Understanding, describing, and mitigating the flow of personal data in ROS 2 systems to comply with the GDPR and beyond.*

Tim Zander[1], Jonas Wohnig[2] and Juergen Beyerer[1,3]

***Abstract*— With the introduction of privacy protection laws such as the GDPR in the EU, carrying out privacy impact assessments (PIA) for new robotics applications interacting with humans is becoming a necessary part of the development process. We discuss a methodology and develop and release software[1] to describe and mitigate the flow of personal data as part of the development process or retroactively in the Robot Operating System - ROS 2.**

I. Introduction

Robots entering a broad range of use cases interacting with humans will almost necessarily process personal data. Because of this, many privacy laws have become applicable to this robotics application. Hence the need to understand, communicate, and mitigate the processing of personal data becomes an integral part of development to fulfill existing privacy laws. Moreover, data protection on the manufacturer side is what users are the most concerned about[2].

We focus on the General Data Protection Regulation (GDPR) of the European Union. This law was a guiding example to many other privacy laws around the globe as part of the so-called Brussels effect [3]. We discuss the principles of GDPR which later guide our technical choices such as the transparency of the processing and the right to privacy by design and default as the GDPR's *"right to explanation"* which is sought to provide meaningful information about the logic involved[4][5]. These principles do de facto dictate a big part of the design of a robotics product that interacts with humans today[6][7]. In terms of the privacy threat analysis framework LINDDUN GO[8], we especially try to reduce the *"Overbroad Exposure of Personal Data"* beyond what is necessary for the application and try to avoid having *"INSUFFICIENT TRANSPARENCY - Data subjects are insufficiently informed about the collection and processing of their personal data."*

The Robot Operating System (ROS) is a software framework for developing robotics applications. In particular, the newest version of the Robot operating system - ROS 2 introduces many interesting new features. We will focus on this version for the rest of the paper. Among the new features is an underlying communication based on the Data Distribution Service (DDS). DDS is a standard by the Object Management Group (OMG) for a middleware for machine-to-machine communication offering certain quality of service capabilities through a publish-subscribe pattern. A ROS 2 program's underlying structure uses nodes that communicate in such a publish-subscribe pattern through some DDS implementation. These nodes use so-called topics, services, and actions to send and receive messages to and from each other[9].

The DDS standard has an additional security specification (DDS-SECURITY) which can be utilized in ROS 2 programs. These capabilities are accessible under the SROS package. SROS has two types of Certificate Authorities (CAs) as centerpieces of the Public Key Infrastructure (PKI): There are Identity CAs to enable authentication of and secure communication between nodes. There are also Permission CAs to sign the Mandatory Access Control (MAC) rules for each node. The overhead of these security features was found suitable even for very resource-constrained environments [10].

SROS utilizes XML files to implement its security features. It uses policy files to generate the permission files that regulate the access controls in DDS. While permission files apply to single nodes, policy files can describe access controls for multiple enclaves, each containing multiple nodes, at once. For these policy files, SROS has a custom XML schema[11]. It is described by a 'policy' root, which contains one 'enclaves' element, which in turn contains one or more 'enclave' elements. These 'enclave' elements stand for one ROS 2 security enclave each, signified by the enclave path. Each 'enclave' consists of one or more 'profiles' elements, which in turn contain one or more 'profile' elements each. A 'profile' contains a set of access rules for one specific ROS 2 node, describing the use of which communication channels are allowed or denied. Together, these 'profile' elements contain all of the connection information needed for the construction of the ROS 2 graph. It describes which nodes are connected to which topics, services, and actions, and in what role. [12]

The goal of this paper is to describe a tool intended to be used by software developers, data protection officers, and professionals from adjacent areas ('users') as defined in the GDPR to self-evaluate and communicate the personal data processing of their ROS 2 system. We focus on data generated or gathered by ROS 2 nodes and transmitted within the ROS 2 graph. Other ways where sensitive data can be transmitted within a system or made available to the outside are not covered by this work. Although this an important step, we assume that the ROS 2 nodes are evaluated by the users to ensure that personal data only flows through the ROS 2 communication layer. Hence as a demarcation to our

*This work was supported by funding from the topic Engineering Secure Systems of the Helmholtz Association (HGF) and by KASTEL Security Research Labs.

[1]Vision and Fusion Laboratory IES, KIT, Karlsruhe, Germany
[2]Salzburg Research FG, Salzburg, Austria
[3]Fraunhofer IOSB, Karlsruhe, Germany

work stands the vulnerability assessment[13] of the ROS2 nodes itself and the of general vulnerabilities of the whole ROS2 system[14][15]. Both are of course important topics for ensuring that privacy and security goals are met. But we consider them out of scope for this work, as we solely concentrate on the flow of the private data in the ROS2 system and consider such techniques as complementary to ensure that the whole system ensures privacy. Also the case of timing attacks to get information out of the system we consider out of scope as this would assume that certain ROS2 nodes we control would be compromised.

