CEA Comité Español de Automática
XLVII JORNADAS DE AUTOMÁTICA CÓRDOBA 2026
UNIVERSIDADE DA CORUÑA

Jornadas de Automática

Cybersecurity: Protecting a Teleoperated ROS 2 Quadruped Robot

Mejía, Irene[a,*], Cervera, Enric[a], Marín, Raúl[b]

[a] *Robotic Intelligence Lab (RobInLab), Dept. of Computer Science and Engineering, Universitat Jaume I, Av. Vicent Sos Baynat, s/n, 12071 Castelló de la Plana, Spain.*
[b] *Interactive and Robotic Systems Laboratory (IRS-Lab), Dept. of Computer Science and Engineering, Universitat Jaume I, Av. Vicent Sos Baynat, s/n, 12071 Castelló de la Plana, Spain.*

To cite this article: Mejía, I., Cervera, E., Marín, R. 2026. Cybersecurity: Protecting a Teleoperated ROS 2 Quadruped Robot. Jornadas de Automática, 47. https://doi.org/10.17979/ja-cea.2026.47.13793

Resumen

Este artículo propone una arquitectura de ciberseguridad en profundidad para un robot móvil (Unitree Go2 Pro) que integra los mecanismos de seguridad existentes de ROS 2 en un marco unificado para la operación remota segura mediante comunicaciones inalámbricas. En lugar de basarse únicamente en un despliegue estándar de SROS2, el enfoque propuesto combina segmentación de red, seguridad a nivel del *middleware* mediante SROS2 en modo *Enforce*, comunicación DDS controlada con *CycloneDDS* y aislamiento mediante contenedores Docker. Una Jetson Orin Nano actúa como pasarela (gateway), permitiendo el descubrimiento controlado de nodos, el acceso restringido a los tópicos críticos y la comunicación exclusivamente entre entidades autorizadas.

La arquitectura se valida experimentalmente sobre una plataforma robótica cuadrúpeda real, evaluando tanto su funcionamiento como su impacto computacional. Los resultados demuestran una protección eficaz frente a accesos no autorizados con una sobrecarga de rendimiento reducida, mostrando que múltiples capas de seguridad complementarias pueden integrarse en un sistema embebido basado en ROS 2 sin comprometer su funcionamiento.

Palabras clave: Robots móviles, Robótica embebida, Modelado y control de sistemas robóticos en red, Telerrobótica, Sistemas robóticos autónomos, Tecnología robótica.

Cybersecurity: Protecting a Teleoperated ROS 2 Quadruped Robot

Abstract

This paper proposes a defense-in-depth cybersecurity architecture for a mobile robot (Unitree Go2 Pro) that integrates existing ROS 2 security mechanisms into a unified framework for secure wireless remote operation. Rather than relying solely on a standard SROS2 deployment, the proposed approach combines network segmentation, middleware-level security enforced through SROS2 in Enforce mode, controlled DDS communication with CycloneDDS, and Docker-based container isolation. A Jetson Orin Nano acts as a gateway, enabling controlled node discovery, restricted access to critical topics, and communication exclusively among authorized entities.

The architecture is experimentally validated on a real quadruped robotic platform, evaluating both its functionality and computational impact. The results demonstrate effective protection against unauthorized access with a reduced performance overhead, showing that multiple complementary security layers can be integrated into an embedded ROS 2 system without compromising its operation.

Keywords: Mobile robots, Embedded robotics, Networked robotic system modeling and control, Telerobotics, Autonomous robotic systems, Robotics technology.

*Autor para correspondencia: al415507@uji.es

1. Introduction

The progressive integration of mobile robots in industrial, logistics, healthcare, and inspection environments has increased the need to ensure not only their functional autonomy, but also the security of their communications and control processes. As cyber-physical systems, a software or network vulnerability can have direct physical consequences, such as unauthorized movements, loss of availability, or alteration of environmental perception.

Several works (Yaacoub et al., 2022; Vilches et al., 2018; Yang et al., 2024) indicate that robotic cybersecurity must be addressed from a multidimensional perspective, considering hardware, software, firmware, and communications. Among the most relevant threats are denial of service, false data injection, *Man-in-the-Middle* attacks, node spoofing, and unauthorized access to exposed interfaces. In distributed systems, where different nodes exchange control, localization, and perception data, these threats could compromise robot operation through unauthorized commands, communication blocking, or alteration of critical data.

