Assessing Network Level Reconnaissance Risks in ROS2 Enabled Service Robots for Smart Infrastructure Operations

Abdul Shakir[1], Hiba Jomaa[1], and Bharadwaj R. K. Mantha*[1,2]

[1]College of Engineering, University of Sharjah, Sharjah, United Arab Emirates.
[2]Department of Civil Engineering, Indian Institute of Technology, Madras, India.
shakirabdul31@gmail.com, u22105324@sharjah.ac.ae, rmantha@sharjah.ac.ae

Abstract –

Service robots have been increasingly deployed across several infrastructure applications. ROS2 has become the widely used operating system for service robots due to its performance and compatibility. However, its default configuration lacks authentication, encryption, or access control. While prior studies have highlighted general ROS2 security weaknesses, the feasibility of reconnaissance attacks in realistic service robot deployments remains underexplored. A controlled setup using ROS2 Humble was developed, in which a custom publisher node continuously transmitted robot positional data (i.e., real time location) over data distribution service (DDS) topics. Fast DDS Spy was used to discover domain participants, topics, data writers, while wireshark captured and analyzed underlying real-time publish–subscribe (RTPS) traffic. The experiments were repeated across ten independent simulation runs to ensure consistent observations. The results demonstrate that in all ten simulation runs, an unauthorized entity on the same network could access eight types of sensitive operational data (e.g., robot's internet protocol (IP)) with consistent observation across trials. These findings highlight integrity and confidentiality risks in default ROS2 deployments and emphasize the need for stronger security controls to safeguard service robots particularly in collaborative environments. These observations can serve as valuable guidance for researchers and practitioners designing and utilizing autonomous service robots. In addition, the study's findings provide valuable insights and significantly complement the smart city initiatives and the UAE vision of a resilient, secure, sustainable, and intelligent infrastructure ecosystem.

Keywords – On-demand Cobots; Cybersecurity; Vulnerabilities; Construction automation; AEC industry; Connected sites; Futuristic infrastructure.

1 Introduction

Service robots are increasingly deployed to perform task-oriented functions across a wide range of real-world environments, including infrastructure inspection, facility management, hospitality, healthcare, and public services. These robots are designed to execute specific operational tasks such as monitoring, inspection, navigation, or human interaction, with development efforts primarily focused on achieving reliable application-level functionalities. As a result, less attention is often given to the underlying operating system and security architecture that support these capabilities, particularly in early-stage development and deployment [1]. In many cases, service robots operate while connected to shared or public wireless networks, remaining continuously exposed to the internet as they roam through open environments, which significantly expands their attack surface. In architecture engineering construction and operations (AECO) industry, robots and drones are used for site inspection, progress monitoring, and hazard detection. In indoor built environments such as commercial buildings and facilities, service robots support environmental monitoring, security patrols, and asset inspection [2][3]. Comparable task driven deployments are observed in hospitality and healthcare sectors, as well as in emerging autonomous transportation systems. The practical importance of these applications is reflected in city-level initiatives; for example, Dubai aims to transition 25% of its transportation systems to autonomous modes by 2030, signaling the growing reliance on robotic systems within critical urban infrastructure.

To enable communication and coordination among robotic components, the Robot Operating System (ROS) is widely adopted as a modular software framework for robotic development. Since its initial open-source release in 2009, ROS1 facilitated rapid prototyping and system integration by providing reusable libraries and

standardized interfaces across heterogeneous hardware and software platforms. However, ROS1 was not designed with security or real-time guarantees as primary considerations, and its key limitations are discussed in detail in Section 2.2 [4]. To address these drawbacks, ROS2 was introduced as a redesigned framework targeting modern service robot requirements. ROS2 improves real-time performance, supports multi-robot and distributed deployments, and adopts middleware technologies such as the Data Distribution Service (DDS) to enable scalable communication [4] [5]. DDS is a middleware standard that enables real-time, reliable, and scalable data exchange between distributed systems using a data centric publish subscribe communication model.

