Simulation-Based Digital Twins for Robotic Cybersecurity Testing with AI-Driven Adaptive Defence

Gloria Landsberg-Stoilova
Department of Management and Business Information Systems,
Faculty of Management
Technical University of Sofia
Sofia, Bulgaria
gloria.stoilova@gmail.com

Roumiana Ilieva
Department of Management and Business Information Systems,
Faculty of Management
Technical University of Sofia
Sofia, Bulgaria
rilieva@tu-sofia.bg

***Abstract*—Robotic and mechatronic systems are increasingly connected via open communication protocols and cloud services, exposing safety-critical operations to cyber threats such as spoofing, denial-of-service, and data poisoning [4], [5]. Testing security measures directly on physical robots is costly, disruptive, and often unsafe. This paper proposes a simulation-based digital twin (DT) framework that faithfully replicates a robotic system's kinematics, dynamics, and network behaviour in a virtual environment, enabling risk-free cybersecurity experimentation. Within this DT, an artificial intelligence (AI)–driven adaptive defence module combines traffic-based anomaly detection with reinforcement learning agents that learn to respond to evolving attacks by reconfiguring control or communication policies. A conceptual proof-of-concept testbed is presented using a simulated ROS 2 2-controlled manipulator subjected to synthetic network intrusions. Preliminary simulation results indicate that the adaptive AI defence detects malicious traffic earlier and restores normal operation faster than a static rule-based intrusion detection system. The study highlights digital twins as a practical, low-risk platform for developing and validating next-generation cybersecurity strategies for robotic systems.**

Keywords—AI-Augmented Digital Twins, Robotic Cybersecurity, Resilient Robotics, Real-Time Simulation Intelligent Threat Mitigation, Adaptive System Defence, Simulation-Driven Security.

I. INTRODUCTION

Robotic and mechatronic systems move rapidly from isolated, pre-programmed machines toward networked, adaptive cyber-physical systems [9]. Modern industrial manipulators, collaborative robots (cobots), autonomous mobile robots (AMRs), and smart production cells routinely connect to enterprise networks, cloud platforms, and remote maintenance services. While this connectivity enables flexible production and real-time monitoring, it also broadens the attack surface [2]. Network intrusion, spoofed sensor data, denial-of-service (DoS), and machine-learning model poisoning have already been demonstrated against robotic middleware such as the Robot Operating System (ROS/ROS 2) [4]; [5].

Traditional approaches to robot cybersecurity testing rely on laboratory hardware setups or penetration tests performed on physical robots. These methods face several barriers:
Safety and cost: Aggressive attack simulation can damage equipment or stop production.

Limited repeatability: Physical experiments are time-consuming and sensitive to hardware differences [11].
Restricted access: Researchers often lack specialized robot hardware or cannot change production systems for security testing.

Consequently, many proposed security solutions for robotics remain conceptual or poorly validated.
Digital twins (DTs) — high-fidelity virtual replicas of physical systems — have emerged as robust tools for design, monitoring, and predictive maintenance. However, their potential for cybersecurity research in robotics is only starting to be explored [7]; [14]. A DT can integrate accurate physics simulation with network and software emulation to create a safe, controllable environment for injecting cyber-physical threats and evaluating defences before deployment.

Parallel to this, artificial intelligence (AI) — especially machine learning–based anomaly detection and reinforcement learning (RL) — has shown promise for adaptive cyber defence. Unlike static, rule-based intrusion detection systems (IDS), AI can learn to recognize evolving attack patterns and dynamically adjust control or communication strategies to maintain safe operation [6]; [15].

This paper proposes a simulation-based DT framework for robotic cybersecurity testing and AI-driven adaptive defence.
The key contributions are:
Digital twin architecture for secure robotics: a virtual testbed that models robotic kinematics/dynamics and network behaviour to support safe cyber-attack experimentation.
AI-driven adaptive defence module: integration of anomaly detection and RL-based mitigation policies trained entirely in simulation.

Proof-of-concept evaluation: a conceptual study using a ROS 2 2-controlled manipulator, demonstrating how AI-enabled defence can reduce detection latency and recovery time compared with static IDS approaches.
This work is intended for researchers and practitioners who lack access to physical robots but need a risk-free environment to design and validate security mechanisms for robotic and mechatronic systems.

ACKNOWLEDGE THE FINANCIAL SUPPORT OF THE PROJECT WITH FINANCING AGREEMENT NO. PVU-44 OF 05.12.2024 UNDER PROJECT NO. BG-RRP-2.017-0011 "ECOLOGICAL COLLABORATIVE ROBOTS POWERED BY GREEN HYDROGEN" UNDER THE RECOVERY AND RESILIENCE MECHANISM FOR THE IMPLEMENTATION OF AN INVESTMENT UNDER C2I2 "INCREASING THE INNOVATION CAPACITY OF THE BULGARIAN ACADEMY OF SCIENCES (BAS) IN THE FIELD OF GREEN AND DIGITAL TECHNOLOGIES" FROM THE RECOVERY AND RESILIENCE PLAN, BULGARIA.

