2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

The Case of Metadata Leakage in ROS 2: Fingerprintability, Security Implications, and Internet-Wide Vulnerability Measurements

Fayzah Alshammari, Sam Der, Qi Alfred Chen

University of California, Irvine
Email: {fayzaha, sder, alfchen}@uci.edu

***Abstract*— The Robot Operating System (ROS) is widely adopted in the robotics community, powering applications from self-driving vehicles to industrial automation. ROS 2 utilizes the Data Distribution Service (DDS) middleware for decentralized communication, making it inherently susceptible to reconnaissance and exploitation attacks. Previous research has examined the security implications of DDS implementations but has not systematically distinguished ROS 2 nodes from standalone DDS deployments, a critical distinction that significantly influences the execution and outcome of cyberattacks. This paper presents the first systematic fingerprinting framework designed specifically for ROS 2, demonstrating how DDS-based metadata leakage can facilitate precise identification and targeted exploitation of robotic systems. Through controlled experiments and an Internet-wide scan of DDS deployments, we identify extensive metadata exposure across actively supported ROS 2 implementations. Despite existing security solutions such as Secure ROS 2 (SROS2), deployments using default configurations remain vulnerable, highlighting the need for enhanced metadata obfuscation, stricter network access policies, and deployment of real-time anomaly detection mechanisms to strengthen the security posture of ROS 2 systems.**

I. Introduction

Robotics systems are increasingly deployed in safety-critical and industrial domains, ranging from autonomous vehicles to drones and medical devices. To support these applications, the Robot Operating System 2 (ROS 2) has emerged as a extremely popular robotics framework. At its core, it relies on the Data Distribution Service (DDS), a decentralized publish-subscribe middleware, to handle peer discovery and message exchange between distributed components. DDS in turn uses the Real-Time Publish-Subscribe (RTPS) protocol, which operates over multicast and unicast transport channels to support discovery and communication.

While DDS's decentralization offers flexibility and scalability, it also introduces significant security challenges, particularly in scenarios where nodes are deployed on public or unsecured networks [1, 2]. Prior work [3]–[5] has demonstrated vulnerabilities in DDS, including amplification-based (CWE-406 [6]) denial-of-service attacks and unsafe assumptions in locator trust models. However, these efforts have largely focused only on exploiting these vulnerabilities and have not rigorously differentiated between generic DDS deployments and those running ROS 2.

This distinction is critical. ROS 2 nodes, layered on top of DDS, often expose additional metadata that is not necessarily present in standalone DDS instances. Moreover, ROS 2 deployments are increasingly being used in real-world robotic systems [7] with physical actuation capabilities, which raises the stakes of reconnaissance or exploitation attempts. Despite the availability of security frameworks such as Secure ROS 2 (SROS2) and FogROS2-SGC, many ROS 2 deployments continue to operate with default, insecure configurations. Our Internet-wide scan reveals over 200 exposed DDS-based (including ROS 2) nodes, as detailed in §IV-B.

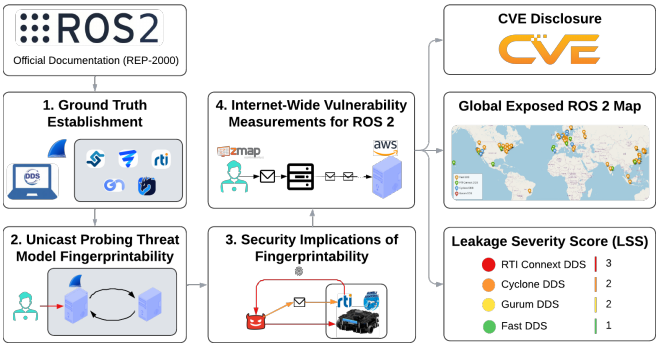

Fig. 1: ROS 2 fingerprintability exploration methodology, from standards documentation (REP-2000) to ground-truth DDS RMW datasets, systematic fingerprinting, targeted exploitation with responsible disclosure, and Internet-wide measurement of real-world deployments.

In this paper, we show that DDS discovery replies themselves introduce a distinct vulnerability class: metadata leakage. With ROS 2, discovery packets routinely expose identifying information, including ROS 2 security enclave identifiers, hostnames, node names, and even usernames in some DDS implementations. This exposure enables fingerprinting, the ability to distinguish ROS 2 nodes from generic DDS services, to infer vendor implementations, and to recover identifiers. Unlike amplification, fingerprinting does not rely on spoofing or reflection: it arises naturally from discovery [1, 2]. This makes it an independent attack vector, grounded in CWE-200 (Exposure of Sensitive Information) [8], CAPEC-312 (Active OS Fingerprinting) [9], and the Reconnaissance tactic of MITRE ATT&CK [10].

We systematically study fingerprintability in ROS 2 deployments and convey its practical security implications as shown in Fig. 1. Using a controlled testbed covering all supported DDS RMWs (Fast DDS, Cyclone DDS, RTI Connext DDS, and Gurum DDS), we build the first dataset of leaked metadata fields and evaluate their utility for classification and exploitation. We propose quantitative metrics, including

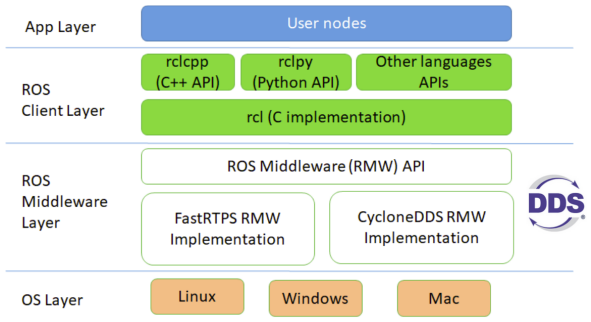

Fig. 2: ROS 2's layered architecture highlighting the middleware ecosystem. The ROS Middleware (RMW) API abstracts over multiple DDS implementations, allowing ROS 2 to operate independently of the underlying vendor [11].

precision, packets to identification, and bytes to identification in §III-B. Further, in §III-C, we demonstrate how leaked metadata enables targeted exploitation through a TurtleBot3 robot with responsible vendor disclosure.

