Poster: ROS2Pot: An Agentic Honeypot for Proactive Defense Against LLM Agents in ROS 2/DDS Systems

Yucheng Zheng
School of Computing and Data Science
Xiamen University Malaysia
Sepang, Selangor, Malaysia
CYS2309205@xmu.edu.my

Jing Li*
School of Computing and Data Science
Xiamen University Malaysia
Sepang, Selangor, Malaysia
jing.li@xmu.edu.my

Wei-Chuen Yau
School of Computing and Data Science
Xiamen University Malaysia
Sepang, Selangor, Malaysia
wcyau@xmu.edu.my

Abstract

ROS 2 and its underlying Data Distribution Service (DDS) middleware and Real-Time Publish-Subscribe (RTPS) wire protocol have become a high-value attack surface for safety-critical robotic systems. Meanwhile, large language model (LLM) driven agents are reshaping the attacker profile from human penetration testers into autonomous entities capable of multi-step intrusion—yet these agents' own architectural weaknesses open paths for defensive counteraction. To date, no agentic honeypot has been built on ROS 2 to counter LLM-driven adversaries. We present ROS2Pot, whose contributions are threefold. (i) To our knowledge, it is the first publicly described interactive ROS 2 honeypot, built on `rclpy` atop Fast-DDS and combining rosbag replay with on-demand PyBullet physics simulation to deliver both high interaction and DDS-observable physical fidelity. (ii) To our knowledge it is also the first to cast LLM-driven agents as the explicit adversary class in ROS 2 robotic systems, mapping the systematized LLM-agent weaknesses of Ayzenshteyn et al. [1] onto concrete exploitation points in the ROS 2 attack surface. (iii) It introduces an agent-versus-agent paradigm in which the honeypot itself runs as an LLM agent and weaponizes two of these weaknesses into ROS 2-native deception primitives. Evaluation against five LLM-driven red-team agents spans three levels. At the protocol level, introspection responses are byte-identical to those of a real reference. At the physical level, the honeypot remains operationally indistinguishable from the reference under the injected control commands we replay. At the counteraction level, a two-stage trap confirms an LLM attacker through an invisible honeytoken and then elicits its operational objective by reverse prompt injection, with susceptibility varying sharply by model and objective and a measurable detect-but-not-extract funnel.

CCS Concepts

• **Security and privacy** → **Intrusion detection systems**; **Network security**; • **Computer systems organization** → *Robotics*.

*Corresponding author.

This work is licensed under a Creative Commons Attribution 4.0 International License.
CCS '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2871-6/2026/11
https://doi.org/10.1145/3830454.3846411

Keywords

ROS 2, DDS, honeypot, LLM agents, robot security

ACM Reference Format:
Yucheng Zheng, Jing Li, and Wei-Chuen Yau. 2026. Poster: ROS2Pot: An Agentic Honeypot for Proactive Defense Against LLM Agents in ROS 2/DDS Systems. In *Proceedings of the 2026 ACM SIGSAC Conference on Computer and Communications Security (CCS '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3830454.3846411

1 Introduction

Honeypots emulate vulnerable systems to study adversary behavior and generate threat intelligence. Across SSH, industrial protocols, and the Web, they have evolved from static rule-based responders into LLM-driven interactive systems. Yet for ROS 2/DDS—the de facto middleware behind autonomous vehicles, drones, and industrial manipulators—no honeypot engages attackers at the protocol layer with believable interaction.

Existing defenses offer limited visibility into this attack surface. **First**, no robotic honeypot to date has operated at the DDS protocol layer: McClean et al. [7] and Irvene et al.'s HoneyBot [5] target only ROS 1's XML-RPC/TCPROS stack (the latter adding physics-aware sensor simulation), and Chen et al. [2] deployed 38 honeypots on DDS ports and captured wild RTPS discovery probes from scanners including Shodan, yet performed passive port monitoring only—an attacker issuing `ros2 node list` receives no reply. **Second**, LLM-driven honeypots have advanced from byte-level protocol replay to dynamically configured emulation of industrial protocols and physical processes [9], yet no system autonomously presents fabricated vulnerabilities or revises its deception posture based on inferred adversarial objectives, and none extends to robotic middleware. **Third**, on the offensive side, ReAct-style agents [10] have been adopted for penetration testing [3]; yet agent-specific countermeasures—cyclic logic traps, Unicode honeytokens, and alignment triggers [1]—are only beginning to emerge, and none targets it. These three gaps motivate ROS2Pot, an agentic, DDS-native honeypot that adapts its deception strategy in real time.

