well as the equations that govern their operations. These attacks are visualized in Fig. 4,

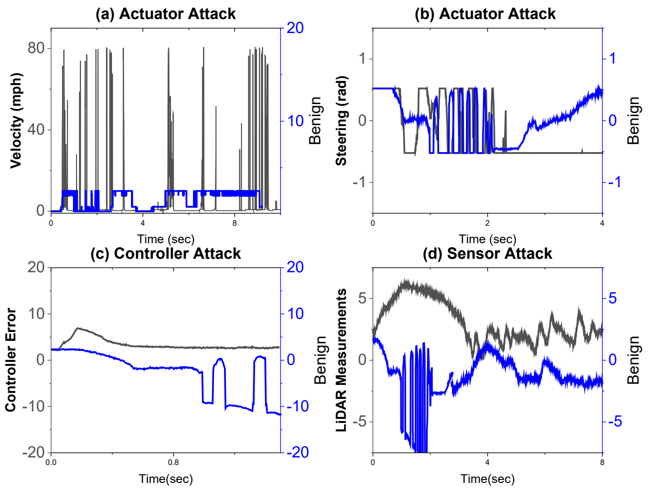

Fig. 4. The (a) velocity, (b) steering, (c) controller, and (d) sensor attacks with their respective benign behaviors as detected by ROBOCOP.

where each scenario demonstrates significant deviations in the robot behavior under attack compared to its standard operation. It is critical to emphasize that, while various modifications can impact the robot's performance, not all lead to outright malicious behavior. Therefore, an accurate annotation of the dataset is crucial to distinguish benign anomalies from genuine attacks. This distinction enables the ROBOCOP framework to effectively differentiate between mere performance degradation and actual malicious activities.

B. Experiment: Zero-Day Attack Detection Performance

The ROBOCOP framework was evaluated under two conditions: globalized and zero-day. Globalized performance evaluates the overall effectiveness of the model in classifying malicious and benign code, in which a stratified mix of all classes of attacks is used in the training. On the other hand, zero-day performance takes a one-vs.-rest approach, wherein one class is completely removed from the dataset and is used in the test set only. Therefore, the model learns nothing about that class from the training stage and must make the decision based on the latent space mapping by deriving correlations. This creates true zero-day conditions. In both cases, the model predicts a benign or malicious class for each input sample.

The model was trained and tested using a 10-fold cross-evaluation while training for 200 epochs. Hyperparameters,

TABLE II

THE COMPARISON OF SCOPE OF ATTACK DETECTION IN ROBOT CPS

Threats	[27]	[18]	[14]	Ours
Sensor Spoofing	✓	✓	✓	✓
Control Parameter	✓	✗	✗	✓
Control Signal Spoofing	✓	✓	✗	✓
Actuator Spoofing	✓	✓	✓	✓
Zero-Day	✗	✗	✗	✓

such as kernel size, dense neurons, and Mahalanobis threshold, were determined using a validation set and fine-tuning. We use the Adam optimizer with a learning rate of 1e-4. After fine-tuning, the kernel sizes of $[3, 5, 8]$ with two max clusters were set as optimal. The Mahalanobis threshold varies between 0.3 and 0.7 for the latent space.

For the agent, training was carried out for 200 epochs, with each having 100 episodes on the F1-tenth simulator with the Porto track as the setting environment. We use a 3-layer MLP for the DQN with 150, 75, and 25 layers. The layers use ReLU activation. Probability thresholds are empirically learned at 0.4 and 0.7, respectively.

For comparison, we chose other established models that have been studied for malware detection and have been evaluated on the RoboMal dataset, particularly. The results of the test are tabulated in Table I. Although globalized performance covers metrics such as accuracy and false negative rate (FNR), the zero-day evaluation shows the accuracy of detection.

The results illustrate the strength of the RoboCop zero-day detection framework to identify known classes and zero-day attacks. Latent space mapping adds to the overall performance of the model and allows it to showcase its strength even on the unseen set of classes. The online monitoring stage further boosts the effectiveness of the framework as is observed by the improvement in the overall performance of the system from just using pre-execution.

Scope of Attacks. Comparing the scope of RoboCop with other models of cyber and physical attack detection, as shown in Table II, we notice that only RoboCop can detect a wide family of attacks that originate in either the cyber realm or the physical realm due to its coupled detection strategy. Most of the other models are specific to sensor attacks or tuned for run-time behavior monitoring only. RoboCop is the only framework among these that can prevent an attacker from gaining control of the robot and detecting zero-day attacks. Moreover, the RoboCop framework can be easily extended to cover purely physical attacks that rely on physical interactions, such as placing adversarial signals to obfuscate sensor measurements.