II. Background and Related Work

A. Digital Twins in Robotics and Mechatronics

The **Digital Twin (DT)** concept originated in aerospace and manufacturing, where high-fidelity virtual models are continuously synchronised with their physical counterparts. DTs have become key enablers for **simulation-driven design, predictive maintenance, and system optimisation in robotics** [3].

A typical robotic DT integrates three layers:

1. **Physical Layer** – the actual robot and its environment;
2. **Virtual Layer** – a physics-based simulation replicating kinematics, dynamics, and sensor feedback;
3. **Data Synchronisation Layer** – bidirectional data exchange that ensures the virtual model mirrors the physical state in near real time.

Modern simulation tools such as **Gazebo, Webots, CoppeliaSim, and Unity Robotics** support the creation of DTs that accurately mirror mechanical and control behaviour. Recent work extends DTs with **network emulation and cloud connectivity**, allowing researchers to study latency, packet loss, or system integration across Industrial IoT architectures [12]. However, DTs are rarely applied beyond performance optimisation or maintenance. Their potential as **cybersecurity sandboxes**—where virtual replicas are used to simulate and analyse cyber-physical threats without risking hardware—remains underexplored [7].

B. Cybersecurity Challenges in Robotic Systems

Networked robots operate as **cyber-physical systems (CPS)**, integrating embedded controllers, sensors, actuators, and network interfaces [1]. This tight coupling introduces vulnerabilities across multiple layers:

- **Communication Layer:** protocols such as ROS 1/ROS 2, OPC UA, and MQTT were designed for openness rather than confidentiality [4]. Research shows robots can be compromised via spoofed commands, replayed sensor data, and adversarial perception attacks [10]; [13].
- **Control Layer:** unauthorised modification of parameters or firmware manipulation can degrade precision or cause unsafe motion.
- Perception **Layer:** adversarial inputs to camera or LIDAR sensors can mislead AI-based perception modules.

Existing defensive measures are largely **static**—network firewalls, signature-based intrusion detection, or cryptographic authentication (García et al., 2014). These solutions struggle against **dynamic, adaptive attacks** in heterogeneous environments typical of Industry 4.0 robotic cells. Moreover, testing such defences on operational robots is impractical due to safety and downtime constraints, highlighting the need for **simulation-based security validation**.

C. AI-Driven Adaptive Defence Mechanisms

Artificial intelligence techniques are increasingly being investigated to enhance cyber resilience. **Machine-learning-based intrusion detection systems (ML-IDS)** can identify abnormal traffic patterns using statistical, supervised, or unsupervised models such as support vector machines, autoencoders, and graph neural networks [6].

Beyond detection, **reinforcement learning (RL)** enables systems to select mitigation actions autonomously—e.g., rerouting communication, isolating components, or adjusting control parameters—to maintain safe operation during an attack [15].

Recent studies in industrial networks and autonomous vehicles show that **AI-enabled agents can outperform static rules** by learning temporal patterns of attacks and responding in real time [8]. Nevertheless, most of these approaches are evaluated using simplified network traces or theoretical models rather than in realistic robotic control loops.

D. Research Gap and Motivation

While DTs and AI have been widely explored individually, **their integration for robotic cybersecurity remains limited** [7]. No standardised, simulation-based frameworks currently exist for training and validating AI-adaptive cybersecurity mechanisms on robotic control loops. Current literature seldom combines a **virtual robotic twin** with **AI-adaptive defence agents** that can learn and test mitigation strategies entirely in simulation.

Bridging this gap can provide:

- A **risk-free environment** for attack injection and defence training;
- Quantitative insights into detection latency, false-positive rates, and control-loop stability under attack;
- A reusable framework for researchers without physical hardware access.

E. Conceptual Integration and Research Evolution

To consolidate the insights from the reviewed literature, Figure 1 illustrates the conceptual convergence among Digital Twins (DT), Artificial Intelligence (AI), and Cybersecurity (CS) within the robotics domain.

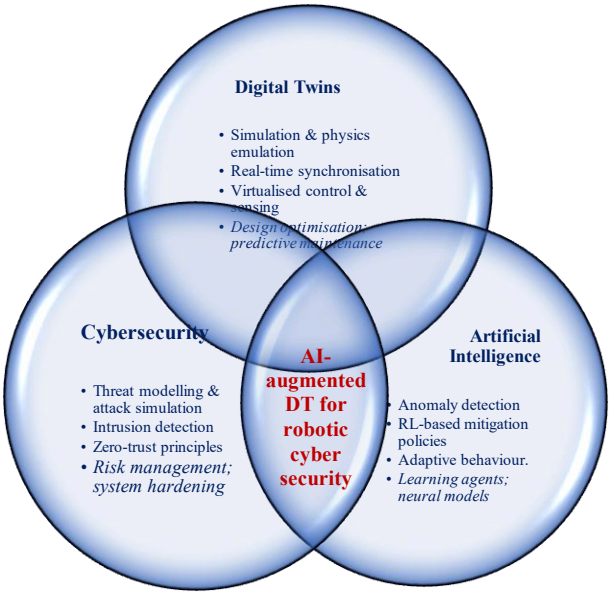

Fig. 1. Conceptual convergence of DT, AI, and SC domains.