Finally, we extend our analysis to the global scale. Using a custom scanner and collector infrastructure, we conduct the first Internet-wide measurement of ROS 2 fingerprintability, validating that metadata leakage affects real deployments worldwide. We observe vendor-specific differences in leakage patterns and confirm that sensitive identifiers are in fact visible in the wild, highlighting a systemic risk for robotics security. Our key contributions are as follows:

- **New attack vector demonstration.** We identify metadata leakage in DDS as an independent fingerprinting threat to ROS 2 based on confidentiality, not availability.
- **A novel ROS 2 systematic fingerprinting.** We build the first fingerprinting dataset for ROS 2, and define quantitative metrics to measure fingerprintability.
- **Exploitation and disclosure.** We demonstrate how leaked metadata enables targeted exploitation, validated on a physical TurtleBot3 robot, and have responsibly reported a previously unrecognized vulnerability to a DDS implementation vendor.
- **Internet-wide vulnerability landscape for ROS 2.** We present the first Internet-wide measurement and fingerprinting of publicly exposed ROS 2 systems, revealing real-world exposure across vendors and regions.

Code/data release. The code/data from this study are available on our project website **`https://sites.google.com/view/secure-safe-ai/roboscan2`**.

II. Background and Threat Model

A. ROS 2 Middleware Ecosystem

While DDS can be used independently as a publish-subscribe middleware, it lacks a robotics package ecosystem, requiring developers to manually implement node creation, message types, and communication logic [1]. ROS 2 addresses this with high-level client libraries, reusable packages, and the ROS Middleware (RMW) abstraction layer. As shown in Fig. 2, developers interact with the application and client libraries, while RMW interacts with DDS. This middleware-agnostic design allows developers to swap DDS implementations without modifying application code and benchmark options based on system needs. Our focus in this paper is limited to the four DDS implementations supported by the current LTS distributions, Humble and Jazzy: Fast DDS, Cyclone DDS, Connext DDS, and Gurum DDS.z'

To handle differences in DDS security feature implementations, SROS2 [12] was created as an abstraction, such as authentication through a unified ROS interface. SROS2 secures internal ROS 2 communication, while tools like FogROS2-SGC provide a complementary approach to protect traffic between distributed nodes at a global scale [13].

B. DDS and RTPS Discovery

DDS implements a publish-subscribe communication model using the RTPS protocol as the standard wire format. Node discovery is a core component of DDS, enabling different components of a system to automatically locate and communicate with each other via multicast. When joining a network, a node will announce its presence via a multicast address by sending RTPS packets that contain submessages with different types of information. `DATA(p)` submessages, for example, include protocol identifier (PID) parameters, some of which are used for instructing nodes where to send discovery packets (e.g., `PID_DEFAULT_UNICAST_LOCATOR`). Other important submessages include `HEARTBEAT` for status checks, and `ACKNACK` packets for data synchronization between nodes. These packets allow nodes to establish unicast communication with other nodes to communicate over shared topics.

C. Related Work

Network Security in ROS and DDS. The security of DDS has been primarily examined through the lens of availability. Maggi et al. [3] showed that participant locators are blindly trusted and nodes reply until acknowledgment, enabling reflection and amplification attacks. We focus on confidentiality risks (CWE-200 [8]) from excessive metadata leakage in ROS 2 DDS discovery, leading to fingerprinting and targeted exploitation rather than general disruption.

Network Reconnaissance and Fingerprinting. Prior methodologies have typically relied on protocol-based signatures [14] or metadata extraction [15], but have not been adapted for distinguishing ROS 2 nodes from standalone DDS deployments or other DDS-based frameworks. While the Robot Hacking Manual [16] uses crafted RTPS packets for triggering different DDS vulnerabilities on ROS 2 nodes, it does not analyze the packets that are reflected for fingerprinting purposes. In addition, the ROS 2 experiments discussed are performed on ROS 2 Foxy, which is no longer supported. Other works on scanning include DeMarinis et al. [17] and Chen et al. [4], who scanned the Internet for publicly exposed ROS 1 instances. While the centralized ROS 1 master node grants attackers immediate, unrestricted topic access, ROS 2's decentralized DDS architecture inhibits this

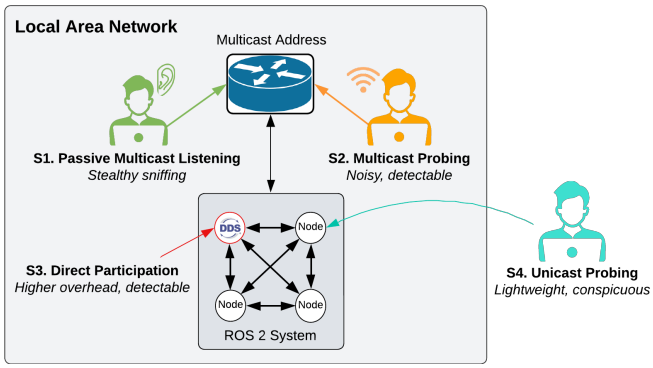

Fig. 3: Threat model for ROS 2 fingerprintability. We consider four adversary scenarios, differing in network position, stealth, and amount of metadata exposed.

direct exploitation. Our findings reveal that implementation-specific metadata leakage (e.g., usernames) downgrades ROS 2's architectural improvements. While both studies confirm that default insecure configurations persist, our work uniquely identifies DDS vendor-specific metadata leakage as a new vulnerability class through ROS 2 classification.