ROS 2 introduces relevant improvements over ROS 1, particularly through its integration with DDS and the incorporation of security mechanisms based on DDS-Security (Mayoral-Vilches et al., 2022). However, these capabilities require proper configuration of access policies, certificates, and middleware, as well as complementary measures for network segmentation and execution environment isolation. Despite this, works that experimentally document a complete SROS2 deployment on real quadruped robots, including its computational impact on embedded platforms, remain scarce.

This work presents a ROS 2-based cybersecurity architecture for a Unitree Go2 Pro quadruped robot (Unitree Robotics, 2024). The proposal follows a defense-in-depth approach by combining network segmentation, SROS2 in Enforce mode, and Docker-based container isolation, addressing complementary threat surfaces related to reachability, authorization, and execution control.

The main contribution of this work is not the individual use of existing cybersecurity technologies, but their integration into a practical defence-in-depth architecture for a real ROS 2 system. The proposed solution combines network-level isolation, middleware-level authorization through SROS2, controlled DDS communication with CycloneDDS, and containerized deployment, and experimentally validates their joint operation and performance on an embedded robotic platform.

2. System Architecture

The developed system is based on a distributed architecture on ROS 2, supported by the middleware with CycloneDDS. The solution is structured into two blocks: the onboard system on the robot and an external supervision station.

- Robot (Jetson Orin Nano 8 GB): runs the perception and processing nodes, manages communication with the robot, publishes the system state (/odom, /tf, sensors) and exposes motion control through /cmd_vel.
- External station (PC): runs visualization and supervision via RViz and Foxglove, and can launch navigation with Nav2.

Table 1 summarizes the hardware configuration used in the physical system.

Table 1: System hardware configuration

Robotic platform	
Robot	Unitree Go2 Pro
Onboard computer	Jetson Orin Nano (8 GB RAM)
CPU	6× ARM Cortex-A78AE @ 1.7 GHz
Memory	8 GB RAM
Operating system	Ubuntu 22.04 + ROS 2 Humble
Middleware	CycloneDDS
External station	
Device	MSI Raider GE68HX-13VF
CPU	Intel i9-13950HX (24 cores)
Maximum frequency	Up to 5.5 GHz
Memory	32 GB RAM
GPU	NVIDIA RTX 4060 Mobile (8 GB)
Operating system	Ubuntu 24.04.4 LTS
WiFi network	
Router	TP-Link AX6600 Wi-Fi 6 Tri-Band Gaming Router
Model	Archer GX90
WiFi standard	802.11a/n/ac/ax (mixed mode)
Band used	5 GHz (5G-1)
PC network interface	Integrated WiFi
Jetson WiFi adapter	RTL88x2BU (Access Point mode)
Network architecture	Jetson as gateway / AP

2.1. Middleware selection: CycloneDDS

Communication between system components is based on DDS (*Data Distribution Service*), the technology upon which ROS 2 builds its distributed architecture. In this work, CycloneDDS was selected over Fast DDS due to its simpler configuration, predictable behavior in distributed environments, and compatibility with the environment previously used on the robot.

One of its main advantages is the ability to explicitly define network interfaces and limit discovery traffic through XML files. This feature is particularly important in the proposed architecture, where public, access point, and container network interfaces coexist, enabling controlled communication between the Jetson and the external station. Furthermore, its integration with ROS 2 security mechanisms facilitates the application of access and encryption policies to critical nodes and topics.

Additionally, the Unitree Go2 Pro SDK uses CycloneDDS by default, making its adoption the natural choice to ensure compatibility with the robot's native software stack and avoid introducing additional integration complexity.

2.2. Requirements

From a security standpoint, the system had to restrict access to critical ROS 2 topics to authorized nodes while keeping the robot isolated from the external network. The threat model assumes an external attacker connected to the same wireless network, capable of executing ROS 2/DDS tools, performing network scans, discovering topics, and publishing or subscribing to ROS 2 topics, but without valid SROS2 credentials or physical access to the robot or Jetson. From a deployment perspective, the solution had to be reproducible and compatible with containerized execution.

2.3. Use Case

The considered use case corresponds to a real teleoperation scenario in which the Jetson runs the robot-side nodes and the external PC supervises the system through RViz and Foxglove. The robot publishes state and perception data, while the external station consumes this information for monitoring and analysis (Figure 1).