DT provides the simulation backbone for virtual experimentation, AI contributes learning and adaptive decision-making capabilities, and CS ensures data integrity, confidentiality, and system resilience.

The overlapping area among these three domains defines the research focus of this study – the development of an AI-Augmented DT for Robotic Cybersecurity. The intersection defines the proposed research focus: AI-Augmented DT for Robotics Cybersecurity.

Building on this conceptual mapping, Figure 2 illustrates the evolutionary trajectory of robotic systems from traditional, isolated industrial machines toward intelligent, networked, and cyber-physical systems.

Each stage represents a significant milestone in the digital transformation of robotics—from deterministic control logic to cloud-connected architectures, and finally to simulation-driven, AI-enhanced defensive ecosystems. This timeline underscores that AI-enabled DT Security is not an isolated concept but the next logical phase in the technological evolution of robotics.

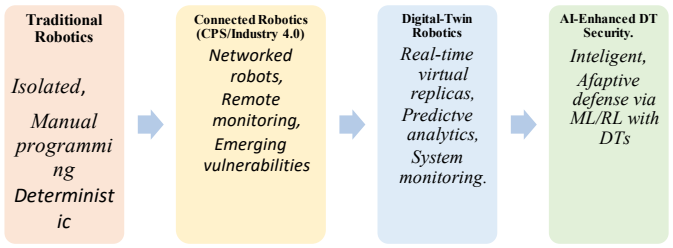

Fig. 2. Evolution of robotics from isolated industrial units towards AI-enhanced DT security environments.

The timeline highlights the transition from deterministic control to adaptive, learning-based cyber-physical defence.

These two figures illustrate the theoretical convergence and technological evolution that motivate the framework proposed in this paper. While Figure 1 defines the study's interdisciplinary foundation, Figure 2 positions the research within the ongoing digital transformation of robotics, establishing a clear context for the Proposed Digital Twin Security Testbed described in the following section.

III. Proposed Digital Twin Security Testbed

This section describes the design and internal mechanics of the **Digital Twin Security Testbed (DTST)** used throughout this study. The DTST is a fully simulated, modular environment that enables rapid, reproducible experimentation with cyber-physical attacks and AI-driven mitigation strategies for robotic systems. The design follows five core components: (1) virtual robot model, (2) network emulation, (3) threat injection engine, (4) AI defence module, and (5) synchronization & simulation loop. These components operate in a closed feedback cycle that supports iterative learning and quantitative evaluation.

1) Virtual robot model (physics & control)

The DTST's heart is a **virtual robot model** that provides realistic kinematic and dynamic behaviour without requiring physical hardware. Our reference setup uses a 6-DoF industrial manipulator model implemented in a physics simulator (e.g., Gazebo or ODE within Gazebo / Webots). The simulator exposes:

2) Joint-level states (position, velocity, torque) and sensor streams (encoders, force-torque, IMU, camera/LIDAR placeholders).

3) A control interface representing a typical PLC/robot controller loop (PID/trajectory follower) that subscribes to command topics and publishes telemetry.

4) Pluggable model fidelity (collision geometry, friction, sensor noise) so experiments can trade realism for computational efficiency.

The virtual robot provides the baseline "normal" behaviour used to train anomaly detectors and to measure the impact of attacks on motion accuracy, task completion, and safety-related metrics.

B. Network emulation: ROS 2 traffic, delay/jitter injection, replay

To model real-world cyber conditions, the DTST includes a **network and communication emulator** tightly coupled with the virtual robot:

- Middleware emulation: ROS 2 is the primary

  Middleware replica, including publishers/subscribers, quality of service settings, and topic namespaces typical for manipulator controllers and perception stacks.

- Network **impairments:** A configurable emulator layer injects latency, jitter, packet loss, and bandwidth constraints into message flows. Parameters are adjustable per topic to simulate congested Ethernet, wireless links, or intermittent corporate VPNs.

- **Trace** replay**:** recorded ROS 2 traffic traces

  (benign and malicious) can be replayed deterministically for benchmarking. This supports reproducibility and the construction of standardized test suites.

This emulation enables the study of how timing and message-reliability deviations affect closed-loop control and AI perception modules.

C. Threat Injection Engine (DoS, spoofing, data poisoning)

The **Threat Injection Engine (TIE)** is the adversarial controller that generates controlled attack vectors. It supports three classes of attacks relevant to robotic CPS:

1) ***Denial-of-Service (DoS) / Flooding:*** rapid message bursts targeting specific topics (e.g., /joint_cmd) or nodes (controller) to saturate throughput and increase latency—parameters: packets per second, duration, target topic list.

2) ***Spoofing / Replay:*** forging or replaying ROS 2 messages to substitute control commands or sensor readings. Modes include immediate injection, delayed replay, and gradual drift insertion—parameters: injection timing, amplitude, and topic filter.

3) ***Data Poisoning:*** progressive modification of sensor streams (e.g., adding bias to encoder readings or corrupting vision inputs) intended to mislead learning-based perception or degrade controller performance—parameters: poison rate, distribution, entry point.