ROS 2 Exploitation. Previous research [3, 18] has demonstrated the feasibility of DoS attacks on DDS-based nodes, but lack a structured attack chain specifically targeting ROS 2 systems. Unlike standalone DDS, ROS 2 introduces abstractions, metadata, and a popular package repository that create unique attack surfaces, necessitating a structured fingerprinting approach for exploiting these vectors.

D. Threat Model

Attacker Goals. We consider the adversary's objectives to be (1) reconnaissance: distinguishing ROS 2 from standalone DDS, (2) fingerprinting: extracting metadata such as enclaves, hostnames, and vendor identity, and (3) exploitation: leveraging metadata for elevated system access.

Attack Scenarios. We categorize attack scenarios by their network position and behavior in Fig. 3. *Scenario S1 (Passive Multicast Listening):* the attacker tracks traffic to multicast addresses using tools like Wireshark. This method provides low visibility and continuous reconnaissance but is restricted to the LAN due to multicast scope. *Scenario S2 (Multicast Probing):* the adversary broadcasts multicast announcements, prompting responses for rapid enumeration, but also remains within the LAN and is highly visible since nodes become aware of its existence [2, 16]. *Scenario S3 (Direct Participation):* the attacker runs their own node, collecting metadata on topics and participant identifiers [19], but faces higher operational overhead and detectability as a new participant [3]. Lastly, *Scenario S4 (Unicast Probing):* the attacker crafts RTPS packets targeted at a specific node's IP address and common discovery ports [3]. This attacker quickly receives responses from the system without deploying a single node, which is efficient when scanning large external IP spaces, but may trigger intrusion detection systems. This paper primarily evaluates fingerprintability under S3 and S4, extending the latter to an Internet-wide scale in §IV.

Deployment Relevance. The exposure of DDS services to public networks is a widely documented issue. RTPS discovery employs deterministic port mapping, binding predictable UDP ports (e.g., 7400) for automatic peer discovery [3]. Without network isolation, these RTPS services become externally identifiable. Our Internet-wide scan (§IV) identified over 200 DDS hosts, including deployments on major cloud providers (AWS, GCP, and Azure), university research networks, and industrial infrastructures. Such exposure often results from cloud robotics architectures connecting distributed nodes across WANs [13] with permissive firewall settings, or systems lacking network segmentation.

III. ROS 2 Fingerprintability and Security Implication Demonstration

Fig. 1 shows our analysis pipeline. We utilize Scenario S3 to examine DDS and non-DDS implementations supported by ROS 2 Humble and Jazzy [20] and build an initial ground truth using metadata leaked by each one. We then apply Scenario S4, discovering a potential CVE and a path for targeted exploitation in the process. Finally, we expand S4 to a global, large-scale scan to measure public ROS 2 exposure.

A. Ground Truth Establishment

Experimental methodology. We establish a ground truth through controlled laboratory experiments on the four DDS implementations and one non-DDS middleware, Zenoh, supported by ROS 2. We evaluate each one in isolation using "talker" and "listener" example nodes that generate authentic RTPS discovery traffic under typical operating conditions. Traffic was captured and analyzed with Wireshark, where we find metadata including ROS 2 strings, node names, hostnames, and usernames. The patterns discovered at this stage provide a reliable baseline for subsequent fingerprinting.

Experimental setup. To trigger the normal join process, we set up a local, controlled environment with two Ubuntu virtual machines each for ROS 2 Humble and ROS 2 Jazzy. Following documentation, we install Ubuntu 22.04 for Humble and Ubuntu 24.04 for Jazzy, along with the same talker and listener nodes. The talker is started on one VM before starting up the listener on the other in order to observe the complete discovery process. We run Wireshark on the talker VM to capture and analyze discovery packets.

Results. The packets we capture follow the discovery process as described in §II, with the exact number of packets represented in Table I. Notable ROS 2 metadata from the payloads during the discovery process is displayed in the Metadata Leaked column of Table II. We find that the string `enclave` is a specific indicator of ROS 2 settings in the payloads of `DATA(p)` submessages as opposed to standalone DDS, which originates from ROS 2's documented security enclave design [21]. While the DDS Security standard outlines core security features such as authentication, access control, and encryption, the notion of an *enclave* with explicitly structured directories, permissions, and metadata is injected into `DATA(p)` submessages by the ROS 2 RMW layer, making it unique to ROS 2 and not part of the DDS

TABLE I: Response patterns for different ROS 2 and DDS combinations. When comparing a ROS 2 node to a DDS node using the same implementation, the number of submessages received in the response is the same.

Type	Fast	Cyclone	RTI Connext	Gurum
HEARTBEAT	1	43	1	0
ACKNACK	8	16	1	1
DATA(p)	0	Continuous	Continuous	1
Total	9	N/A	N/A	2

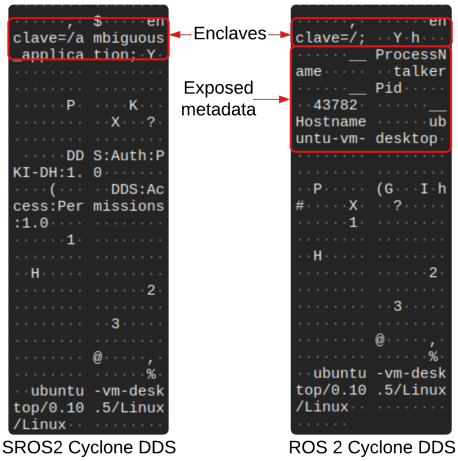

Fig. 4: Wireshark captures of SROS2 (left) and ROS 2 (right) RTPS packets using Cyclone DDS. Metadata associated with unencrypted ROS 2 Cyclone DDS RTPS packets include node names and hostnames, but those are obscured when using SROS2, save for the hostname on the bottom.

standard by design. Our controlled experiments Table II further confirm this: the `enclave` string consistently appears only in ROS 2 nodes and is absent from standalone DDS nodes across all tested implementations. We further verify the specifications of other popular [22] middleware frameworks that rely on DDS besides ROS 2, including the Automotive Open System Architecture (AUTOSAR) for automotive systems and the Open Field Message Bus (OpenFMB) for energy/utility controls, and confirm that none of them have the `enclave` string by design [23, 24].