Despite these architectural improvements, many ROS2 deployments continue to operate using default middleware configurations that do not enforce authentication, encryption, or fine-grained access control. As service robots largely operate over shared wireless networks or public Wi-Fi, these default configurations expose communication channels to passive and active adversaries. Vulnerabilities at the operating system or middleware level can compromise confidentiality, integrity, and availability of robotic data and control flows, potentially affecting mission execution and human safety [6]. Therefore, systematic analysis of network-level exposure, protocol behavior, and security weaknesses in ROS2-based systems is essential. This applies to all service robot deployments, since continuous connectivity and human interaction can increase risks, potentially affecting the confidentiality, integrity, and availability across all layers of robotic operation. This study addresses this gap by examining the type of security relevant information that can be identified using currently available tools. Unlike prior studies that limited their analysis to traffic observation and protocol identification, this study uniquely combines Fast DDS Spy at the middleware layer with Wireshark at the network layer to systematically enumerate eight categories of sensitive operational data extractable through passive reconnaissance, without authentication or direct system interaction. It also serves to enhance the awareness among business stakeholders by demonstrating how the under prioritization of these risks can lead to significant financial and operational consequences incase of potential cybersecurity incidents involving service robotic operation.

2 Literature Review

The reviewed literature indicates that although ROS2 introduces significant improvements over its predecessor such as enhanced modularity and the adoption of DDS based communication, its security architecture is still not fully mature [7]. Multiple studies such as [5],[6],[8] report persistent weaknesses, including 1) insufficient authentication mechanisms between nodes, 2) limited or optional encryption configurations, and the 3) absence of built-in intrusion detection capabilities. As a result, ROS based systems remain vulnerable to a range of cyber threats. These threats can affect the performance of any service robot, regardless of its application, by putting the robot's primary tasks at risk. These threats can lead to several potential impacts on robotic systems, including 1) disruption of service continuity, 2) degradation of system performance, and 3) compromise of one or more elements of the Confidentiality, Integrity, and Availability (CIA) triad [9]. Furthermore, the literature highlights that the security posture of a ROS2 system is strongly influenced by the underlying DDS middleware implementation.

For the safe operation of service robots, multiple aspects were examined from prior studies to identify key findings and research gaps. The review begins at the build environment level, where Section 2.1 discusses operating environments, followed by Section 2.2, which analyzes the prominent DDS in robot communication. The discussion then extends to the larger system level, where Section 2.3 focuses on network-level communication and analysis based on relevant studies. This motivated the division of the literature review into three sections: 2.1 Analysis of different types of environments used to assess vulnerabilities of service robots in ROS2, 2.2 Analysis of DDS suppliers, i.e., the middleware entities, to examine the state-of-the-art literature in this domain, and 2.3 Review of studies related to understanding the network operability of service robots. Additionally, the review aimed to critically analyze the current literature to identify gaps in ROS2 security research, highlighting areas where cybersecurity risks and vulnerabilities have been underexplored. Together, these sections offer a cohesive understanding of the environment, middleware characteristics, and network interactions, highlighting opportunities to expand on existing research

2.1 Analysis of Environment Type

A total of 25 recent highly relevant studies were reviewed and discussed in detail to examine the experimental environments used in prior ROS2 and service robot research. The main objective was to examine the types of experimental environments used in previous studies such as simulations, hybrid or live deployments. Table 1 summarizes the distribution of experimental environments across the reviewed studies.

Simulation based experiments were the most adopted approach, with 13 out of 25 studies conducted entirely in simulation. Tools such as Gazebo and Webots, alongside virtual machines running Ubuntu, were employed for

fully virtual testing and system modeling. The prevalence of simulation reflects its practicality due to lower costs, reduced logistical complexity, and the ability to safely conduct experiments without disrupting operational service robots. Simulation also allows the construction of diverse scenarios, including potential attack vectors and operational conditions, enabling systematic testing that closely mirrors real-world interactions [10]. This controlled approach ensures repeatable and consistent results, providing confidence in the findings and supporting their relevance to operational service robots. 5 out of 25 studies adopted a hybrid approach, combining simulation with validation on mobile robots such as TurtleBot3 and MiR100. This method balances the safety and flexibility of simulation with the realism of physical testing, addressing some limitations of purely simulated environments. 7 studies performed experiments entirely on physical robots, including UR5 robotic arms and Unitree G1 humanoids, or conducted real-world usability evaluations and security audits. While real-world testing provides high-fidelity insights, it is often limited by high costs, complex integrations, and the potential for disrupting operational service robots [9].

Table 1 : Frequency (out of 25) of the most recently reviewed literature based on environment type.