II. Methodolgy and Implementation

The basis for our reasoning about ROS 2 systems is a directed graph structure with some additional information. A directed graph is the main way a ROS 2 system is presented in the literature and by ROS 2 itself[16][17][9]. We aim to detect unwanted information flow in a ROS 2 system by understanding the flow of personal information within such a ROS 2 graph. We describe in the following the steps that were taken to reason about the information flow of private information in such a system and how to improve its privacy.

A. Categorization

As a first step in our tool, the user categorizes nodes, topics, services, and actions. For this, a set of labels depending on the possible creation, transformation, or transportation of sensitive data by a node or the content transmitted by topics, services, and actions is applied. We now describe the different privacy categories each node can fall into and presents examples for each.

The node category *source* is applied to any node that potentially introduces personal data into the ROS 2 system. This could mean a node directly acquiring personal data such as a camera stream with humans. Another way would be a node that has access to personal data that was already generated or collected outside of the ROS 2 system such as a database with pictures of faces. This category of nodes is the first and most crucial to be identified by the user because only these nodes by definition produce or bring personal data into the ROS 2 system. Independently from the status of incoming edges, all outgoing edges of a source node are by default assumed to potentially carry personal data.

The node category *leak* encompasses nodes that potentially either directly make personal data accessible to untrusted parties or relay personal data to untrusted systems outside the ROS 2 graph. This would be a violation as it risks a *"Overbroad Exposure of Personal Data"* as *Personal data is shared with more services or external parties than necessary* as defined in LINDDUN GO[8].

The node category *conduit* is applied to nodes that neither introduce personal data to the ROS 2 graph nor relay such data to untrusted outside systems. A conduit node instead has the potential to transmit personal data to other nodes within the graph without changing their status as personal data. This could be a node that takes a video stream and adds semi-transparent image segmentation information onto the video stream or it applies some face recognition algorithm to a video and forwards the resulting information.

A node falls into the category of *sanitizer* when it takes data containing personal data as input, processes the data such that the personal data is removed, and then returns non-critical data as output. This could be a node that pseudonymizes or anonymizes data before it is forwarded to other nodes. As an example we can use them as seen in [18] and [19], or even through neural network-based semantic segmentation[20]. Independently from the status of incoming edges, all outgoing edges of a sanitizer node are by default assumed to carry no personal data.

The communication channels between nodes get assigned one of two labels. The first label *mundane* is assigned to any channel which by its nature never transmits personal data even if the adjacent nodes handle personal data. The second label is that of *sensitive* which marks channels that do or may transmit personal data. It is the default label if there is no clear evidence that the label mundane applies.

If there exists an input to a conduit node that may transmit personal data, all outputs of that conduit node by default are assumed to possibly transmit personal data. Inversely, if, for a conduit node, there exists no input that may transmit personal data, all outputs are assumed to not transmit personal data.

B. Implementation

We built and tested our software tool and ran our experiments using the Humble Hawksbill distribution of ROS 2. The code and the data of the experiments can be found online[1]. The Humble Hawksbill distribution is the current distribution with long-term support until 2027. As a DDS implementation the default "Fast-DDS" by eProsima which is shipped with Humble Hawksbill was used. We built code in Python and made use of a Python package 'lxml'[21] for parsing the SROS security XML files of ROS 2 systems. Furthermore, the ROS 2 graphs computation and visualizations were done with the package 'networkx'[22].

First, we describe how to get the necessary SROS policy files. The second part describes how these policy files are turned into a ROS 2 system graph. Then the categorization of nodes, topics, services, and actions is discussed. As a last step in the process, the inspection of the graph for vulnerabilities is presented. Finally, the different views that are generated by the tool and their purposes are introduced.

The first step is to procure the SROS policy files for the ROS 2 system that is being subjected to the audit. If the system already exists and is in use, there are two scenarios. If the ROS 2 system is already using SROS to encrypt communication then there exist some policy files which are collected. If the ROS 2 system is not using SROS MAC or when governance and permission files were created without using a policy file, the policy file can be generated via the built-in SROS generate policy functionality. This results in a policy file that sets the access rule to ALLOW for all connections that were observed during some chosen time-span called *spin-time*. This is necessary as there are situations where client nodes are only active to send

a request and receive a response and are destroyed right afterward and would otherwise not be captured. Furthermore, we constructed our program in such a way that if a graph of the ROS 2 system in question was already constructed we can add new information from policy files to this graph.