The architecture ensures that this interaction occurs through the Jetson gateway, avoiding direct exposure of the robot to the external network and restricting access to authorized entities.

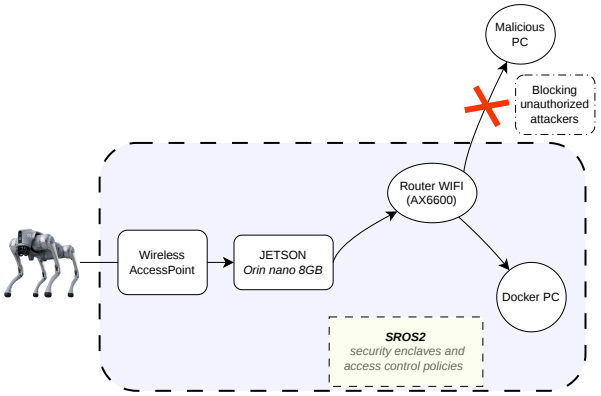

Figure 1: General scheme of the architecture and communications.

3. Network-Level Isolation

A first security measure has been implemented at the network level through the configuration of an Access Point on the Jetson, using an externally connected USB WiFi adapter configured manually. The architecture establishes controlled network segmentation, where the Jetson acts as an intelligent gateway between the robot and the external network.

To this end, the Jetson operates with two distinct interfaces: the integrated WiFi, in client mode, connected to the public network; and a USB WiFi adapter, in Access Point mode, which defines a private network dedicated to the robot. In this way, the robot is not directly connected to the external network, but only to the private network generated by the Jetson.

3.1. Access Point Implementation

The implementation of the Access Point required several significant adaptations at the network level. First, it was necessary to manually compile the driver for the RTL88x2BU USB WiFi adapter, ensuring its compatibility with the exact kernel version of the Jetson. A specific private subnet (*192.168.55.0/24*) was then configured, assigning the Jetson the address *192.168.55.1* as the gateway and the robot the address *192.168.55.59*.

In addition to IP address assignment, it was necessary to explicitly adjust network routes and metrics to ensure deterministic system behavior (Table 2). Specifically, dedicated routes toward the robot network were defined, interface metrics were tuned, and traffic destined for the robot was prevented from being resolved through incorrect interfaces (Angarita Noriega et al., 2025).

Table 2: Network interface configuration.

Interface	Role in the system	Priority (metric)
wlp1p1s0	External network connection	High (600)
wlx1cbfce3ae9d6	Access point	Very high (5)
docker0	Internal container network	N/A

3.2. Security Implications

This architecture allows the robot to be effectively isolated from the network. As a result, the robot is not visible from the general network, its exposure to scans and unauthorized access from the public network is significantly reduced, and its ROS topics are not openly exposed. All communications pass through a controlled point, which reduces the attack surface and facilitates traffic filtering and monitoring.

4. ROS 2 Security with SROS2

In addition to network isolation, security has been incorporated through SROS2 (Open Robotics, 2023), the ROS 2 security mechanism based on certificates, enclaves, and access policies, configured in Enforce mode.

To integrate this mechanism, the *go2_ros2_sdk* repository (Abizov, 2024) was adapted so that all nodes inherit the security configuration and operate within the same enclave.

The security policy is enabled through the following environment variables:

```
ROS_SECURITY_KEYSTORE=./src/go2_ros2_sdk/
    go2_security/go2_keystore/
ROS_SECURITY_ENABLE=true
ROS_SECURITY_STRATEGY=Enforce
```

This configuration specifies the location of the cryptographic material and enforces secure communication within the */go2* enclave. Following the principle of least privilege, the governance and permissions policies define which authenticated nodes are authorized to publish or subscribe to each topic, while all unspecified permissions are denied by default.

4.1. Foxglove and RViz within the Secure Architecture

Foxglove and RViz are incorporated as monitoring tools within the controlled communications scheme. Foxglove runs from the external station to remotely monitor the system state and perception data generated on the Jetson (Figure 2), while RViz is used primarily during development, debugging, and validation phases within the ROS 2 ecosystem.

The combination of network segmentation and SROS2 ensures that access to this data is restricted to systems with appropriate certificates and permissions. In this way, visualization tools do not act as open access points, but as authorized components within the secure architecture.