With enclaves disabled (the default configuration), the value in `DATA(p)` submessages defaults to `/`, resulting in plaintext RTPS packets. Conversely, enabling enclaves through ROS 2 (i.e., SROS2) effectively mitigates metadata leakage by preventing unauthorized interactions and encrypting RTPS packet payloads, underscoring its significance as an accurate fingerprinting mechanism. Official ROS documentation includes instructions on how to encrypt RTPS traffic with enclaves and examine it using tools such as Wireshark [25]. We confirm this functionality in Fig. 4 using Cyclone DDS as the implementation.

The `enclave` appears in Fast DDS, Cyclone DDS, and RTI Connext DDS, which indicates that the string can be used to identify whether a host is running a ROS 2 or a standalone DDS node. Meanwhile, Gurum DDS reveals a similar string `securitycontext` in its own `DATA(p)`

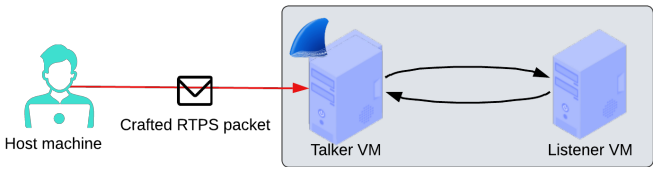

Fig. 5: Fingerprintability experiment setup. The talker and listener VMs share a network, with Wireshark running on the talker. An external attacker sends a crafted RTPS packet to the talker; the captured traffic is forwarded to the collector.

submesasges, and perusal of its RMW layer source code reveals that this string is functionally equivalent to `enclave`, meaning it is also an identifier unique to ROS 2 [26].

Other metadata can reveal important information about the node. For example, ROS 2 RTI Connext DDS `DATA(p)` submessages contain the executable path running the node, which in turn, reveals the ROS 2 distribution being used, since the default installation path for ROS 2 is `/opt/ros/[DISTRIBUTION]/...`. This elevates an adversary's fingerprinting beyond distinguishing ROS 2 from standalone DDS to a much narrower attack vector through vulnerabilities associated with an individual distribution.

We also tested Zenoh in our controlled experiment setup, finding that TCP is used by default in place of RTPS for node communications. As it does not adhere to the DDS standard, it is a different publish-subscribe architecture requiring alternative fingerprinting methodologies. Our focus on the four DDS vendors ensures relevance to today's ROS 2 deployments while maintaining experimental rigor.

B. Fingerprintability in Unicast Probing Threat Model

Experimental methodology. Leveraging the distinct metadata patterns identified in our established ground truth, we perform ROS 2 fingerprinting using the Unicast Probing scenario defined in our threat model in §II. Given a target IP address and a collector endpoint, Algorithm 1 constructs a lightweight RTPS discovery probe that embeds the endpoint into the `PID_DEFAULT_UNICAST_LOCATOR` and `PID_METATRAFFIC_UNICAST_LOCATOR` parameters and transmits it to the target. Because these parameters are used by nodes to determine where to send discovery traffic to, the target will send RTPS packets directly to the collector if the target is running a DDS-based system. Fig. 5 demonstrates the flow of this experiment, which we run more than 30 times on both ROS 2 and standalone DDS nodes to ensure stability in our findings.

Responses received by the collector are subsequently analyzed using Algorithm 2. We search for the PID parameter `PID_USER_DATA` in the `DATA(p)` submessage, which will check for the existence of a security enclave [21].

Experimental setup. With these patterns obtained from the ground truth, we perform similar reconnaissance from outside of the system's network. This requires a different process as it is not possible to send traffic to a multicast address from the outside of a network. Thus, we implement Algorithm 1, focusing on common RTPS ports across the different DDS implementations, and launch the talker and

Algorithm 1: ROS 2 Unicast Probing

Input: Collector IP *c_ip*; collector port *c_port*
Initialization: `RTPS_PORTS` = [7400, 7410, 7411, 7412]
`/* CraftPacket returns RTPS packet with collector IP and port embedded */`
packet ← `CraftPacket`(*c_ip*, *c_port*)
for *port in RTPS_PORTS* **do**
  `/* SendPacket sends crafted packet to ip on port */`
  `SendPacket`(*ip*, *port*, *packet*)
end

Algorithm 2: ROS 2 Fingerprinting

Input: Packet list *packet_list* received from node
Output: Boolean answer `Ans` if node is ROS 2 or `Unknown` if inconclusive
Initialization: `Ans = Unknown`
for *rtps_packet in packet_list* **do**
  if *rtps_packet has* `DATA(p)` *submessage* **then**
    `/* GetPidUserData returns PID_USER_DATA parameter value */`
    payload ⟵ `GetPidUserData`(*submessage*)
    `Ans` ⟵ *bool*(`enclave` in *payload*)
    break
  end
end
Return: `Ans`

listener VMs. From outside the talker-listener network, we send the crafted RTPS packet to the talker node through its public IP address and track the individual packets of its response with Wireshark. We then utilize Algorithm 2 to fingerprint the system using these packets.

Results. We represent our findings in this section through Table II, where we define several metrics: fingerprinting success rate (FSR), the success rate of our fingerprinting methodology in identifying whether the nodes we run are ROS 2 or standalone DDS; packets to identification (PTI), the number of packets examined in a response until we conclude whether a node is ROS 2; and bytes to identification (BTI), PTI in bytes. The FSR is the percentage of detected DDS-based systems Algorithm 2 correctly predicted to be ROS 2 based on the ground truth formed in §III-A.