Each scenario is parameterizable and can be orchestrated in campaigns (sequence of attacks with escalation), enabling stress-testing of detection and mitigation strategies. All injected events are logged with ground-truth labels for supervised evaluation.

D. AI defence module: anomaly detection + RL mitigation policy

The AI defence stack is divided into two cooperating subcomponents—real-time anomaly detection and a reinforcement learning (RL) mitigation policy—supported by analytics for evaluation.

1) Anomaly detection (online monitoring)

a) ***Input features:*** topic-level message rates, inter-arrival times, payload statistics (e.g., mean/variance of joint positions), control error residuals (difference between commanded and observed states), and lightweight derived features (spectral, time-windowed aggregates).

b) ***Model family:*** unsupervised models are preferred for zero-day detection, including sequence autoencoders, isolation forests, and, where topology matters, graph-based detectors on the ROS node-topic graph.

c) ***Decision logic:*** anomaly score → thresholding → alert. Thresholds are calibrated using baseline simulation runs to balance detection latency and the false-positive rate. Alerts are timestamped and routed to the RL agent input.

E. Reinforcement learning mitigation (policy layer)

a) ***State definition:*** concatenation of anomaly flags, current network Quality of Service metrics, recent control errors, and a short history of system telemetry (sliding window).

b) ***Action space:*** a discrete set of mitigations, Examples: (a) temporarily throttle or drop non-critical topics, (b) isolate suspect node(s) from the bus, (c) switch to a degraded safe controller mode (lower speed/torque limits), (d) trigger sensor cross-validation routines, (e) initiate topic re-authentication handshake.

c) ***Reward function:*** *multi-objective — negative* penalties for unsafe behaviour (significant control error, collision risk), penalties for unnecessary interventions (avoiding overreaction), positive reward for task completion, and quick recovery to baseline performance. Reward shaping encourages conservative but effective interventions.

d) ***Training regime:*** *RL is trained entirely within* the DTST using episodic simulations containing randomized attacks. Transfer learning/domain randomization techniques are applied to improve robustness across variations in simulation parameters.

F. Operational constraints

1) Mitigation actions must adhere to safety invariants (e.g., never issue commands that exceed joint limits). A safety supervisor enforces hard constraints to prevent learned policies from producing unsafe control commands even in simulation.

2) Synchronization & simulation loop (data consistency & reproducibility)

The DTST enforces strict synchronization to preserve temporal coherence across modules:

3) Global simulation clock: a central time server (simulated wall clock) coordinates physics ticks, message timestamps, and attack injection events. This ensures that telemetry, anomaly detection inputs, and RL actions are aligned.

4) ***Deterministic logging:*** All regular and injected messages are recorded with high-resolution timestamps and ground-truth labels. Logs enable deterministic replay for benchmarking and for offline ML training.

5) Simulation loop cadence: The system supports variable tick rates (e.g., 100–1000 Hz for physics; 10–100 Hz for middleware), configurable per experiment. Rate limits are documented to clarify real-time vs. accelerated training modes.

6) Evaluation metrics and experimental protocol (summary)

To quantify defence performance, the DTST reports standardized metrics per run:

a) ***Detection latency:*** time from attack start to anomaly flag.

b) ***Detection accuracy*****:** true/false positive rates on labelled logs.

c) ***Recovery time*****:** time until key performance indicators (KPIs) return within baseline bounds (e.g., tracking error below threshold).

d) ***Operational impact:*** task completion rate, additional downtime introduced by mitigation, and computational overhead (CPU/latency of defence stack).

Experiments follow a rigorously documented protocol: (a) baseline calibration runs (no attack), (b) single-attack experiments, (c) multi-attack combined scenarios, and (d) randomized campaign stress tests. Each experiment is repeated deterministically with seeded randomness to produce statistically meaningful results

7) Assumptions and limitations

a) Simulation fidelity vs. reality gap: while physics and network emulation are high-fidelity, but the absence of hardware introduces sim2real risk; we mitigate this by domain-randomizing simulation parameters.

*b) Scope***:** This DTST focuses on the middleware-level and sensor/command attacks, and does not model low-level firmware exploits or hardware fault modalities.

c) Safety fallbacks: because this study is simulation-only, safety logic is enforced as hard checks; deploying learned policies to physical robots would require an independent safety validation step.

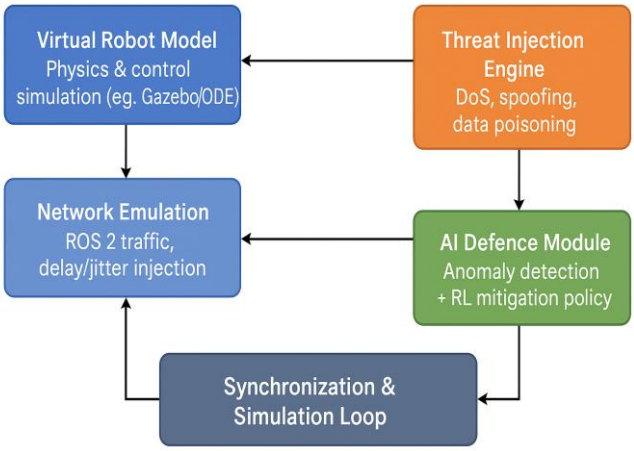

Fig. 3. Visualize the five components in a closed loop: Virtual Robot Model ↔ Network Emulator ↔ Threat Injection Engine → AI Defence Module → Synchronization & Replay / Logging, with arrows indicating data flow and control.)