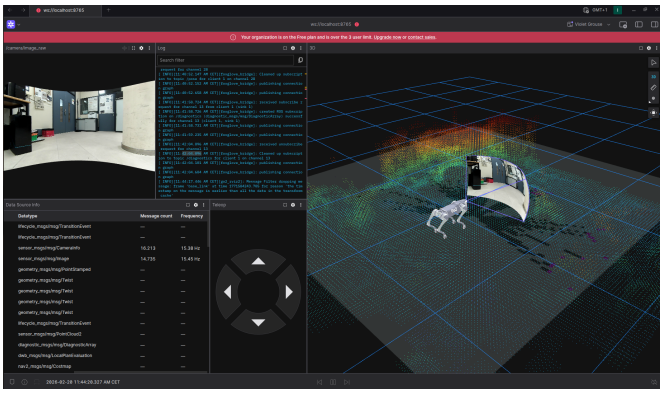

Figure 2: Robot visualization in Foxglove.

4.2. Comparison with VPN-Based and ROS 2 Security Approaches

Existing ROS 2 cybersecurity approaches typically rely on communication channel protection (e.g., VPNs), middleware-level security through DDS-Security and SROS2, or alternative frameworks such as ROS2Sec, which simplify authentication and access control management through a centralized security model (Park and Park, 2025). VPNs protect the communication channel but do not enforce ROS 2 publish/subscribe permissions, whereas a standard SROS2 deployment provides authentication and access control without preventing direct network exposure. Docker improves isolation and reproducibility, but does not replace middleware-level security.

The proposed architecture combines these complementary mechanisms into a defence-in-depth framework. Network segmentation limits reachability, SROS2 enforces node- and topic-level authorization, CycloneDDS enables controlled discovery, and Docker isolates the execution environment.

4.2.1. Unauthorized Access and Secure Flow

Even if an attacker were to gain access to a machine's terminal, directly modifying the enclave files would not be sufficient to compromise the architecture. Therefore, only nodes with valid credentials can be integrated into the system; policies cannot be modified without invalidating their signature, and any incorrect alteration causes runtime errors.

The secure operating flow of the system can be summarized in the following steps:

1. A keystore is created with the system's cryptographic material.
2. An enclave is defined (*/go2*), which groups the authorized nodes.
3. Each node is assigned specific certificates and permissions, cryptographically signed.
4. During launch, each node attempts to initialize within its enclave.
5. The middleware verifies certificates, signatures, and permissions before authorizing execution.

5. Containers and Access Control

The system has been deployed using Docker containers with the aim of improving reproducibility, controlling dependencies, and operationally separating the different execution environments. This layer does not replace the ROS 2 security model, but integrates with it: containers use the same keystore, operate within the corresponding enclave, and run nodes with valid certificates and defined access policies.

In this way, authorized containers preserve the same security policies, while Docker provides execution isolation and reproducible deployment.

6. Performance and Communications Impact Evaluation

The proposed cybersecurity architecture was evaluated in terms of computational and communication overhead by comparing CPU usage, memory consumption, and network throughput with and without security enabled. Each experiment lasted approximately 15 minutes under identical operating conditions, and the results presented correspond to a representative trial, as repeated experiments showed consistent behavior with only minor variations.

6.1. Impact on the Jetson

For the Jetson, GPU usage remained practically zero throughout the trials, since system visualization was performed on the external monitoring station, while the main load on the Jetson fell on CPU, memory, and communications.