Surprisingly, the packets we receive in the responses from this experiment deviate from the discovery process described in §II. Although discovery is defined by the DDS standard, the packets that are sent as part of that process is dependent on the DDS vendor, resulting in varying responses between vendors. For example, Cyclone DDS and RTI Connext DDS-based nodes continuously send packets to the collector until the node is stopped, with each packet revealing sensitive system metadata. This includes the process running the node, the system's hostname, and most alarmingly for RTI Connext DDS, the username of the user running the node. Fig. 6 illustrates the metadata in Wireshark packet captures of ROS 2 systems running these implementations. Cyclone DDS takes the longest to identify, needing more than 20 packets before finding a `DATA(p)` submessage (high PTI & BTI). Meanwhile, we observe the `securitycontext`

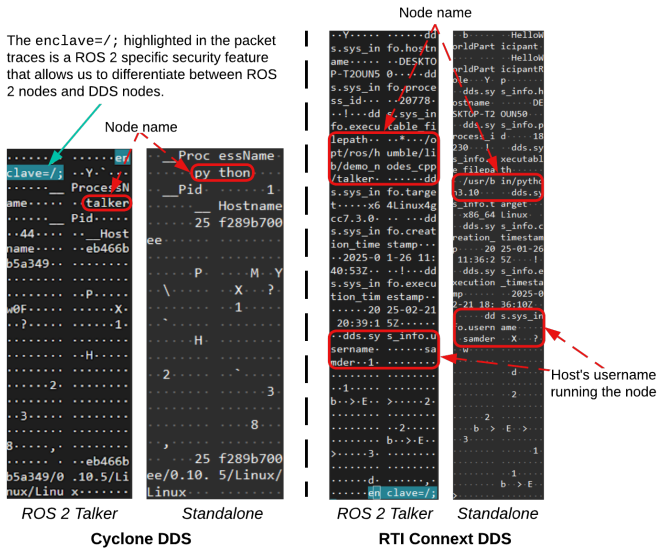

Fig. 6: Wireshark packet traces showing RTPS discovery responses from ROS 2 nodes versus standalone DDS nodes for Cyclone DDS and RTI Connext DDS. `enclave`, highlighted in red, appears only in ROS 2 nodes, revealing a unique marker that enables fingerprinting of ROS 2 deployments.

string being sent collector by Gurum DDS-based ROS 2 nodes and only after examining one packet (low PTI & BTI).

Conversely, Fast DDS does not include `DATA(p)` submessages or expose detailed metadata. Only `HEARTBEAT` and `ACKNACK` submessages are present in its response packets, limiting fingerprinting for this implementation. Its behavior is also both intuitive and expected for discovery [1]. Without sending back properly structured RTPS packets to a Fast DDS node, it will drop attempts at further unicast communication. Indeed, we observed a backoff sending pattern for each of the 30 times we send a packet to the Fast DDS node.

Beyond enclaves, supplementary metadata extracted from these `DATA(p)` submessages include node names, hostnames, and usernames. While these can corroborate fingerprinting findings, they are not robust identifiers on their own because they appear in different forms across DDS implementations. In fact, some do not appear at all depending on the implementation, like Fast DDS. Fast DDS's resistance to fingerprinting reflects an intentional vendor design rather than a methodological limitation. The DDS standard [1] leaves metadata exposure to vendor discretion [2]. Fast DDS omits sensitive fields, whereas other vendors embed richer identifiers, likely to support diagnostics and management, at the cost of increased fingerprintability. While security frameworks like SROS2 mitigate metadata exposure by encrypting RTPS payloads, the root vulnerability lies in DDS implementation designs. RTI Connext DDS embeds usernames and executable paths, while Cyclone DDS exposes node names and hostnames. As Fast DDS demonstrates, functional discovery requires no such metadata, confirming this leakage is an unnecessary information exposure (CWE-200 [8]) that compounds the risks of default insecure configurations.

To summarize these findings, we define a Leakage Severity

TABLE II: Quantitative Comparison of Fingerprinting Effectiveness and Metadata Leakage Severity in ROS 2 DDS Implementations. We evaluate four DDS implementations by their Leakage Severity Score (LSS), which ranks the exploitability of exposed metadata. Metrics include fingerprint success rate (FSR), packets (PTI) and bytes (BTI) to identification, where higher LSS indicates more severe leakage.

Implementation	FSR (%)	PTI (# packets)	BTI (# bytes)	Metadata Leaked	LSS (0-3)
RTI Connext DDS	100%	2	160	Username, executable paths (`/opt/ros/...`), `enclave` identifier	**3 (Critical)**
Cyclone DDS	100%	22	2320	`enclave`, node names (`talker`), hostnames	**2 (Moderate)**
Gurum DDS	100%	2	374	Namespace, `securitycontext`, node names	**2 (Moderate)**
Fast DDS	0%	1	< 100	No `enclave` identifiers	**1 (Minimal)**

Score (LSS), an ordinal ranking of metadata leakage severity. Minimal leakage, akin to the response we received from Fast DDS, has an LSS of 1. Moderate leakage, including node names, hostnames, namespaces, and enclaves have an LSS of 2, since the values that are used in this metadata can provide insight into the system's nodes. Lastly, severe leakage, such as the username and executable pathfrom RTI Connext DDS responses have an LSS of 3, as these paths can reveal whether a node is ROS 2 as well as the ROS distribution being used.