C. Ablation Study: The Impact of Online Monitoring

We further investigate the effectiveness of online monitoring in detecting cyber-physical attacks. To this end, a set of completely unseen attacks were deployed onto the software that the pre-execution stage could not identify. The evaluation was carried out in both simulations and on a physical robot for 20 crafted attacks. In the simulation, the robot vehicle takes about 48 seconds to cover one loop around the track. The physical robot vehicle navigates a hallway in 4 minutes, as shown in Fig. 5. Without RoboCop, the compromised controller causes the robot to crash into the track wall in 17 seconds. However, RoboCop stops the robot in 7 seconds to prevent such a crash. In 18 of 20 cases, the model stops the vehicle in less than 25% of the total execution time. Only one crash is reported, and in one case the robot vehicle stops in 30% of the execution time. Furthermore, we note that the overall computational overhead of the online verification system is only 12% during runtime, while the framework takes about 20MB of space after deployment through the TinyML pipeline [30], [31].

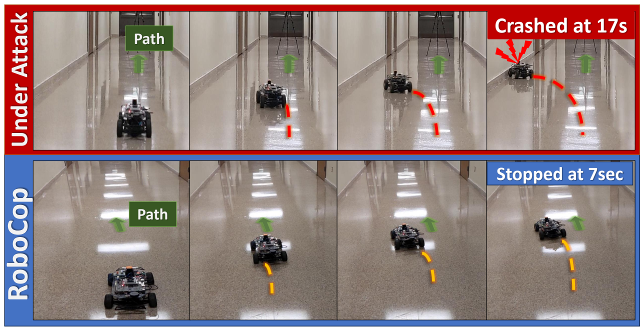

Fig. 5. Without RoboCop, a compromised robot crashes while navigating a hallway. However, with RoboCop in place, the robot is safely halted before any collision occurs.

V. Conclusion and Discussion

RoboCop represents an advancement in security for robotic cyber-physical systems (CPS), effectively addressing the critical challenge of zero-day cyber-physical attack detection. Its strength lies in its two-tiered defense architecture. The pre-execution verification leverages zero-shot learning to identify and neutralize unseen attacks by analyzing static software features. This proactive approach is complemented by a sophisticated online monitoring mechanism that employs reinforcement learning to provide continuous and dynamic protection during robot operation. The combination of these components enables RoboCop to offer a robust defense against the complex landscape of cyber-physical threats.

Our experimental validation in both simulation and real-world deployment on the F1-tenth autonomous car platform demonstrates RoboCop's capability to detect and mitigate a wide range of cyber-physical threats. The system achieves an average zero-day detection accuracy of 93% and an overall accuracy of 94%. Moreover, RoboCop operates efficiently during active robot deployments, preventing attacks in 7 seconds, with only a 12% increase in computational overhead.

The implications of RoboCop extend beyond its immediate technical contributions. By strengthening the security of robotic CPS against zero-day threats, RoboCop improves the trust and adoption of these technologies. In future work, we aim to improve RoboCop's capabilities to be robot and controller-agnostic using physics-based methods.

References

[1] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixao, F. Mutz *et al.*, "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, p. 113816, 2021.

[2] E. Martinez-Martin and A. P. del Pobil, "Personal robot assistants for elderly care: an overview," *Personal assistants: Emerging computational technologies*, pp. 77–91, 2018.

[3] M. D. Simoni, E. Kutanoglu, and C. G. Claudel, "Optimization and analysis of a robot-assisted last mile delivery system," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102049, 2020.

[4] H. M. Rouzbahani, H. Karimipour, A. Rahimnejad, A. Dehghantanha, and G. Srivastava, "Anomaly detection in cyber-physical systems using machine learning," in *Handbook of Big Data Privacy*. Springer, 2020, pp. 219–235.

[5] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno, and H. J. Chizeck, "To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots," *arXiv preprint arXiv:1504.04339*, 2015.

[6] B. Ly and R. Ly, "Cybersecurity in unmanned aerial vehicles (uavs)," *Journal of cyber security technology*, vol. 5, no. 2, pp. 120–137, 2021.

[7] H. Kim, M. O. Ozmen, A. Bianchi, Z. B. Celik, and D. Xu, "Pgfuzz: Policy-guided fuzzing for robotic vehicles." in *Network and Distributed System Security (NDSS) Symposium*, 2021.

[8] H. Kim, M. O. Ozmen, Z. B. Celik, A. Bianchi, and D. Xu, "Pgpatch: Policy-guided logic bug patching for robotic vehicles," in *IEEE Symposium on Security and Privacy (SP)*, 2022.