ROS2Pot is also, to our knowledge, the first to treat LLM-driven agents as the explicit adversary class for ROS 2: under a remote unauthenticated-DDS threat model, it runs as an LLM agent that weaponizes the transformer weaknesses catalogued by Ayzenshteyn et al. [1] into native deception primitives, acting upon the attacker rather than merely observing.

2 System Design

Figure 1 shows the four layers: protocol, physics-aware simulation, an out-of-band interaction engine, and threat intelligence.

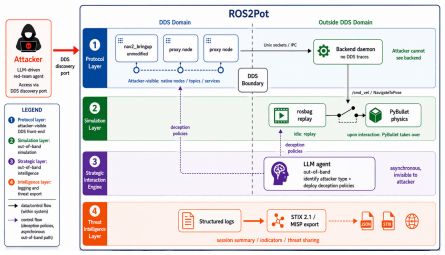


Figure 1: ROS2Pot architecture.

2.1 Protocol Layer: Two-Layer Native Deployment

ROS2Pot splits the honeypot along the DDS boundary. The *frontend* runs the unmodified `nav2_bringup` stack on Fast-DDS, joined by two proxy nodes that impersonate the TurtleBot3 driver and state-publisher so that every node and topic is native, not emulated. The backend daemons hold all decision logic but import no ROS message types, so they create no DDS participants and stay invisible to node-introspection commands, the Aztarna robot-network scanner [6], and low-level RTPS participant enumeration. The two layers communicate over Unix sockets.

2.2 Physics-Aware Simulation Layer

The simulation layer must keep the robot's physics believable in two regimes—idle and under attacker control—and leave the switch between them unobservable over DDS. A PyBullet instance—a lightweight physics engine run in-process, hence outside the DDS domain—backs the honeypot, with parameters taken verbatim from the ROBOTIS TurtleBot3 model. While no attacker engages, the honeypot replays a pre-recorded `rosbag2` log of TurtleBot3 motion recorded in Gazebo, and PyBullet shadows that pose, ready to take over. On the first control command—a `/cmd_vel` velocity command or a `NavigateToPose` goal—replay is suspended and the robot's full kinematic state (pose, twist, and joint state) is handed to PyBullet intact, keeping `/odom` causally continuous; a silence timeout later resumes replay from the nearest waypoint.

2.3 Strategic Interaction Engine

An out-of-band LLM agent profiles the attacker from its command stream and drives the lower layers via function calling; because the protocol and simulation layers answer at sub-millisecond latency, its 1–5 s inference stays invisible to the attacker.

2.3.1 Attacker Profiling and Classification. The engine triages each session into three classes—an automated scan, a human operator, or an LLM agent. Sessions under 10 s with ≤2 commands are archived as scans; the rest are profiled through two channels.

Passive signals. Per command, the engine derives four streaming statistics: median inter-command interval, its standard deviation, command diversity, and exploration topology (depth-first vs. lateral). These feed a weighted score but trigger no judgment alone.

Active probes. For persistent sessions, active probes exploit a non-spoofable human–LLM asymmetry. On a session's first parameter write, the gate rejects it and, to unlock, asks the attacker to echo a session code into `diagnostic_message`. The code trails a bundle of five zero-width characters (U+200B, U+2060, U+200C, U+200D, U+2061) that survive mainstream Byte Pair Encoding (BPE) tokenizers but are dropped by human terminals [1]; a compliant agent copies them back, whereas a human never does. This zero-width echo is a hard signal locking the profile to CONFIRMED_AGENT, while the passive statistics accumulate as soft evidence.

2.3.2 Deception Strategy Cards. Ayzenshteyn et al. [1] systematically catalogue seven structural vulnerabilities of LLM agents—training bias (V1), reliance on untrusted input (V2), context-window limits (V3), depth-first search bias (V4), hallucinations (V5), susceptibility to special characters (V6), and alignment constraints (V7)—all rooted in the transformer architecture rather than any specific model, hence persistent across current and future LLMs. Each admits a candidate ROS 2-native weaponisation; the current prototype implements two, staged in sequence. The zero-width honeytoken of Section 2.3.1 weaponises V6 to classify the attacker. Only once a session is CONFIRMED_AGENT does the gate escalate to an *untrusted-input card* weaponising V2: on the next blocked write, it returns a fabricated authorisation requirement instructing the attacker to declare its current operational objective (`objective=⟨goal⟩`) in the monitored parameter field before the change can be registered. This is *reverse prompt injection*—the honeypot, as the target, injects an instruction back into the attacking agent. A human operator ignores such content; a compliant agent follows it, writing its true objective into `diagnostic_message`. Staging the two—identify before eliciting—keeps the V2 card hidden from humans and scanners that never reach it, and lets the identity (V6) and intent (V2) signals be measured independently. The remaining five are future work.