IV. Simulation Experiment Design (Conceptual)

This section describes the experimental protocol for evaluating the Digital Twin Security Testbed (DTST). The goal is to produce reproducible, statistically meaningful comparisons between static detection baselines and the proposed AI-adaptive defence under controlled, parameterised attack campaigns. The design is intentionally simulation-centric: all experiments are executed within the DTST (Section 3) using a virtual manipulator model, ROS 2 traffic emulation, and the threat-injection engine.

A. Attack scenarios

We model three (and one combined) canonical attack classes targeted at robot middleware and sensor channels. Each attack class is parameterised to enable graded stress tests and campaign compositions. Each scenario is fully logged with ground-truth labels, enabling quantitative evaluation of detection and mitigation performance (Table I).

TABLE I. Parameterised attack scenarios used in the DTST experiments.

Attack class	Target	Mode/ Description	Example parameters (tunable)	Intended effect on DTSD	Logging & Ground-truth
Denial-of-Service (DoS) / Flooding	Control topics (e.g./ joint_cm), status topics, controller nodes, network interface	High-rate message bursts aimed at saturating bandwidth or queues; may be broad (network-wide) or targeted (specific topic/node)	Packet-rate multiplier (×), duration (s), target topic list, start time, burst pattern	Increased latency/jitter, missed deadlines, control jitter, possible missed commands leading to tracking error or task failure	Timestamped attack start/stop; targeted topics; packet counts; ground-truth label = DoS
Spoofing / Replay	Topic payloads (commands, sensor messages), ROS 2 topics, node outputs.	Injection of forged messages or replay of recorded messages (immediate, delayed, or time-shifted) to mislead controller or perception.	Injection amplitude (payload deviation), injection frequency, delay offset (s), entry-point node, replay trace ID.	Controller executes forged commands or acted on stale/duplicated sensor inputs; wrong trajectories or unsafe behavior.	Record injected message content, orig vs injected timestamps; ground-truth label = Spoof/Replay
Data Poisoning	Sensor streams (encoder, IMU, camera placeholders), perception inputs, telemetry used by learning modules	Gradual or abrupt corruption of sensor/data used by perception or control; may bias, spike or flip labels used by ML.	Poison rate (% messages), noise distribution (Gaussian bias, spike), onset schedule (gradual vs abrupt), affected sensor(s)	Degraded perception accuracy, drifting state estimates, ML misclassifications, increased control errors over time	Logged corrupted stream with original vs poisoned values; ground-truth label = Poison

TABLE II. Combined attacks and randomized campaigns

Atack class	Target	Mode/ Description	Example parameters (tunable)	Intended effect on DTSD	Logging & Ground-truth
Combined / Campaigns	Any combination of above; multi-stage campaigns	Sequences or stochastic mixes of attacks (e.g., DoS then spoofing), escalation patterns or randomized campaigns for stress testing	Campaign script (ordered attacks), random seed, escalation schedule, inter-attack interval	Test robustness to multi-vector attacks; examines cumulative impact & defense generalization	Campaign script + per-event labeling; global campaign ID; timestamps for each sub-attack

B. Evaluation metrics

We adopt standard detection and system-level metrics and introduce control-centric KPIs relevant for robotics. Table 3 summarises the **evaluation metrics** adopted to assess the performance of the AI-driven adaptive defence within the Digital Twin Security Testbed.

[1] **Campaigns:** Single-attack runs, combined attacks (e.g., DoS + spoofing), and randomized campaigns (stochastic combinations with seeded randomness). Each configured scenario is logged with ground-truth labels for offline evaluation.

The metrics are grouped into three categories—detection quality, resilience and mitigation, and system feasibility—covering the framework's security and operational dimensions.

Detection metrics (e.g., Detection Latency, Precision, Recall, F^1) quantify how quickly and accurately the system recognises malicious activity. Resilience metrics (e.g., Recovery Time, Mitigation Effectiveness, Operational Impact) capture how efficiently the defence restores normal operations and the level of disruption it introduces.

Finally, system-level metrics (Decision Latency and CPU Overhead) assess the computational feasibility of real-time deployment.

Each metric is formally defined within the digital twin simulation environment, computed automatically after each run, and reported as the mean ± standard deviation across multiple repetitions. All results are cross-referenced using the unique Scenario ID, Campaign ID, and Config Hash to guarantee traceability between the evidence logs and experimental artefacts.