Environment Type	Frequency
Simulation	13
Hybrid	5
Physical robot	7

Given that simulation is the most widely adopted and practical experimental environment, this study focuses on simulation-based experiments to assess ROS2 vulnerabilities using reconnaissance i.e. active or passive information gathering. By focusing on simulation, this study can construct multiple scenarios yet independent, systematically test ROS2-based systems in a controlled, repeatable, realistic, and safe manner.

2.2 Analysis of DDS suppliers

This section examines the most widely used DDS middleware in ROS2 deployments as elaborated in the latest most relevant literature. Figure 1 summarizes the middleware distribution across the reviewed literature. The analysis focuses on identifying the DDS middleware used in each study, as middleware selection directly influences communication behavior and security characteristics in ROS-based systems. Out of the 25 studies, 9 explicitly stated the use of the default DDS middleware provided by ROS2. These studies primarily focused on system performance and communication behavior without modifying the default middleware configuration. Since the default middleware is widely adopted and explicitly mentioned in previous studies, minimal additional setup is required, making it a practical choice for this study. Consequently, we proceeded with the default Fast DDS implementation. In 13 of the 25 studies, the DDS middleware was either not treated as a primary variable or was not explicitly mentioned, often marked as not applicable (N/A). This highlights a gap in literature, suggesting that middleware selection is frequently overlooked and security is not the primary focus, as most studies concentrate on application-level aspects of the robot. Only one study specifically analyzed Cyclone DDS, indicating limited attention to alternative DDS implementations, while two studies were based on legacy ROS1, which does not natively support DDS.

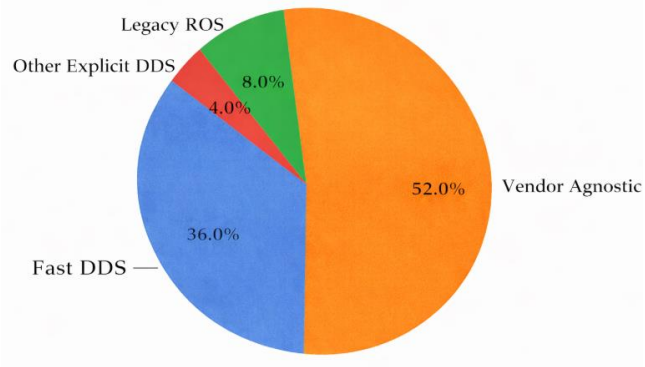


Figure 1: Frequency (in % out of 25) of the most recently reviewed literature based on Middleware distribution.

Fast DDS not only requires minimal configuration but also provides a standardized and widely supported environment ensuring consistency with existing studies. According to [11], it facilitates a focus on security analysis and scenario development without altering underlying communication mechanisms. Overall, there is a clear research gap in the explicit analysis and comparison of DDS middleware implementations in ROS2, particularly regarding security and network communication properties. Therefore, this study leverages the use of Fast DDS, a widely adopted and well documented middleware, while focusing on addressing security concerns that have been largely overlooked.

2.3 Relevant Studies related to Network Analysis

In several studies [12], [13], and [14], network analysis tools such as Wireshark were employed to capture and analyse data packets, including Address Resolution Protocol (ARP), Dynamic Host Configuration Protocol (DHCP), Domain Name System (DNS), Multicast DNS (mDNS), as well as traffic related to the Real-Time Publish–Subscribe (RTPS) protocol. These works demonstrated the effectiveness of network-level monitoring in exposing communication behaviors within ROS-based systems and highlighted

interoperability considerations at the middleware layer. However, their analysis largely remained at the level of traffic observation and protocol identification, without extending to an evaluation of how such observable communications could be exploited or how they might affect robots during active operation. For instance, the potential impact of network-level vulnerabilities on robots performing active tasks such as sensing, navigation, or service delivery was not investigated. In particular, the implications of reconnaissance and passive observation attacks on robots operating over shared or public networks remain unexplored.

Furthermore, a common pattern emerges i.e. regardless of the robot's specific application, whether collecting environmental data in a corridor or delivering goods within a neighborhood the underlying system architecture remains predominantly consistent. This architectural similarity implies comparable attack surfaces, especially at the network and middleware layers. Nevertheless, prior research has only partially examined these shared vulnerabilities, stopping short of evaluating their significance in operational service robot deployments. Accordingly, a representative network-based reconnaissance scenario involving a service robot system is included in the methodology section to contextualize the discussion. Notably, no prior study has systematically quantified the specific types of sensitive operational data that can be passively extracted during an active reconnaissance phase targeting a ROS2-based service robot.