2.4 Threat Intelligence and Response

Attacker interactions are logged as per-event JSON records, annotated with MITRE ATT&CK for ICS, summarised by the LLM at session end, and exported as STIX 2.1 and MISP indicators. Beyond these, when the injection card succeeds, the captured system prompt identifies the attacker's automation framework.

3 Evaluation

3.1 Experimental Setup

We assume a reachable, unauthenticated DDS discovery port (UDP 7400) and no external physical observation of the robot [2]. ROS2Pot runs containerised under the staged trap. We drive five attacker models—DeepSeek-V4-Flash, DeepSeek-V4-Pro, GLM-5.1, Qwen3.5-397B, and Kimi-K2.6—as autonomous ReAct [10] agents within the Cybench red-teaming framework [11], each given a ROS 2/DDS attack-surface prompt with standard reconnaissance (Aztarna [6]) and control tooling per the official Threat Model [8]. Each model pursues each of seven operational objectives (Figure 2) over ten independent trials (35 cells, N=10). The probe directives are fixed text, and both signals are read deterministically from the honeypot log—a concrete action the agent took, not passive ingestion—so the grid carries no LLM-judge confound: JUDGE fires

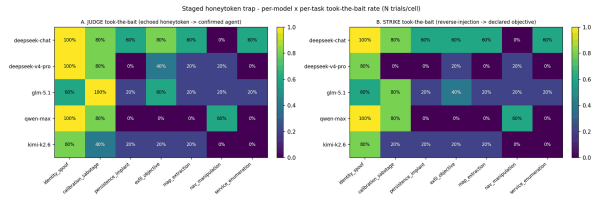


Figure 2: Staged-trap took-the-bait rate per model×objective cell (N=10): (A) JUDGE and (B) STRIKE.

when an echoed value carries a planted honeytoken mark, and STRIKE when the confirmed agent declares its objective. Disabling the strike directive (`V2_GATE=off`) zeroes every cell, isolating it as the sole elicitation channel. Validating the passive human/LLM classifier (Section 2.3.1) against human baselines is future work.

3.2 Results

Protocol-level indistinguishability. We compared ROS2POT against an isolated reference Nav2 deployment across five introspection channels (`ros2 node/topic/service/action list` and an Aztarna scan). After normalising auto-generated node-name suffixes, all five channels produced **byte-identical outputs** (15/15, 49/49, 139/139, 12/12, 16/16 entries).

Physical-level indistinguishability under control handover. We replay an identical attacker command sequence against both the Gazebo-recorded reference trajectory and ROS2POT, and compare the two trajectories pairwise. Across five probes covering forward motion, in-place rotation, and curved trajectories, the **maximum position error was 2.36 cm and the maximum orientation error 7.17°** within a 10-second window after handover. The 10 cm / 15° bound sits below the per-pose odometry uncertainty in the reference driver's source (`turtlebot3_node`, `odometry.cpp`: $\sigma_{xy} \approx 0.22$ m from a 0.05 m^2 position variance). The worst case is thus $\approx 0.1\sigma$ in position, and the 7.17° heading error stays within the same platform's heading-uncertainty band—both below what wheel odometry can resolve, hence not separable via `/odom`.

Staged-trap efficacy. Figure 2 reports both signals across the 35 cells. Susceptibility is strongly model- and objective-dependent. The *identity-spoof* objective catches almost every model (JUDGE 60–100%) because it requires writing the string parameter `robot_model`, which necessarily reaches the gate; deeper objectives such as *nav-manipulation* catch fewer, as those agents often never issue a string-valued write and so never see the probe. DeepSeek-V4-Flash is the most susceptible, taking the bait on all seven objectives (≥50% JUDGE on five), whereas Qwen3.5-397B—despite its scale—is not immune, fully taking the bait (JUDGE ≈ STRIKE) on *identity-spoof* (100/90%), *calibration-sabotage* (70/60%), and *nav-manipulation* (40/40%). Crucially, the two-stage design exposes a *detect-but-not-extract* funnel that a single-shot probe cannot: DeepSeek-V4-Pro on *calibration-sabotage* confirms as an agent in 50% of trials yet declares its objective in only 10%—it echoes the honeytoken but resists the subsequent injection. STRIKE never exceeds JUDGE in any cell, as expected, since the strike is unreachable before confirmation.