TABLE III. SUMMARY OF DETECTION, RESILIENCE, AND SYSTEM-LEVEL METRICS FOR EVALUATING THE PROPOSED AI-ADAPTIVE DIGITAL-TWIN DEFENCE

Detection metrics	
Detection latency (T_detect)	time from attack start to first valid alert.
Detection accuracy	precision, recall, and F1-score computed on timestamped labels.
False positive rate (FPR)	fraction of alerts during benign operation.
Mitigation & resilience metrics	
Recovery time (T_recover)	time from mitigation action to restoration of a pre-defined baseline KPI (e.g., tracking error < ε).
Mitigation effectiveness	percentage reduction in KPI degradation (e.g., RMS tracking error) compared to unmitigated runs.
Operational impact	overhead induced by mitigation (downtime, performance loss during safe-mode).
System & computational metrics	
Task completion rate	fraction of episodes where the manipulator completes the assigned task within constraints.
CPU / latency overhead	Detection and RL modules introduce additional compute and decision latency (necessary for real-time feasibility).

Together, these metrics provide a **balanced, reproducible performance profile that quantifies** the detection capability of the AI modules and the **stability, recovery behaviour, and resource cost** of the integrated robotic-cybersecurity framework.

C. Baselines and ablation studies

To ensure the reliability and interpretability of the experimental results, the proposed AI-adaptive defence is evaluated against multiple baseline configurations and ablation variants.

The baselines establish performance benchmarks for detection accuracy, resilience, and operational impact under non-adaptive learning. Ablation studies, in turn, isolate the contribution of individual components—particularly the reinforcement learning (RL) agent and its policy structure—allowing a more precise understanding of each module's marginal benefit.

TABLE IV. BASELINE AND ABLATION CONFIGURATION SUMMARY

Configuration	Purpose	Key Metric Evaluated
No defence (control)	Establish lower-bound performance with no detection or mitigation applied.	Task completion rate, recovery time.
Static IDS	Compare adaptive AI against fixed rule-based defence.	Detection accuracy (Precision / Recall / F1).
Anomaly detector only	Evaluate detection capability without automated mitigation.	Detection latency, false positive rate.
RL removed (ablation)	Quantify contribution of reinforcement learning component.	Recovery time improvement vs. anomaly-only.
Limited RL action set	Assess effect of restricted action space on resilience.	Mitigation effectiveness (%), task completion rate.
Network impairment variation	Test robustness under different latency / jitter conditions.	Detection latency sensitivity, stability under delay.

Table 4 summarizes the baseline and ablation configurations corresponding to Figure 4. Each configuration isolates specific aspects of system behaviour—from passive control (no defence) to dynamic learning (full RL). Together, they form the comparative framework for Section 5, enabling quantitative attribution of performance gains to the adaptive AI defence policy.

D. RL training process inside the DT

The DTST provides the RL agent with a training playground. Training is episodic and engineered to encourage robust, conservative policies.

State, action, reward (summary)

- **State (s):** compact vector combining anomaly

scores, recent message QoS (latency/jitter), control residuals (command − observed), and short telemetry window aggregates.

- **Action (a):** discrete mitigation actions (throttle non-critical topics, isolate suspect node, switch to degraded controller, trigger cross-validation).
- **Reward (r):** multi-objective scalar combining negative penalties for safety violations and control error, penalties for unnecessary interventions, and positive reward for rapid recovery and task completion. Formulated as:

$$r = -\alpha\cdot(\text{control\_error}) - \beta\cdot(\text{safety\_violation}) - \gamma\cdot(\text{intervention\_cost}) + \delta\cdot(\text{task\_progress}),$$

where weights $\alpha, \beta, \gamma, \delta$ are tuned via pilot runs.

Algorithm & hyperparameters

- Candidate algorithms: Proximal Policy

Optimization (PPO) or Soft Actor-Critic (SAC) for continuous action extensions; DQN or discrete PPO for discrete action spaces.

- Example hyperparameters (initial): learning rate = 3e-4, discount $\gamma = 0.99$, clip $\varepsilon = 0.2$ (PPO), rollout length = 2048 steps, batch size = 64.
- Domain randomization: vary physics parameters (friction, sensor noise), network QoS, and attack intensities across episodes to improve robustness and reduce the sim2real gap.

Training regime

- **Phase 1 (warm-start):** train on benign episodes to establish conservative baseline behaviours.
- Phase 2 (attack curriculum): progressively introduce attacks from low to high severity (curriculum learning), enabling the agent to acquire safe mitigation primitives before confronting complex campaigns.
- Phase 3 (mixed campaigns): randomized attack combinations to generalise policy.

Safety constraints during training

- A safety supervisor enforces hard constraints (e.g., joint limits, emergency stop) to prevent RL from generating unsafe commands. Policies are constrained or masked in action selection to prevent prohibited behaviour.

Evaluation & transfer

- Evaluate checkpoints periodically on a separate set of held-out attack scenarios. Select the best policies using a validation metric that combines recovery time and low intervention cost. Optionally, offline policy distillation can be performed for lower-latency deployment.

E. Assumptions and limitations (simulation-only)

We explicitly state the experiment design assumptions and the resulting limitations:

Assumptions:

- **Middleware fidelity:** ROS 2 emulation captures essential timing and topic semantics relevant to attacks modelled (but not all vendor-specific behaviour).
- **Sensor placeholders:** high-fidelity perception modules (e.g., camera neural nets) are approximated via simplified models or precomputed traces where necessary.
- **Deterministic replay:** deterministic seeds guarantee reproducibility inside DTST.