3 Methodology

The main goal of this study is to evaluate network level vulnerabilities in ROS2 based service robots by combining middleware level monitoring with operational use-case validation. Figure 2 illustrates the overall methodology, designed to address the critical gaps identified in sections 2.1, 2.2 and 2.3. The methodology is structured into four main steps as described below. Step 3.1 describes the experimental environment and the tools required to conduct the study. Step 3.2 focuses on standard reconnaissance scenario, detailing the tools used and the validation process to determine the type of observational data that can be collected. Building on this, Step 3.3 applies the approach to a specific use case i.e. robot position monitoring in ROS2 using Fast DDS Spy to assess how the observations translate into practical insights. Finally, Step 3.4 synthesizes and analyzes the additional operational data collected across the experiments, highlighting the implications for system security and exposing potential attack surfaces. Together, these steps form a cohesive framework that combines controlled experimentation, scenario-based evaluation, and operational analysis, enabling a comprehensive assessment of network-level behaviors in ROS2 service robots.

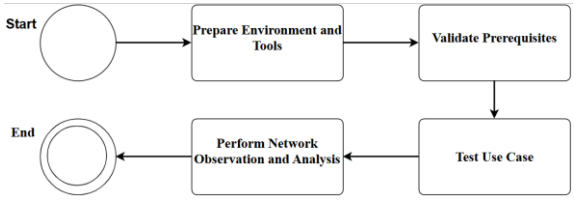


Figure 2: Methodology Diagram

3.1 Prepare Environment and Tools

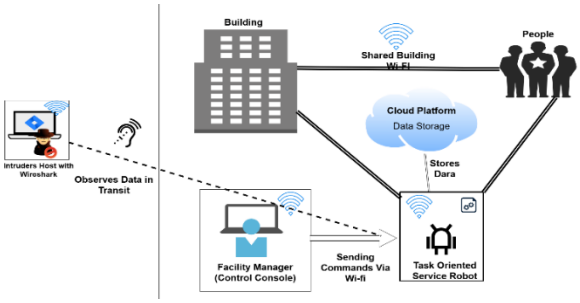
The experimental setup used Ubuntu 24.04 LTS as the host operating system, ROS2 humble as the robotic framework. To analyze communication at the middleware and network layers, multiple diagnostic tools were employed. Fast DDS spy was used on the Ubuntu system to monitor and inspect Data Distribution Service (DDS) communication between ROS2 nodes, enabling visibility into topic discovery and messages exchange at the middleware level. To capture those packets, a software tool called wireshark was executed on a separate windows-based host to capture and analyze network traffic across different machines, supporting cross-host packet-level inspection. For simulation-based evaluation and validation, the Webots 3D robot simulator was used to provide a physics-based virtual environment for modeling, programming, and testing robotic systems. In addition, Turtlebot3 platform was utilized to asses navigation, sensing, and control behaviors under realistic operational conditions. Together, these tools provided a practical setup for simulation and analysis within ROS2 systems.

3.2 Validate Prerequisites

Figure 3 depicts a task-oriented service robot operating within an open wireless network (e.g., Wi-Fi), where it communicates bidirectionally with human operators and smart building infrastructure to support routine operational functions. Such communication may include status updates, sensor data exchange, and control or coordination messages required for normal robot operation. The figure also introduces a potential intruder (left) who gains physical access to the building premises and comes into proximity of the service robot. Without interacting directly with the robot or its control systems, the intruder passively monitors the network by running standard diagnostic tools, such as Wireshark, on the same wireless network. Through this passive observation, the intruder can capture and inspect network traffic associated with the robot's ongoing operations, thereby gaining visibility into various forms of operational and communication data. For the purposes of this study, the attacker is assumed to be an external, unprivileged individual with no prior knowledge of the target system's

architecture or credentials. The attacker is assumed to have physical proximity to the shared wireless network and is equipped only with standard, publicly available diagnostic tools. No active interaction with the robot or its control systems is assumed, confining the threat model to passive network observation consistent with a reconnaissance-phase adversary.