C. Security Implication Demonstration of Fingerprintability

Experimental methodology. Our fingerprinting demonstrates that ROS 2 nodes can be identified and classified, but the critical security impact arises when these leaked fields are mapped into exploitation vectors. Specifically, hostnames and node names reveal the structure of robotic deployments, and usernames create a direct bridge to credential guessing or SSH login attempts. We formalize these pathways by mapping them to established taxonomies (CWE-200, CAPEC-312, and ATT&CK Reconnaissance [8]–[10]), providing a proof-of-concept exploitation chain on a ROS 2 TurtleBot3 robot, and describing the CVE reporting process that arose from our analysis in §III-B.

Experimental setup. Using a TurtleBot3 Waffle Pi, we install ROS 2 Humble with RTI Connext DDS on Ubuntu 22.04 Server following Fig. 7. We create an account with a weak username and password (`admin:admin`), run the bringup commands with this account to start the associated ROS 2 nodes, and publicly expose its IP address along with SSH port 22. The attacker presently knows nothing about the Turtlebot, but can send a crafted RTPS packet to its IP address and receive packets as described in Table I.

Results. When analyzing the response from the TurtleBot3, `DATA(p)` submessages reveal the `admin` username of the user running the ROS 2 node as well as the path of the node's process. As an attacker, the username allows us to launch targeted dictionary or brute-force attacks against the robot's exposed SSH service [27]. By doing this, we can log in via SSH to gain full unauthorized control over the robot. We can then issue commands to halt the robot's mobility or worse, cause serious injury. While default credentials were used in our demonstration, predictable passwords remain prevalent in embedded systems [28] and password reuse is widespread [29]. Moreover, through metadata leakage, attackers can sell exposed IPs on dark web markets [30] and reveal the ROS distribution used by the robot, enabling

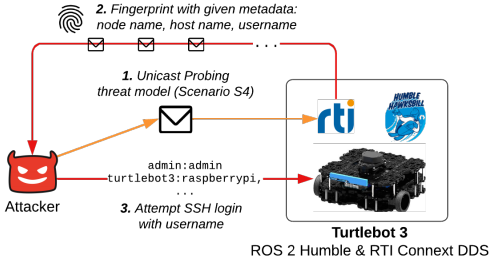

Fig. 7: End-to-end exploitation demonstration on a TurtleBot3 robot running ROS 2 Humble with RTI Connext DDS. The attacker fingerprints the robot and obtains metadata that enables SSH-based intrusion, illustrating how metadata exposure during discovery directly facilitates further attacks.

distribution-specific attacks on these systems [31]. This entire attack chain stems from RTI Connext DDS exposing usernames by default in DATA(p) submessages, which we have reported to RTI as CVE.

Ethical Considerations. We conducted only minimal, rate-limited DDS discovery scans, performed exploitation solely on our own TurtleBot3 robot, and responsibly disclosed the identified vulnerability to the affected vendor.

IV. Internet-Wide Vulnerability Landscape Measurements for ROS 2

A. Measurement Methodology and Setup

To quantify the public exposure of ROS 2 deployments, we conducted an Internet-wide measurement using our fingerprinting methodology in §III-B. Initially, we considered utilizing public search engines, notably Shodan, which allow users to search for devices connected to the Internet. However, analysis of the RTPS packets returned by Shodan revealed that only Cyclone DDS-based systems were being captured, limiting its fingerprinting capabilities. We confirmed this on our own ROS 2 honeypots from Chen et al. [4]: Shodan packets were using the DDSPerf tool used for testing the performance of Cyclone DDS systems. Given these constraints, Shodan proved inadequate for assessing the exposure of different ROS 2 DDS implementations [32].

Consequently, we adapted our fingerprinting methodology to a more rigorous scanning framework [3]. Utilizing an AWS EC2 server as our collector, we send crafted RTPS packets with the `scapy` Python library (Algorithm 1) to every 3.7 billion IPs in the public IPv4 address space, using a pseudorandom permutation of the address space to avoid overwhelming a subnet [15]. We host a public webpage on

TABLE III: We use Algorithm 2 to predict how many nodes found by our scan are ROS 2, excluding our 15 honeypots and unspported DDS implementations. We find 141 nodes, 11 of which were predicted to be ROS 2. The FSR (§III-B) is the percentage of correct predictions based on our ground truth. The Fast DDS and Gurum DDS FSRs are N/A as Fast DDS responses did not include `DATA(p)` submessages and no Gurum DDS systems were detected.

DDS Impl.	Count	Pred. ROS 2	PTI	BTI	FSR
Fast	89	0	1	110	N/A
Cyclone	17	9	22	2922	100%
Connext	35	2	2	160	100%
Gurum	0	0	1	374	N/A
Total	141	11	Average FSR		100%

Fig. 8: Global exposure map of DDS-based systems detected by our Internet-wide scan, colored by implementations supported by ROS 2 Humble and Jazzy. Deployments were found across the Americas, Europe, and Asia, highlighting the real-world attack surface of public ROS 2 systems.

our scanner server to disclose our scanning and provide a contact if one wishes to be excluded from scans. RTPS responses are streamed to our collector endpoint, stored in PCAP files, and systematically searched (Algorithm 2) for ROS 2-specific metadata. This extension enabled comprehensive fingerprinting across DDS implementations, providing robust measurements of real-world ROS 2 node exposure.

B. Results and Findings

Our Internet-wide scanner found more than 200 DDS-based hosts scattered across the IPv4 address space in 3 weeks of scanning at approximately 4000 IPs per second. From these hosts, Algorithm 2 identified 11 ROS 2 hosts, excluding honeypots that we deployed previously. The large number of publicly exposed DDS-based hosts is alarming, considering that approximately the same number of ROS 1 instances were discovered in the scans by Chen et al. [4].

Fig. 8 displays the nodes' locations on a world map colored by their DDS implementation. Table III displays the number of DDS nodes partitioned by their DDS implementation, the number of nodes that were predicted to be ROS 2 by our methodology, and the success rate of our fingerprinting when comparing to our ground truth. Fast DDS makes up the majority of the responses, primarily because it is currently the default implementation for ROS 2. As this scenario is also Unicast Probing, the PTI and BTI remain the same as inTable II. Conversely, we did not receive any responses from Gurum DDS-based systems, which we believe is due to its closed source nature and limited public documentation [33].