The policy files are then loaded and a ROS 2 graph of the system is then constructed. This is done by converting the XML files contained thereby identifying every profile within each policy and adding them iteratively to the graph. ROS 2 nodes, topics, services, and actions are represented as graph vertices. In addition to the type and name of each element, which includes the namespace, the vertices are assigned their enclave. Of the two vertices between which an edge can exist, one has to be a ROS 2 node and the other a message channel. These edges signify the direction of data flow between the node and the channel. While the role of a node in relation to a topic would be clear from the directional information alone (i.e., if the data flows from the node, it is the publisher, and if it flows towards the node, it is a subscriber), the same is not true for services and actions since both have bidirectional communication. This is why edges also store information about the role the node has in relation to the channel. The roles are publisher and subscriber for topics, server and client for services, and caller and executor for actions. At this point, the graph can be visualized from a general ROS 2 perspective to get a graphic representation of the status quo of the system. This also serves as a starting point for the next step namely the identification of the flow of potential personal data.

At this point, the graph elements have to be evaluated in terms of their relevance to the protection of personal data within the graph. Ideally, this is done by one or more users who together have the expertise to understand the code of ROS 2 nodes and the implications of their behavior for the protection of personal data. For this, these categorizations as defined in Section II-A are collected in a JavaScript Object Notation (JSON) file. For a start, all source nodes need to be identified and entered into the tool. This is the most important part of this step because vulnerabilities can only be detected if it is clear where personal data is introduced into the system. Then the remaining nodes should be categorized into leaks, conduits, and sanitizers, while communication channels are categorized into mundane and sensitive. This is done by starting with the vertices directly connected to a source node and iteratively inspecting the neighboring vertices. The remaining nodes that do not fit the description of either source, leak, or sanitizer are conduits. Similarly, if it cannot be ensured that a message channel fits the description of the mundane category, it is classified as sensitive.

After applying the labels from the categorization files to the graph, our tool constructs a privacy graph based on the ROS 2 graph. While the ROS 2 graph depicts the general data flow in the ROS 2 system, the privacy graph is meant to only represent the vertices and edges that potentially transport personal data. To achieve the construction of such a privacy graph, we start with a copy of the ROS 2 graph. Then, all sanitizer nodes are removed because they can only be endpoints for personal data. In the next step, all mundane topics, services, and actions are deleted because they do not transmit personal data. Next, all edges that represent connections that are set to 'DENY' in the policy are removed because no data is allowed to flow there. Lastly, all edges for which at least one vertex was deleted are removed as well, because without either of the vertices, there can be no data flow. To check the resulting privacy graph for vulnerabilities, all edge-disjointed paths from source to leak nodes are determined. For this, paths that already contain leaks not as endpoints are dismissed because their sub-paths will already be contained in the list of paths. If this results in at least one path, the system is considered vulnerable, and the list of paths is returned to the user. The user can now make appropriate changes to the ROS 2 system and the categorization of nodes and vertices and run the analysis again until no vulnerable paths remain. Finally, the tool then does not consider the ROS 2 system to be vulnerable based on the provided information and the resulting report can be included in a privacy impact assessment for example.

In summary, two kinds of graphs were created. The ROS 2 graph represents the whole ROS 2 system, while the privacy graph focuses on the elements of the system that handle personal data. We also provide visualizations of these graphs which both are presented in a way that expresses different aspects of them.

The ROS 2 graph view shows the full ROS 2 system as a network of nodes, topics, services, and actions. It differentiates between nodes and communication channels by the shape of the vertices and then between the different types of channels by their color. Edges are either black arrows signifying an allowed connection between a node and a communication channel and its direction or grey to signify an explicitly denied connection. This view is meant to serve as an overview of the whole system and the connections between its elements.

The privacy view contains only the relevant sub-graph in which vulnerabilities could be. The full ROS 2 graph serves as a starting point. From there, first, nodes that are categorized as sanitizers, as well as topics, services, and actions that are categorized as mundane, are removed because they cannot transmit personal data. The same is done for edges between nodes and communication channels that are disallowed by the SROS policy. Then all vertices that are not reachable from a source node are excluded since no personal data can reach them within the ROS 2 system. The remaining vertices are colored according to their privacy categorization. Edges are either depicted in red if they are part of a vulnerable path or in black if they are not. This view is meant to focus on the relevant information when analyzing a ROS 2 system for its vulnerabilities. An overview of all of this described can be found in Figure 1.