Figure 3: Overview of the network-based reconnaissance



targeting of a task oriented service robot operating in built environment

The scenario depicted in Figure 3 is validated using a controlled experimental environment with a limited set of tools. The validation setup consisted of an Ubuntu 24.04 LTS system running ROS2 with the default DDS middleware, Fast DDS (Fast RTPS), complemented by a Windows host system running Wireshark for capturing and analyzing network traffic. This setup enables observation of operational communications in a controlled manner while reflecting the reconnaissance scenario introduced earlier. Three terminal sessions were opened within the Ubuntu environment. The first session was used to source the ROS2 environment, while the second and third sessions were used to execute ROS2 publisher and subscriber nodes, respectively. ROS2 communication was generated. The execution of these nodes resulted in continuous RTPS communication over user datagram protocol (UDP) using the Fast DDS middleware. Simultaneously, Wireshark was used to capture network traffic on the active network interface, where RTPS packets corresponding to ROS2 communication were successfully observed and identified. This validation confirms that ROS2 DDS based communication can be passively captured and analyzed at the network level using standard packet inspection tools. That is, RTPS traffic generated by ROS2 nodes was captured using Wireshark on the host system by monitoring the shared network interface of the virtualized Ubuntu environment.

Figure 4 illustrates a Wireshark capture of RTPS traffic generated by a ROS2 system using the Fast DDS middleware. The capture shows multiple RTPS packets exchanged over UDP between a ROS2 participant and a multicast destination address. The packets are identified as RTPS data messages, including INFO_TS and DATA sub messages, which are characteristic of DDS based communication. For example, this RTPS packet exchange corresponds to a service robot operating in built environment publishing laser scan data on /scan (similarly odometry and velocity commands).

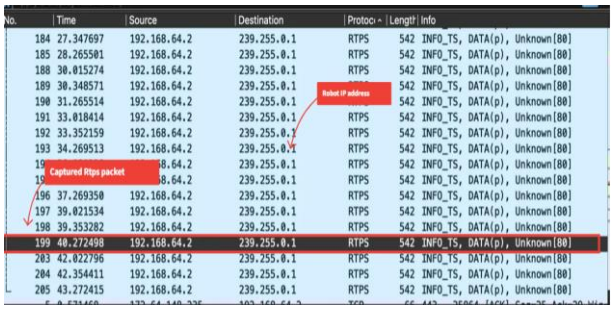


Figure 4: Captured RTPS traffic from a ROS2 system observed using Wireshark

The source IP address (192.168.1.38) corresponds to a ROS2 node within the system, while the destination address (239.255.0.1) indicates multicast communication used for DDS participant discovery and data dissemination. The transport layer is UDP, with the destination port set to 7400, which is the default port range used by DDS/RTPS. The packet lengths range between approximately 500 to 600 bytes, reflecting periodic data exchange between ROS2 nodes. For instance, DDS multicast communication compromises the robot's real-time position and sensor state (e.g., door access). The consistent timing and repetition of packets demonstrate continuous publisher–subscriber communication, confirming active ROS2 node interaction over the network.

From the captured RTPS traffic, several key reconnaissance insights were obtained through active network monitoring. The analysis revealed critical communication details, including the source and destination IP addresses, as well as the corresponding source and destination ports used by ROS2 nodes. These observations expose the internal network structure and confirm the use of multicast-based communication for data distribution [15]. Furthermore, the protocol and packet characteristics allows the identification of the DDS middleware in use, which in this setup was eProsima Fast DDS. In addition, application layer information embedded withing the RTPS traffic discloses the username of the source system account from which the ros2 nodes were executed. Interestingly, these findings also demonstrate that, even without active interaction, an external observer can extract sensitive system and operational details during the reconnaissance phase, highlighting the exposure present when DDS communication is left unencrypted. Building upon the observed and validated ability to passively capture RTPS traffic, the subsequent phase of this study examines what

additional operational information can be obtained during reconnaissance usingaxz standard tools and configurations.

3.3 Test Use Case

Robot position monitoring in ROS2 via Fast DDS Spy is presented as a foundational and representative monitoring scenario for evaluating data exposure in DDS-based robotic systems. Access to a robot's positional data is critical because position information underpins core robotic functions such as navigation, task execution, collision avoidance, and interaction with the surrounding environment.