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

[10] G. Loukas, *Cyber-physical attacks: A growing invisible threat*. Butterworth-Heinemann, 2015.

[11] A. Bezemskij, G. Loukas, R. J. Anthony, and D. Gan, "Behaviour-based anomaly detection of cyber-physical attacks on a robotic vehicle," in *2016 15th international conference on ubiquitous computing and communications and 2016 international symposium on cyberspace and security (IUCC-CSS)*. IEEE, 2016, pp. 61–68.

[12] P. O'Neill, "Hackers can trick a tesla into accelerating by 50 miles per hour," 2020.

[13] K. Chung, X. Li, P. Tang, Z. Zhu, Z. T. Kalbarczyk, R. K. Iyer, and T. Kesavadas, "Smart malware that uses leaked control data of robotic applications: The case of raven-ii surgical robots," in *22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID) 2019*, 2019, pp. 337–351.

[14] S. Etigowni, S. Hossain-McKenzie, M. Kazerooni, K. Davis, and S. Zonouz, "Crystal (ball): I look at physics and predict control flow! just-ahead-of-time controller recovery," in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 553–565. [Online]. Available: https://doi.org/10.1145/3274694.3274724

[15] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Out of control: stealthy attacks against robotic vehicles protected by control-based techniques," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 660–672.

[16] P. Dash, G. Li, Z. Chen, M. Karimibiuki, and K. Pattabiraman, "Pid-piper: Recovering robotic vehicles from physical attacks," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2021, pp. 26–38.

[17] P. Dash, M. Karimibiuki, and K. Pattabiraman, "Stealthy attacks against robotic vehicles protected by control-based intrusion detection techniques," *Digital Threats: Research and Practice*, vol. 2, no. 1, pp. 1–25, 2021.

[18] 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*, 2018, pp. 801–816.

[19] T. Bonaci and H. J. Chizeck, "On potential security threats against rescue robotic systems," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2012, pp. 1–2.

[20] U. Kaur, H. Zhou, X. Shen, B.-C. Min, and R. M. Voyles, "Robomal: Malware detection for robot network systems," in *IEEE International Conference on Robotic Computing (IRC)*, 2021, pp. 65–72.

[21] U. Kaur, Z. B. Celik, and R. M. Voyles, "Robust and energy efficient malware detection for robotic cyber-physical systems," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2022, pp. 314–315.

[22] J.-P. A. Yaacoub, H. N. Noura, O. Salman, and A. Chehab, "Robotics cyber security: Vulnerabilities, attacks, countermeasures, and recommendations," *International Journal of Information Security*, vol. 21, no. 1, pp. 115–158, 2022.

[23] B. Dieber, R. White, S. Taurer, B. Breiling, G. Caiazza, H. Christensen, and A. Cortesi, "Penetration testing ros," *Robot Operating System (ROS) The Complete Reference (Volume 4)*, pp. 183–225, 2020.

[24] S. Han, M. Xie, H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: Techniques and challenges," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1052–1062, 2014.

[25] H. Kim, R. Bandyopadhyay, M. O. Ozmen, Z. B. Celik, A. Bianchi, Y. Kim, and D. Xu, "A systematic study of physical sensor attack hardness," in *IEEE Symposium on Security and Privacy (S&P)*, 2024, pp. 143–143.

[26] A. Ding, M. Chan, A. Hassanzadeh, N. O. Tippenhauer, S. Ma, and S. Zonouz, "Get your cyber-physical tests done! data-driven vulnerability assessment of robotic vehicle," in *International Conference on Dependable Systems and Networks (DSN)*, 2023.

[27] A. Ding, P. Murthy, L. Garcia, P. Sun, M. Chan, and S. Zonouz, "Mini-me, you complete me! data-driven drone security via dnn-based approximate computing," in *International Symposium on Research in Attacks, Intrusions and Defenses*, 2021, pp. 428–441.

[28] G. Panice, S. Luongo, G. Gigante, D. Pascarella, C. Di Benedetto, A. Vozella, and A. Pescapè, "A svm-based detection approach for gps spoofing attacks to uav," in *International Conference on Automation and Computing (ICAC)*, 2017, pp. 1–11.

[29] M. O'Kelly, V. Sukhil, H. Abbas, J. Harkins, C. Kao, Y. V. Pant, R. Mangharam, D. Agarwal, M. Behl, P. Burgio *et al.*, "F1/10: An open-source autonomous cyber-physical platform," *arXiv preprint arXiv:1901.08567*, 2019.

[30] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov *et al.*, "Benchmarking tinyml systems: Challenges and direction," *arXiv preprint arXiv:2003.04821*, 2020.

[31] P. Warden and D. Situnayake, *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.