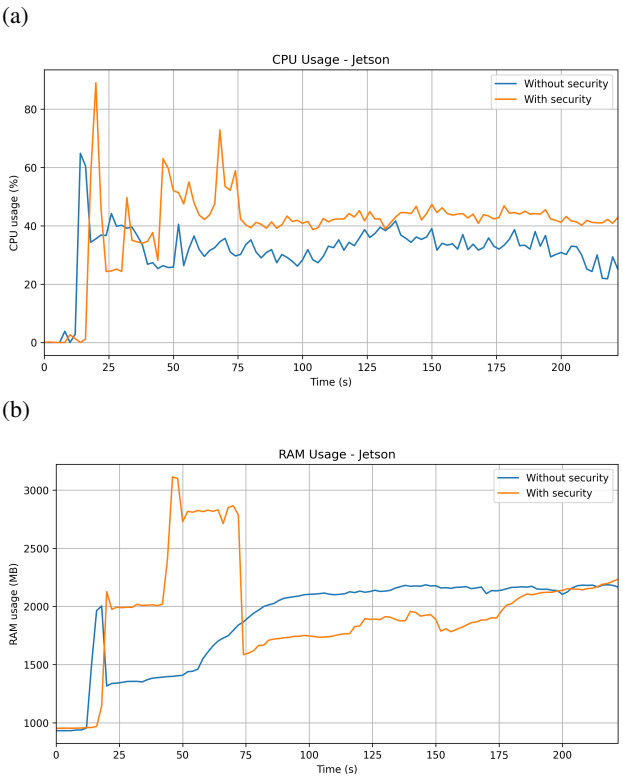

Figure 3: Resource usage on the Jetson Orin Nano: (a) CPU usage; (b) RAM consumption.

Figure 3 (a) shows the evolution of CPU usage on the Jetson over time. Although both curves exhibit notable variance during the first 100 s due to initialization and session establishment, the security-enabled scenario consistently shows higher utilization throughout. In steady-state operation, the unsecured scenario tends toward 30–35%, while the secured scenario stabilizes in the 40–43% range, yielding an approximate

overhead of 8–10 percentage points attributable to the cryptographic and authentication operations introduced by SROS2.

Figure 3 (b) presents the RAM consumption on the Jetson. During initialization, the secured scenario exhibits a pronounced peak of up to 3113 MB, associated with loading cryptographic material and establishing secure DDS sessions. Once steady state is reached, both scenarios converge to similar levels (2000–2200 MB), indicating that the memory overhead introduced by SROS2 is transient rather than sustained.

Network traffic was also analysed to evaluate the effect of the architecture on system communications, as shown in Figure 4. It is important to note that the two interfaces compared are not directly equivalent: the unsecured scenario captures raw DDS traffic over a shared WiFi interface, while the secured scenario measures traffic through the private access point managed by the Jetson. In this configuration, the main communication with the robot is routed through a segmented network path, while SROS2 restricts DDS discovery and topic access to authorized entities.

Under these conditions, the secured scenario exhibits lower throughput (1.5–3 Mb/s) than the unsecured one (5–8 Mb/s). This behaviour suggests that the proposed architecture reduces the amount of exposed communication by limiting DDS discovery and topic visibility to authorized entities. Therefore, the observed reduction should not be interpreted as a direct consequence of cryptographic overhead, whose effect is more clearly reflected in the increased CPU usage reported in Figure 3. A controlled comparison over equivalent interfaces is left as future work.

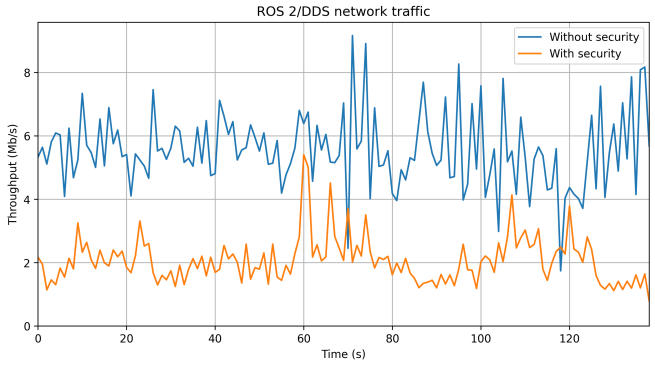

Figure 4: Comparison of network traffic.

These results indicate that the proposed architecture does not introduce a communication bottleneck under the tested conditions. The lower throughput observed in the secured scenario is mainly associated with the modified communication topology, where traffic is routed through the private access point managed by the Jetson instead of the direct WiFi interface used in the unsecured configuration. Overall, the results suggest that the proposed security mechanisms are suitable for practical deployment on embedded robotic platforms.

6.2. *Impact on the PC*

The external station is equipped with an Intel Core i9-13950HX, featuring 24 cores and 32 threads based on a hybrid architecture that combines performance cores (2 threads each) and efficiency cores (1 thread each). Given this computational capacity, accumulated CPU usage is reported across all logical processors and can therefore exceed 100% (Figure 5). Both scenarios show similar aggregate utilization in steady state, confirming that the security overhead is negligible on the PC side.