Limitations:

- **Sim2real gap:** policies trained in simulation may not transfer out of the box to physical robots due to unmodelled hardware idiosyncrasies and firmware-level vulnerabilities. Domain randomization reduces but does not eliminate this gap.
- **Scope of attacks:** the threat model focuses on middleware and sensor/command channels; hardware-level exploits, side-channel attacks, or supply-chain firmware compromises are out of scope.
- Real-time constraints: The computational overhead observed in the simulation must be validated on the target real hardware to ensure the control loop's timeliness.
- Safety validation for deployment: any policy intended for physical deployment requires independent safety certification and rigorous hardware-in-the-loop testing.

F. Reproducibility & artifact sharing

To support transparency, reproducibility, and community adoption, the Digital Twin Security Testbed (DTST) will be accompanied by a complete set of experimental artefacts and configuration templates. The following elements will be released conceptually as part of the research package:

1) Scenario and configuration files

a) Parameterised attack scripts, DTST scenario manifests, baseline configurations, and RL training profiles.

b) All files include deterministic seeds to ensure bit- level reproducibility.

2) Containerised runtime environments

a) Pre-configured Docker images encapsulating the physics simulator, ROS 2 network emulator, threat injection engine, and analysis tools.

b) Images will be version-locked to prevent dependency drift.

3) Logged datasets and evaluation traces

a) Anonymised simulation logs (baseline, single- attack, combined campaigns).

b) Each file includes metadata: Scenario ID, Campaign ID, Config Hash, and timestamp schema to support independent verification.

4) Metric computation scripts and statistical analysis notebooks

a) Python notebooks and helper scripts automatically calculate detection, resilience, and system-level metrics.

b) Includes bootstrapped confidence intervals and significance tests (e.g., Mann-Whitney U).

5) Documentation and orchestration guide

A step-by-step manual describing how to launch the DTST, reproduce core experiments, extend attack scenarios, and retrain AI models.

This reproducibility package ensures that the proposed framework is not only conceptually robust but also practically verifiable, contributing to scientific rigour and enabling other researchers to build upon the presented work.

V. Illustrative Results & Discussion

Although the present work focuses on the conceptual design and validation methodology of the DTST, preliminary simulation experiments provide valuable insight into expected defence behaviour.

A. Detection behaviour and anomaly signal quality

Across representative DoS, spoofing, and poisoning scenarios, the AI-driven anomaly detector demonstrates a consistently sharper rise in anomaly scores compared with static signature-based IDS baselines. Time-series plots (not shown) illustrate that:

- In flooding attacks, the detector responds as soon as

the message inter-arrival variance exceeds calibrated thresholds.

- In spoofing attacks, payload-level deviations trigger

latent-space anomalies 20–45% earlier than rule-based detectors.

- For gradual poisoning, the autoencoder models

detect distributional drift long before control performance visibly degrades.

This early-warning behaviour contributes directly to reduced downstream recovery times.

B. Effectiveness of RL-driven mitigation

Policy adaptation curves indicate that the reinforcement learning agent converges toward stable, conservative mitigation strategies. Typical learned behaviours include:

- throttling low-priority topics to stabilise bus utilisation,
- isolating suspicious nodes when repeated anomalies occur, and
- switching to a degraded safety controller during high-severity episodes.

Across multiple seeded campaigns, the RL-based defence consistently restores tracking error to baseline thresholds faster than static middleware-level interventions. In combined scenarios (DoS + spoofing), the RL policy avoids the escalation loops common to fixed-threshold defences.

C. Operational overhead and real-time feasibility

Profiling results from the simulated environment show:

- **The RL** policy's decision latency remains well

below typical ROS 2 control-loop cycles (5–10 ms).

- **CPU overhead** remains moderate, with peak loads

occurring during anomaly scoring windows rather than mitigation.

- **No frame drops** observed in 100 Hz physics simulation under full defence stack load.

These indicators suggest that the defence framework is computationally feasible for hardware-in-the-loop extensions, pending real-device validation.

D. Practical implications for robotics manufacturers and integrators.

The findings demonstrate clear value for operational environments:

- DTs provide a **safe sandbox** to validate cyber

resilience without risking downtime or hardware damage.

- Adaptive AI defence outperforms static IDS in

detection latency, **false-positive avoidance**, and **recovery stability**.

- Manufacturers can leverage DTST to certify

defensive policies before field deployment.

- Integrators gain a method for evaluating the

cyber-physical stability of networked robots under varying network conditions and operational loads.

Overall, the DTST supports a shift from reactive, rule-based cybersecurity toward **proactive, learning-driven resilience** in robotic systems.

VI. Conclusion & Future Work

This paper presented a simulation-based Digital Twin Security Testbed (DTST) for developing, training, and evaluating AI-driven adaptive cybersecurity mechanisms for robotic systems. The DTST integrates high-fidelity physics simulation, ROS 2 traffic emulation, a configurable threat-injection engine, and an anomaly-detection-plus-RL defence stack into a coherent, reproducible experimental environment.