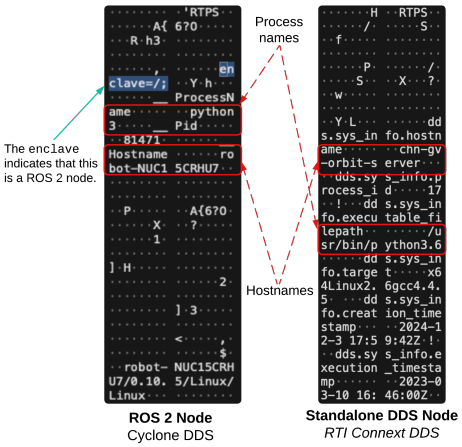

Fig. 9: Comparison of ROS 2 and standalone DDS nodes from our scan. Our fingerprinting methodology identifies the `robot-NUC15CRHU7` (left) and the Grass Valley Orbit Server (right) as ROS 2 and standalone DDS respectively.

Case Study: `robot-NUC15CRHU7`. One of the examples our fingerprinting methodology determines is a ROS 2 node in South Korea with the hostname `robot-NUC15CRHU7`. The packet trace on the left of Fig. 9 shows an enclave in the payload and that the node is running Cyclone DDS. Further, searching for the serial number in the hostname reveals that the device is most likely a robot running on an NUC 15 Pro computer.

Case Study: Grass Valley Orbit Server. Our fingerprinting determined that this system in India is a standalone DDS node running RTI Connext DDS version 6.1.2. Although both ROS 2 Humble and Jazzy support this version, an enclave cannot be found in the response payload, as shown in the packet on the right of Fig. 9. The hostname `chn-gv-orbit-server` implies that this system is most likely running Grass Valley's Orbit application, a dynamic system orchestrator for media transfer and broadcasting.

V. Conclusion

Metadata leakage in DDS discovery exposes a critical, underexplored vulnerability in ROS 2 deployments. Unlike prior attacks, our work introduces the first fingerprinting technique for ROS 2, allowing adversaries to distinguish ROS 2 nodes from standalone DDS instances. By scanning the public IPv4 space for DDS-based systems, we confirm that implementations expose unique metadata making them susceptible to ROS 2 fingerprinting, such as node names, hostnames, and even usernames in the case of RTI Connext DDS, which we emphasize is severe enough to be a CVE. Cyclone DDS and RTI Connext DDS provide sufficient metadata for identification, while Fast DDS does not, warranting further investigation into alternative reconnaissance methods.

We move beyond past DDS security research by demonstrating a structured attack chain on ROS 2 systems using RTI Connext DDS, showing how precise fingerprinting of these systems enables a more targeted, efficient exploitation. Our TurtleBot3 experiment illustrates how metadata leakage can lead to unauthorized access, emphasizing the need for

stronger security defaults. Our Internet-wide scan confirms that deployments with default configurations exist and that no discovered nodes exhibited SROS2 enabled settings. SROS2 requires manual certificate generation and enclave configuration, creating significant adoption friction that aligns with CWE-1188 (Insecure Default Initialization of Resource) [34]. The persistence of default insecure deployments despite available security tooling underscores that optional security is insufficient. To mitigate the risks of such deployments, we recommend strengthening defenses through architectural changes from node configurations to network security. Restricting public IP exposure through mechanisms like virtual private networks; reducing metadata shared over a network; enforcing usage of security plugins and ROS enclaves by default; and deploying real-time anomaly detection to prevent reconnaissance and exploitation are all ways to improve the security of these deployments. Future work we are considering includes refining fingerprinting methods beyond metadata extraction to accommodate implementations that reveal little metadata such as Fast DDS and Gurum DDS, automating more of our ROS 2 reconnaissance framework, and extending exploitation analysis across robotic platforms.

ACKNOWLEDGEMENTS

This research was supported in part by the National Science Foundation under grants CNS-2145493 and the United States Department of Transportation University Transportation Center Program under grant 69A3552348327 for the CARMEN+ University Transportation Center.

REFERENCES

[1] Object Management Group Standards Development Organization, "OMG Data Distribution Service (DDS)," 2015. [Online]. Available: https://www.omg.org/spec/DDS/1.4/PDF

[2] Object Management Group Standards Development Organization, "The Real-time Publish-Subscribe Protocol DDS Interoperability Wire Protocol (DDSI-RTPS™) Specification," 2022. [Online]. Available: https://www.omg.org/spec/DDSI-RTPS/2.5/PDF

[3] F. Maggi, R. Vosseler, M. Cheng, P. Kuo, C. Toyama, T.-L. Yen, and E. Boasson, "A Security Analysis of the Data Distribution Service (DDS) Protocol," 2022. [Online]. Available: https://documents.trendmicro.com/assets/white_papers/wp-a-security-analysis-of-the-data-distribution-service-dds-protocol.pdf

[4] W. Chen, S. Der, Y. Luo, F. Alshammari, and Q. A. Chen, "Understanding the Internet-Wide Vulnerability Landscape for ROS-based Robotic Vehicles," in *Symposium on Vehicle Security & Privacy*, 2024.

[5] M. J. Michaud, T. Dean, and S. P. Leblanc, "Attacking omg data distribution service (dds) based real-time mission critical distributed systems," in *International Conference on Malicious and Unwanted Software (MALWARE)*, 2018.

[6] MITRE, "CWE-406: Insufficient Filtering of Blacklisted Characters in Input," Common Weakness Enumeration, 2024. [Online]. Available: https://cwe.mitre.org/data/definitions/406.html

[7] D. Portugal, R. P. Rocha, and J. P. Castilho, "Inquiring the robot operating system community on the state of adoption of the ros 2 robotics middleware," *International Journal of Intelligent Robotics and Applications*, 2025.