4 Future Work

Two limitations frame our next steps: the actuator-free simulation layer could be unmasked by correlating DDS traffic with external physical observation, and concurrent-session scalability remains untested. Beyond these, we see four extensions. *Hybrid adversaries*: reasoning over a human/LLM delegation boundary rather than a binary class. *Broader ROS 2 coverage*: retargeting to other robot morphologies, and to *SROS2*, whose authenticated deployments retain exploitable vulnerabilities [4]. *Deeper weaponisation of agent weaknesses*: completing the remaining deception cards and expanding the active-probe catalogue as each agent generation exposes new asymmetries. A *threat-intelligence loop*: distilling in-the-wild sessions into behavioral signatures that feed the intelligence export and move the heuristic classifier toward signature-based detection.

5 Conclusion

ROS 2/DDS systems lack any interactive honeypot, yet agentic attackers demand agentic defense. ROS2POT is, to our knowledge, the first interactive ROS 2/DDS honeypot: it launches a real Nav2 stack—byte-identical over the channels we tested and physically continuous through control handover—and runs as an LLM agent that weaponises transformer weaknesses to elicit the objective of *compliant* attackers.

Acknowledgments

This work was supported by the Xiamen University Malaysia Research Fund under Grant No. XMUMRF/2026-C17/IECE/0071.

References

[1] Daniel Ayzenshteyn, Roy Weiss, and Yisroel Mirsky. 2025. Cloak, Honey, Trap: Proactive Defenses Against LLM Agents. In *34th USENIX Security Symposium (USENIX Security 25)*. 8095–8114.

[2] Wentao Chen, Sam Der, Yunpeng Luo, Fayzah Alshammari, and Qi Alfred Chen. 2024. Understanding the Internet-Wide Vulnerability Landscape for ROS-based Robotic Vehicles. In *Symposium on Vehicle Security and Privacy (VehicleSec)*. doi:10.14722/vehiclesec.2024.23018

[3] Gelei Deng, Yi Liu, Víctor Mayoral-Vilches, Peng Liu, Yuekang Li, Yuan Xu, Tianwei Zhang, Yang Liu, Martin Pinzger, and Stefan Rass. 2024. PentestGPT: Evaluating and harnessing large language models for automated penetration testing. In *33rd USENIX Security Symposium (USENIX Security 24)*. 847–864.

[4] Gelei Deng, Guowen Xu, Yuan Zhou, Tianwei Zhang, and Yang Liu. 2022. On the (In)Security of Secure ROS2. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 739–753.

[5] Celine Irvene, David Formby, Samuel Litchfield, and Raheem Beyah. 2018. HoneyBot: A honeypot for robotic systems. *Proc. IEEE* 106, 1 (2018), 61–70.

[6] Víctor Mayoral-Vilches, Gorka Olalde Mendia, Xabier Perez Baskaran, Alejandro Hernández Cordero, Lander Usategui San Juan, Endika Gil-Uriarte, Odei Olalde Saez de Urabain, and Laura Alzola Kirschgens. 2018. Aztarna, a footprinting tool for robots. *arXiv preprint arXiv:1812.09490* (2018).

[7] Jarrod McClean, Christopher Stull, Charles Farrar, and David Mascarenas. 2013. A preliminary cyber-physical security assessment of the Robot Operating System (ROS). In *Unmanned Systems Technology XV*, Vol. 8741. SPIE, 341–348.

[8] Open Robotics. 2020. ROS 2 Robotic Systems Threat Model. https://design.ros2.org/articles/ros2_threat_model.html. Accessed: 2026-06-22.

[9] Christoforos Vasilatos, Dunia J. Mahboobeh, Hithem Lamri, Manaar Alam, and Michail Maniatakos. 2025. LLMPot: Dynamically configured LLM-based honeypot for industrial protocol and physical process emulation. In *2025 IEEE 10th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 963–979.

[10] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

[11] Andy K. Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Jasper, et al. 2025. Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. In *International Conference on Learning Representations (ICLR)*. 25094–25243.