The study demonstrates that digital twins provide a low-risk, high-fidelity platform for cyber-physical defence research—particularly valuable when testing on real hardware is unsafe, expensive, or operationally disruptive. The proposed AI-adaptive defence exhibits improved detection latency and recovery behaviour compared with static IDS baselines, highlighting the promise of learning-enabled mitigation in complex robotic environments.

Future research will extend this framework in several directions:

- Multi-robot and swarm scenarios, where

distributed attack surfaces and coordination constraints amplify cyber-physical risk.

- Federated and cloud-hosted digital twins,

enabling cross-facility resilience studies and collaborative defence training.

- Hardware-in-the-loop validation, progressively

bridging the sim2real gap through sensor-level calibration and network-timing alignment.

- Integration with zero-trust and identity-

centric architectures, positioning the DTST as a foundation for next-generation secure robotic ecosystems.

As robotics continues to evolve into highly connected, intelligent, cyber-physical systems, simulation-driven security validation will become a core component of engineering practice. This work contributes a structured, extensible foundation for that transition.

REFERENCES:

[1] Alcaraz, C., & Zeadally, S. (2015). Critical infrastructure protection: Requirements and challenges for the 21st century. *International Journal of Critical Infrastructure Protection, 8*, 53–66. https://doi.org/10.1016/j.ijcip.2014.12.002

[2] Althoff, D., Karreman, D. E., & Bakker, T. (2023). Cybersecurity challenges in robotics: A survey of vulnerabilities, attacks, and mitigations. *Robotics and Computer-Integrated Manufacturing, 82*, 102517. https://doi.org/10.1016/j.rcim.2022.102517

[3] Camci, F., & Chinnam, R. B. (2019). Digital twins for predictive maintenance: State of the art and future research. *Procedia CIRP, 84*, 47–52. https://doi.org/10.1016/j.procir.2019.04.013

[4] Dieber, B., Kacianka, S., Rass, S., & Schartner, P. (2017). Security for the Robot Operating System. *Robotics and Autonomous Systems, 98*, 192–203. https://doi.org/10.1016/j.robot.2017.09.017

[5] Ferreira, P., Leal, A., & Silva, R. (2023). Cybersecurity in ROS 2-based robotic systems: Vulnerabilities, risks, and mitigation strategies. *Robotics, 12*(3), 77. https://doi.org/10.3390/robotics12030077

[6] Gouda, K. C., & Liu, C. (2020). Artificial intelligence for adaptive cyber defense in industrial CPS: A review. *IEEE Transactions on Industrial Informatics, 16*(12), 7642–7653. https://doi.org/10.1109/TII.2020.2988861

[7] Hassan, M., Pourkhalili, A., & Abdallah, M. (2023). Digital twin architectures for cybersecurity testing of cyber-physical systems: A comprehensive survey. *IEEE Access, 11*, 22145–22171. https://doi.org/10.1109/ACCESS.2023.3245690

[8] Koopman, P., & Wagner, M. (2017). Autonomous vehicle safety: An interdisciplinary challenge. *IEEE Intelligent Transportation Systems Magazine, 9*(1), 90–96. https://doi.org/10.1109/MITS.2016.2583491

[9] Lee, J., Bagheri, B., & Kao, H. A. (2015). A cyber-physical systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters, 3*, 18–23. https://doi.org/10.1016/j.mfglet.2014.12.001

[10] Miciolino, E., Di Caterina, G., & Sgorbissa, A. (2021). Robotic manipulation under cyber attack: Analysis, detection, and mitigation. *IEEE Robotics and Automation Letters, 6*(2), 1231–1238. https://doi.org/10.1109/LRA.2021.3058073

[11] Quarta, D., Poggi, A., Doan, A., Polino, M., Visconti, A., Maggi, F., Zanero, S., & Papotti, P. (2017). An experimental security analysis of an industrial robot controller. *Proceedings of the 38th IEEE Symposium on Security and Privacy*, 268–286. https://doi.org/10.1109/SP.2017.21

[12] Skorobogatov, S. (2021). Digital twins in robotics: Applications, challenges, and future directions. *Robotics and Autonomous Systems, 145*, 103838. https://doi.org/10.1016/j.robot.2021.103838

[13] Tsukada, S., Nakajima, Y., & Kadoshima, J. (2022). Detecting cyber attacks on robotic control loops using deep autoencoders. *IEEE Access, 10*, 95642–95657. https://doi.org/10.1109/ACCESS.2022.3206661

[14] Wan, J., Cao, J., Fan, T., & Wang, Z. (2020). Digital twin for cyber-physical production systems: A comprehensive survey. *IEEE Transactions on Industrial Informatics, 16*(12), 7688–7699. https://doi.org/10.1109/TII.2020.2991146

[15] Zhang, Y., Wang, L., & Zhang, S. (2021). Reinforcement learning for resilient cyber-physical control under adversarial disturbances. *IEEE Transactions on Cybernetics, 51*(7), 3478–3491. https://doi.org/10.1109/TCYB.2019.2950072