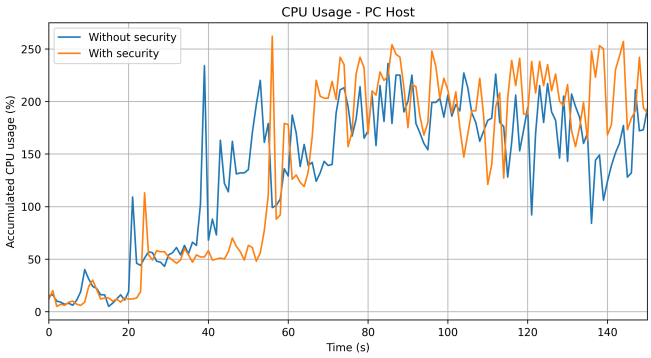

Figure 5: Comparison of CPU usage on the external PC.

Overall, the results show that the proposed cybersecurity architecture introduces a moderate and manageable overhead (Table 3). The observed computational cost remains acceptable given the security improvements achieved, demonstrating that operational functionality and cybersecurity can coexist without compromising system performance.

Table 3: Performance metrics summary.

Metric	**Without security**	**With security**
Jetson Orin Nano		
Mean CPU usage (%)	31.30	40.11
Peak CPU usage (%)	64.83	89.00
Mean RAM usage (MB)	1887.27	1958.32
Peak RAM usage (MB)	2186.00	3113.00
Network range (Mb/s)	5–8	1.5–3
External PC (accumulated)		
Mean CPU usage (%)	134.09	139.14
Peak CPU usage (%)	236.00	262.00

Therefore, the measurements on the external PC mainly reflect the visualization workload rather than the computational cost of the cybersecurity mechanisms themselves.

7. Experimental Results

The experimental validation covered multiple attack vectors, summarized in Table 4. All attempts were conducted from a machine on the same external WiFi network, in some cases with prior knowledge of the robot's IP address.

Table 4: Security validation against attack vectors.

Attack vector	**Blocking layer**	**Result**
/Cmd_vel commands	SROS2	Blocked
Subscribe to topics	SROS2	Blocked
Network scan	Network segmentation	Robot not visible
Connection to robot IP	Network segmentation	Connection refused
Topic discovery	SROS2 + segmentation	No topics returned
Modify enclave files	SROS2 signing	Node rejected

According to this threat model, the evaluation considers unauthorized publication of motion commands, subscription to protected topics, DDS topic discovery, network scanning, direct connection attempts, and modification of enclave configuration files.

In all cases the robot's odometric pose remained stable and no unauthorized command reached the platform. The most representative scenario involved three computers connected to the same external WiFi. Two computers with valid SROS2 credentials acted as authorized stations: one monitored the robot, while the other recorded ROS 2 traffic. A third computer acted as an attacker, using the same *ROS_DOMAIN_ID* as the robot but without valid SROS2 credentials.

Initially, the attacker repeatedly attempted to publish velocity commands to */cmd_vel*. This attempt corresponds to the initial interval of the plot, before the authorized command is sent. During this period, no valid */cmd_vel* command was observed by the authorized recording station and the odometric pose remained stable, indicating that the unauthorized publication did not induce robot motion.

Afterwards, an authorized node published velocity commands through */cmd_vel/linear/x*. Figure 6 compares the commanded linear velocity with the evolution of the robot odometric position. In this case, the command produced a measurable variation in */odom/pose/pose/position/x* and */odom/pose/pose/-position/y*, confirming that only authorized commands resulted in robot motion. Since the odometric twist field did not reliably reflect the robot motion in this setup, the validation was based on the temporal evolution of the odometric pose.

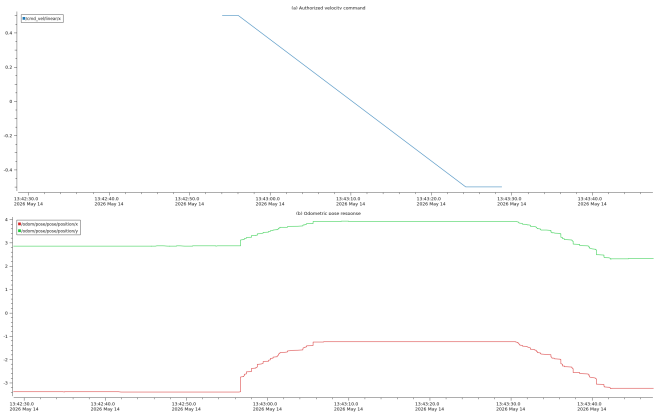

Figure 6: Robot response to /cmd_vel commands.