The base platform used was ROS2 Humble, with a custom python publisher node (publisher.py) that generates random robot positions at one second interval. The published topic was rt/robot_position with advanced features. The tools used in this experiment include: 1) ROS2 humble, serving as the foundational operating system 2) Fast DDS Spy with version 1.4.0 , a command line introspection tool developed by eProsima for discovering DDS entities such as participants, topics, writers, and readers, 3) Wireshark, a network traffic analyser utilized to capture RTPS packets, 4) Webots Simulator, and 5) TurtleBot3, both serving as robot simulators for generating motion data. The procedure for conducting this experiment is as follows: 1) Publisher Setup: A Python based ROS2 node was developed within the robot_position_publisher package to publish three dimensional coordinates (x, y, z) at a fixed frequency of 1 Hz on the rt/robot_position topic. The node generated randomized positional data to simulate real time robot movement, enabling subsequent monitoring and analysis. 2) ROS2 Workspace Setup: The ROS2 workspace was sourced and built using colcon build, ensuring that the robot_position_publisher package and its associated nodes were correctly recognized and accessible within the ROS2 environment. This setup facilitated seamless execution and integration of the publisher node into the DDS based communication network.

Continuous access to this data provides deep insight into a robot's movement patterns or operational behaviour. By observing positional data over time, an external entity can track where a robot is moving, infer routines, predict future locations, or identify areas of operational importance. This capability is particularly relevant for autonomous service robots operating across multiple locations, including smart buildings, hospitals, or industrial facilities, where positional awareness directly correlates with sensitive operational workflows. For example, during building audits or patrol operations, knowledge of robot trajectories may inadvertently reveal restricted zones, high-value assets, or confidential activities [5][16]. If such information is exposed without appropriate protection, it can enable targeted security, privacy, or safety-related attacks. In this study, robot position data is deliberately selected as a baseline DDS data stream to demonstrate non-intrusive monitoring within a ROS2 environment. The experiment focuses on capturing and analysing position updates published on DDS topics to verify real-time data flow, timing behaviour, or observability without interfering with system operation. While the implementation monitors three-dimensional coordinates (x, y, z) as a proof of concept, the same monitoring methodology is directly applicable to other operational DDS topics, including sensor readings, task states, navigation goals, or interactions with smart building infrastructure. As such, this scenario establishes a reusable framework for analyzing DDS-level data exposure beyond positional information alone.

3.4 Perform Network Observation and Analysis

The following observations were recorded, and the results are illustrated in the Figure 5 below and are explained in detail as follows.

Figure 5: Participants recorded by Fast DDS Spy

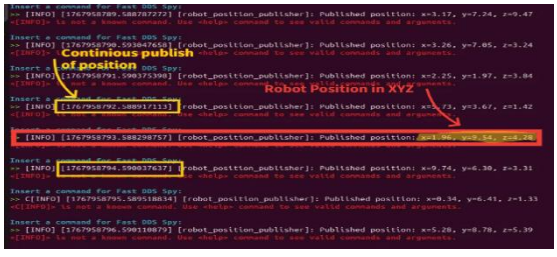
1) Participants: In the context of DDS and ROS2, a participant refers to a robot or software application actively engaged in network communication. Each participant is associated with a Globally Unique Identifier (GUID), which serves as a digital identifier to distinguish individual nodes or applications. For instance, a participant may have an ID such as 01.0f.ae.c2.03.75.b4.90.00.00.00.00, uniquely identifying that specific robot or program within the DDS domain.

2) Topics: Each line represents a topic discovered on the DDS network. rt/robot_position is the topic to which the Python publisher was publishing simulated robot positions, with type Point (from geometry_msgs). (1|0) indicates one DataWriter publishing while no DataReaders are subscribed. [0.780132 Hz] indicates the publishing frequency. Topics such as rt/parameter_events and rt/rosout are standard ROS2 topics for parameter updates and log messages.As shown in Figure 6, Fast DDS Spy monitored the DDS network as a passive introspection tool, discovering active

participants, topics, data writers, and data readers without impacting running nodes. This confirmed that the rt/robot_position topic was broadcasting successfully at 0.78 Hz.Analysis of the topics and participants highlights potential cybersecurity implications. For example, an inspection robot continuously collecting position and environmental data in a corridor could be passively observed by an external party, revealing the robot's location, movement patterns, planned paths, and sensor feeds without interfering with its normal operation. This study focuses on read-only access to evaluate how much operational detail can be inferred; data modification and write access are reserved for future research. This setup demonstrates the capability of Fast DDS Spy to audit ROS2 systems at the DDS middleware level, emphasizing the importance of securing communication infrastructure