[8] MITRE, "CWE-200: Exposure of Sensitive Information to an Unauthorized Actor," Common Weakness Enumeration, 2006. [Online]. Available: https://cwe.mitre.org/data/definitions/200.html

[9] MITRE, "CAPEC-312: Active OS Fingerprinting," Common Attack Pattern Enumeration and Classification, 2018. [Online]. Available: https://capec.mitre.org/data/definitions/312.html

[10] MITRE, "Reconnaissance, Tactic TA0043 - Enterprise," MITRE ATT&CK, 2020. [Online]. Available: https://attack.mitre.org/tactics/TA0043/

[11] M. Hansen, A. Blasdel, and C. Buscaron, "ROS 2 Foxy Fitzroy: Setting a new standard for production robot development," 2020. [Online]. Available: https://aws.amazon.com/blogs/robotics/ros-2-foxy-fitzroy-robot-development/

[12] OMG, "DDS Security specification, version 1.2," Tech. Rep., 2024. [Online]. Available: https://www.omg.org/spec/DDS-SECURITY/1.2/PDF

[13] K. Chen, R. Hoque, K. Dharmarajan, E. LLontopl, S. Adebola, J. Ichnowski, J. Kubiatowicz, and K. Goldberg, "FogROS2-SGC: A ROS2 Cloud Robotics Platform for Secure Global Connectivity," in *IROS*, 2023.

[14] "Shodan Search Engine." [Online]. Available: https://www.shodan.io/dashboard

[15] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-Wide Scanning and its Security Applications," in *USENIX Security*, 2013.

[16] V. Mayoral-Vilches, "Robot hacking manual (rhm)," *arXiv preprint arXiv:2203.04765*, 2022.

[17] N. DeMarinis, S. Tellex, V. P. Kemerlis, G. Konidaris, and R. Fonseca, "Scanning the Internet for ROS: A View of Security in Robotics Research," in *ICRA*, 2019.

[18] Cybersecurity & Infrastructure Security Agency (CISA), "ICS Advisory (ICSA-21-315-02): Multiple Data Distribution Service (DDS) Implementations," 2021. [Online]. Available: https://us-cert.cisa.gov/ics/advisories/icsa-21-315-02

[19] W. Woodall, "ROS on DDS," 2014. [Online]. Available: https://design.ros2.org/articles/ros_on_dds.html

[20] M. Arguedas, S. Ragnarok, D. Thomas, A. Nash, G. Biggs, and M. A. Gutierrez, *ROS 2 Releases and Target Platforms*, 2018. [Online]. Available: https://ros.org/reps/rep-2000.html

[21] R. White and M. Arguedas, "ROS 2 Security Enclaves," 2020. [Online]. Available: https://design.ros2.org/articles/ros2_security_enclaves.html

[22] M. T. Inc., "Autosar® ecosystem," 2026. [Online]. Available: https://www.microchip.com/en-us/solutions/automotive-and-transportation/autosar

[23] AUTOSAR, "Integration of DDS Security," 2025. [Online]. Available: https://www.autosar.org/fileadmin/standards/R25-11/AP/AUTOSAR_AP_TR_DDSSecurityIntegration.pdf

[24] The Electric Power Research Institute, "IEC 61968-5 Distributed Energy Optimization to Open Field Message Bus (OpenFMB) Mapping," The Electric Power Research Initiative, Tech. Rep., 2019.

[25] Open Robotics, "Setting up security." [Online]. Available: https://docs.ros.org/en/humble/Tutorials/Advanced/Security/Introducing-ros2-security.html#setting-up-security

[26] Gurum Networks, "rmw_gurumdds_cpp/src/rmw_context_impl.cpp." [Online]. Available: https://github.com/gurumnet/rmw_gurumdds/blob/rolling/rmw_gurumdds_cpp/src/rmw_context_impl.cpp#L81-L86

[27] Y. Wu, P. Cao, A. Withers, Z. T. Kalbarczyk, and R. K. Iyer, "Mining threat intelligence from billion-scale ssh brute-force attacks," in *NDSS Poster Program*, 2020. [Online]. Available: https://www.ndss-symposium.org/wp-content/uploads/2020/02/NDSS2020posters_paper_13.pdf

[28] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "IoTPOT: Analysing the Rise of IoT Compromises," in *USENIX Workshop on Offensive Technologies (WOOT)*, 2015.

[29] A. Das, J. Bonneau, M. Caesar, N. Borisov, and X. Wang, "The Tangled Web of Password Reuse," in *Network and Distributed System Security Symposium (NDSS)*, 2014.

[30] B. Covrig, E. Barrueco Mikelarena, C. Rosca, C. Goanta, G. Spanakis, and A. Zarras, "Upside down: Exploring the ecosystem of dark web data markets," in *IFIP International Conference on ICT Systems Security and Privacy Protection (SEC)*, 2022.

[31] National Institute of Standards and Technology, "CVE-2024-30963: Buffer Overflow in ROS 2 navigation2-humble," 2024. [Online]. Available: https://nvd.nist.gov/vuln/detail/CVE-2024-30963

[32] A. Tundis, E. M. Modo Nga, and M. Mühlhäuser, "An exploratory analysis on the impact of Shodan scanning tool on the network attacks," in *International Conference on Availability, Reliability and Security*, ser. ARES, 2021.

[33] Gurum Networks, "GurumDDS." [Online]. Available: https://gurum.cc/gurumdds_eng

[34] MITRE, "CWE-1188: Insecure Default Initialization of Resource," Common Weakness Enumeration, 2021. [Online]. Available: https://cwe.mitre.org/data/definitions/1188.html