These results, together with the performance analysis presented in the previous section, show that the proposed architecture protects critical robot communications while maintaining normal operation. While not intended as a strict real-time solution, it provides a practical cybersecurity framework for real robotic deployments.

Additional attack scenarios include denial-of-service attacks, which may attempt to disrupt communications or exhaust system resources, credential compromise, which could allow an attacker to impersonate an authorized node, and Man-in-the-Middle attacks targeting communication channels. Although these scenarios were not experimentally evaluated, the proposed architecture would help mitigate their potential impact.

8. Conclusions

The proposed architecture for the Unitree Go2 Pro demonstrates that operational functionality and cybersecurity can be integrated in a real robotic platform. The solution combines network segmentation, SROS2-based node-level access control, and container isolation, forming a defence-in-depth strategy aligned with Zero Trust principles. (Rose et al., 2020).

The experiments conducted show that enabling SROS2 and network segmentation increases CPU and memory usage on the Jetson without affecting system operation or remote supervision. It was also verified that nodes without valid credentials could not integrate into the ROS 2 communication, reinforcing access control over critical topics and services.

Although the proposed architecture was validated on a real robotic platform, its evaluation was limited to the attack scenarios considered in this work. Future work includes evaluating additional attack scenarios, incorporating data-at-rest encryption through encrypted file systems, and extending authentication and access control mechanisms at the user level. These developments would further strengthen the system's overall security beyond the communications plane.

Acknowledgements

This work was made possible thanks to the resources provided by the RobInLab laboratory and the CIRTESU research group at Universitat Jaume I. This work was also developed within the framework of the CENTAURO research project (PLEC2023-010360), funded by the Spanish Ministry of Education and Science. The support, guidance, and follow-up of the professors involved in the project development are also gratefully acknowledged.

References

Abizov, N., 2024. go2_ros2_sdk: Unofficial ros2 sdk for unitree go2. `https://github.com/abizovnuralem/go2_ros2_sdk`.

Angarita Noriega, J. C., Marqués Verdegal, J., Pino Jarque, A., Cervera Mateu, E., Marín Prades, R., 2025. Sistema embebido multimodal portable para navegación y mapeo. In: Jornadas de Automática. Vol. 46. DOI: `10.17979/ja-cea.2025.46.12213`

Mayoral-Vilches, V., White, R., Caiazza, G., Arguedas, M., 08 2022. Sros2: Usable cyber security tools for ros 2. In: Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS). DOI: `10.1109/IROS47612.2022.9982129`

Open Robotics, 2023. Introducing ros 2 security. `https://docs.ros.org/en/humble/Tutorials/Advanced/Security/Introducing-ros2-security.html`.

Park, J., Park, H., 2025. Design and implementation of ros2 security module for performance and security harmonization. IEEE Access PP, 1–1. DOI: `10.1109/ACCESS.2025.3649485`

Rose, S., Borchert, O., Mitchell, S., Connelly, S., Aug. 2020. Zero Trust Architecture. National Institute of Standards and Technology. DOI: `10.6028/nist.sp.800-207`

Unitree Robotics, 2024. Go2 pro robot. `https://www.unitree.com/go2`.

Vilches, V. M., Kirschgens, L. A., Calvo, A. B., Cordero, A. H., Pisón, R. I., Vilches, D. M., Rosas, A. M., Mendia, G. O., Juan, L. U. S., Ugarte, I. Z., Gil-Uriarte, E., Tews, E., Peter, A., 2018. Introducing the robot security framework (rsf), a standardized methodology to perform security assessments in robotics. CoRR abs/1806.04042.

Yaacoub, J.-P. A., Noura, H. N., Salman, O., Chehab, A., 2022. Robotics cyber security: vulnerabilities, attacks, countermeasures, and recommendations. International Journal of Information Security 21, 115–158. DOI: `10.1007/s10207-021-00545-8`

Yang, S., Guo, J., Rui, X., 2024. Formal analysis and detection for ros2 communication security vulnerability. Electronics 13 (9). DOI: `10.3390/electronics13091762`