3.4.1 Key Inferences

While the observability of unencrypted DDS traffic is broadly acknowledged in ROS2 documentation, this study advances beyond this general observation by systematically identifying and validating eight specific categories of sensitive operational data extractable through passive reconnaissance, demonstrating the practical extent of exposure in a realistic service robot deployment scenario. Using basic reconnaissance tools, it was possible to successfully observe and capture 8 forms of sensitive operational data from the ROS2 system. This included source and destination IP addresses, source and destination ports, user identity associated with the source system, the underlying DDS middleware in use, as well as DDS participants, ROS2 nodes, and active publisher nodes. These findings demonstrate that even limited, non-intrusive tools can reveal critical system and communication details without requiring authentication or direct interaction with the target robot. Such exposure highlights a significant attack surface, where reconnaissance alone can serve as an effective entry point for subsequent attack phases, including command-and-control (C2) based attacks.



4 Conclusions, Future work and Limitations

This study demonstrates that effective reconnaissance of a ROS2-based robotic system is feasible using a minimal set of passive, non-intrusive tools when the attacker is positioned within the same network domain. By deploying Fast DDS Spy and Wireshark on a shared network, it was possible to discover DDS participants, enumerate active topics, and observe real-time robot telemetry without modifying the target system or requiring authentication. Through this process, eight distinct types of operational data were identified, including positional information transmitted over the rt/robot_position topic. The conclusions are strictly confined to the reconnaissance phase; references to subsequent attack phases such as C2, spoofing, and man-in-the-middle attacks are intended solely to contextualize reconnaissance as a precursor and do not form part of the claims of this study. Future work will extend beyond reconnaissance to evaluate C2 feasibility, focusing on correlating velocity and positional data to reconstruct full motion trajectories in real time, enabling a deeper assessment of integrity and availability risks [18][19]. Further research will explore active attack techniques including man-in-the-middle attacks, message replay, and topic spoofing, as well as mitigation strategies such as DDS-Security, network segmentation, and secure ROS2 configurations. Practitioners are advised to enable DDS-Security plugins, restrict DDS domain IDs, and isolate robotic systems from shared networks as immediate best practices. While Fast DDS Spy proved effective for passive discovery and monitoring, the experiment has several limitations. First, the tool operates only with DDS implementations compatible with eProsima Fast DDS and relies on default ROS2 middleware configurations. The experiments were conducted exclusively using Fast DDS, which was selected based on its status as the default ROS2 middleware and its prevalence in the reviewed literature. Whether alternative DDS implementations such as Cyclone DDS exhibit different reconnaissance behaviors under similar conditions remains an open question and is identified as a direction for future investigation. Furthermore, the experimental scope was intentionally limited to default ROS2 configurations, reflecting the most commonly encountered real-world deployment conditions. A comparative evaluation between default and security-enabled DDS configurations is recognized as an important direction and is explicitly identified as part of the planned future work.Systems using alternative DDS vendors, customized discovery settings, or secured DDS (DDS Security) profiles may not be fully observable using this approach[20]. Second, Fast DDS Spy is primarily an introspection tool and does not natively support message manipulation or injection, limiting its use to reconnaissance rather than active exploitation. The experiment also focused on positional data only and did not evaluate the impact of QoS policies, encryption, authentication, or access control mechanisms

that may be present in hardened deployments. Finally, the experimental setup was conducted in a controlled environment, and real-world conditions such as packet loss, network segmentation, or intrusion detection systems were not considered. Performance-related metrics such as data rates and traffic patterns fall outside the scope of this reconnaissance-focused study and are reserved for future work.

5 References

[1] Mayoral-Vilches, V., Makris, A., & Finisterre, K. (2025). Cybersecurity AI: Humanoid Robots as Attack Vectors, https://doi.org/10.48550/arXiv.2509.14139.

[2] Mantha, B. R. K., Menassa, C. C., & Kamat, V. R. (2016). Ambient data collection in indoor building environments using mobile robots. *33rd ISARC* (pp. 428–436). https://doi.org/10.22260/isarc2016/0052

[3] Mantha, B. R., Garcia de Soto., B., Menassa C. C., and Kamat V. R., (2020) "Robots in Indoor and Outdoor Environments", *Taylor and Francis*, DOI: https://doi.org/10.1201/9780429398100.