C. Example; a factory system in developement

In this imaginary case, we tackle a scenario for the planning of sanitizer nodes for a ROS system in its design phase. In this scenario, a modern factory is being planned that involves production steps where humans and robotic arms

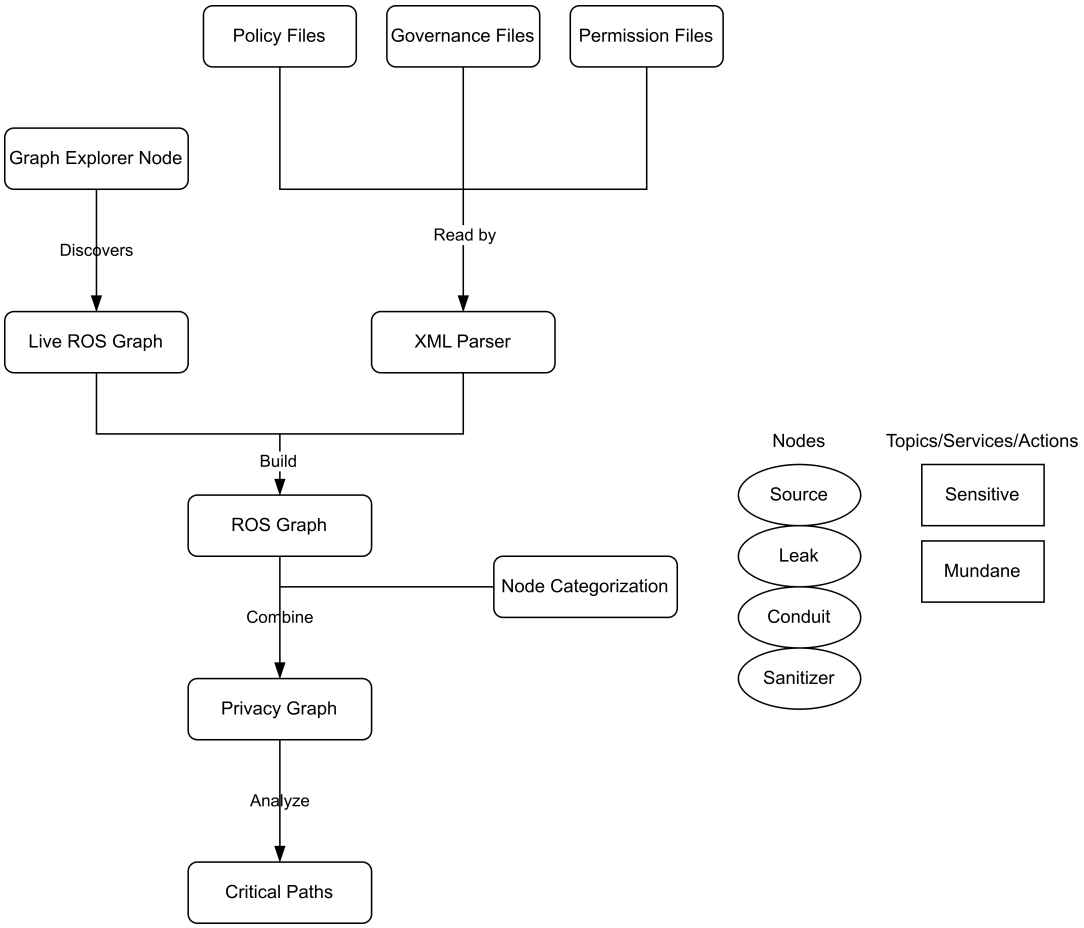


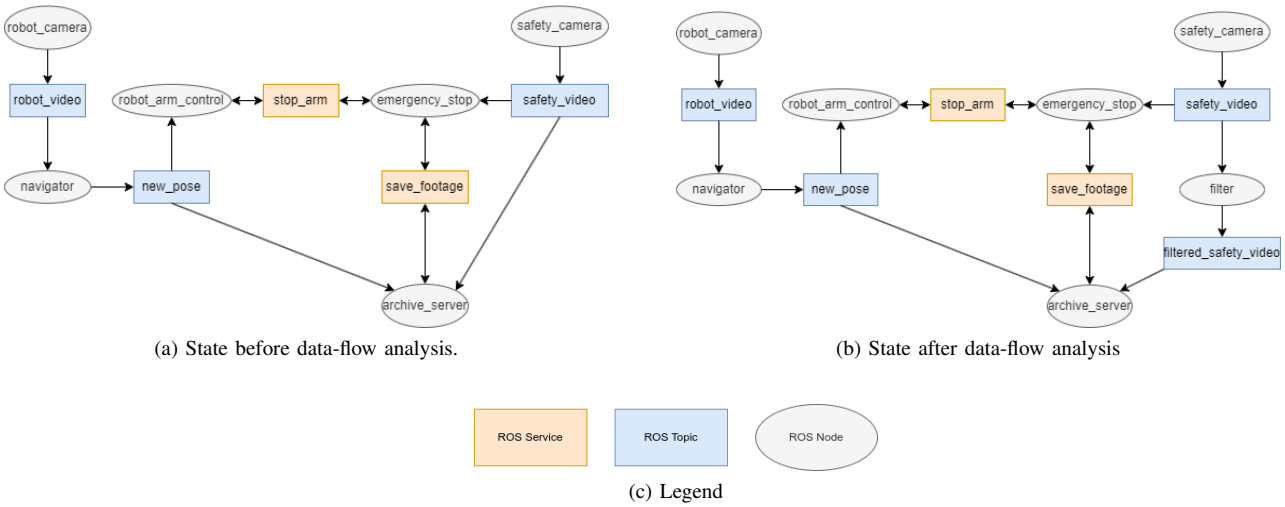
Fig. 1. Schematics of the general architecture described in Section II-B. The *Live ROS Graph* is not used to build the ROS Graph yet in the implementation. The same is true for the procession of the *Governance and Permission Files* by the *XML Parser*.

work collaboratively in spatial proximity to each other. To enable this complex work, each robot has its own cameras, which feed a safety and navigation system. The camera views the area around the robot and, as such, can also monitor human staff. The safety and navigation system takes the video stream as an input and a pose of the humans in the video. This pose is used to plan the future moves of the robot to protect humans from collisions with robot arms. In some cases, the robot needs to slow down or stop its actions to avoid collisions. Such events are safety-critical incidents and to analyze these events, the video feed from a few seconds before such an event and a few seconds after should be saved.

To realize this project, first, a ROS 2 prototype is developed by the user which is depicted in Fig. 2. Since video data generated in this section of the factory suffices to identify identifiable natural persons (i.e., faces in the video can be used to identify and track the factory staff), it is personal data. This makes the company that plans and then runs the factory and the staff involved in the respective activity data controllers and data processors. This means that they are mandated by Article 25 of the GDPR to implement appropriate technical and organizational data-protection measures, such as data minimization already at the design stage. The user also already has a filter that anonymizes faces in video footage, but this is computationally expensive so its use is limited.

The first step is to construct the ROS 2 graph for a prototype that does not use any SROS features yet. Then our tool takes the privacy evaluations of nodes and communication channels as input and returns output that serves both as a pointer to which elements need to be evaluated and if the evaluation is completed, as input for decisions about how to rectify vulnerable paths within the system.