[4] Kim, J., Smereka, J. M., Cheung, C., Nepal, S., & Grobler, M. (2018). Security and Performance Considerations in ROS 2: A Balancing Act https://doi.org/10.48550/arXiv.1809.09566.

[5] DiLuoffo, V., Michalson, W. R., & Sunar, B. (2019). Credential Masquerading and OpenSSL Spy: Exploring ROS 2 using DDS security. https://doi.org/10.48550/arXiv.1904.09179.

[6] Aartsen, M., Banga, K., Talko, K., Touw, D., Wisman, B., Meinsma, D., & Bjorkqvist, M. (2022). Analyzing Interoperability and Security Overhead of ROS2 DDS Middleware. 30th MED, 976–981. https://doi.org/10.1109/MED54222.2022.9837282.

[7] Rivera, S., & State, R. (2021). Securing Robots: An Integrated Approach for Security Challenges and Monitoring for the Robotic Operating System (ROS).

[8] Bistrom, D., Westerlund, M., Duncan, B., & Jaatun, M. G. (2022). Privacy and security challenges for autonomous agents: A study of two social humanoid service robots. IEEE CloudCom, 230–37, https://doi.org/10.1109/CloudCom55334.2022.00040.

[9] Botta, A., Rotbei, S., Zinno, S., & Ventre, G. (2023). Cyber security of robots: A comprehensive survey. Intelligent Systems with Applications, 18, 200237. https://doi.org/10.1016/j.iswa.2023.200237

[10] Deng, G., Xu, G., Zhou, Y., Zhang, T., & Liu, Y. (2022). On the (In)Security of Secure ROS2. ACM SIGSAC, 739–753. https://doi.org/10.1145/3548606.3560681.

[11] Patel, Y., & Rughani, P. H. (2023). qcROS2: An Architecture to Secure Network Communication of ROS2-based Environment. 8th ICRAE, 253–258. https://doi.org/10.1109/ICRAE59816.2023.10458571.

[12] Patil, U., Gunasekaran, A., Bobba, R., & Abbas, H. (2024). ROS2-Based Simulation Framework for Cyberphysical Security Analysis of UAVs. https://doi.org/10.48550/arXiv.2410.03971

[13] Goerke, N., Timmermann, D., & Baumgart, I. (2021). Who Controls Your Robot? An Evaluation of ROS Security Mechanisms. 7th ICARA, https://doi.org/10.1109/ICARA51699.2021.9376468.

[14] Kim, J., Smereka, J. M., Cheung, C., Nepal, S., & Grobler, M. (2018). Security and Performance Considerations in ROS 2: A Balancing Act. https://doi.org/10.48550/arXiv.1809.09566.

[15] Fernandez, I. A., Neupane, S., Chakraborty, T., Mitra, S., Mittal, S., Pillai, N., Chen, J., & Rahimi, S. (2024). A Survey on Privacy Attacks Against Digital Twin Systems in AI-Robotics. IEEE CIC, https://doi.org/10.1109/CIC62241.2024.00019.

[16] Mayoral-Vilches, V., White, R., Caiazza, G., & Arguedas, M. (2022). SROS2: Usable Cyber Security Tools for ROS 2. 2022 IEEE/RSJ IROS, https://doi.org/10.1109/IROS47612.2022.9982129.

[17] Oruma, S. O., Sánchez-Gordón, M., Colomo-Palacios, R., Gkioulos, V., & Hansen, J. K. (2022). A Systematic Review on Social Robots in Public Spaces: Threat Landscape and Attack Surface. Computers, 11(12), 181. https://doi.org/10.3390/computers1112018. 1

[18] Patel, Y., Rughani, P. H., & Desai, D. (2022). Network Forensic Investigation of Collaborative Robots: A Case Study. 2022 7th ICMERR, 51–54. https://doi.org/10.1109/ICMERR56497.2022.10097787

[19] Rivera, S., Lagraa, S., Iannillo, A. K., & State, R. (2019). Auto-Encoding Robot State Against Sensor Spoofing Attacks. 2019 IEEE ISSREW, 252–257. https://doi.org/10.1109/ISSREW.2019.00080.

[20] Rivera, S., Lagraa, S., Nita-Rotaru, C., Becker, S., & State, R. (2019). ROS-Defender: SDN-Based Security Policy Enforcement for Robotic Applications. 2019 IEEE SPW, 114–119. https://doi.org/10.1109/SPW.2019.00030.