We constructed this synthetic example in a way that allows the application of all the designed features of our tool. At the same time, it is an attempt to model a real process as closely as possible. At the start of this scenario, the user has a prototype of the ROS 2 system, including the robot arm and the security camera watching the restricted area around it. First, the user generates a policy for the initial prototype utilizing the SROS tools. This was done with the built-in ROS 2 functionality. A spin-time of 30 seconds was enough to generate a full working policy. This generated policy is then given to the tool as input, which creates the ROS 2 graph shown in Fig. 2. This is subsequently combined with a list of personal data sources that were identified by the user. In this case, the obvious sources are the robot's camera and the



(a) State before data-flow analysis.

(b) State after data-flow analysis

(c) Legend

Fig. 2. The ROS 2 graph of the system under development in Section II-C before and after the usage of our tool till a satisfactory state was reached. ROS Nodes are oval and colored gray, Topics are rectangular and blue, and Services are rectangular and of orange color.

camera surveying the restricted area around the robot. Then, iteratively, the paths to potential leak nodes are inspected, which means the potential leak nodes and communication channels to them are categorized according to their behavior relating to personal data. In total 5 runs of the tool for the construction of the privacy graph were necessary to reach a state where the privacy analysis was without vulnerable paths. These steps were the steps taken in each of the runs.

1) Categorized the nodes *robot camera* and *safety camera* as sources.
2) Categorized the node *archive server* as a leak node, the nodes *navigator* and *emergency stop* as conduits and the topics *robot video* and *safety video* as sensitive.
3) Categorized the topic *new pose* and the service *stop arm* as mundane.
4) Installed the sanitizer node *filter* and *filtered safety video*, a mundane topic, between the topic *safety video* and the node *archive server*.
5) Categorized the service *save footage* as mundane

Step 0 was the starting position. In steps 1 and 2, the user inspected the previously uncategorized nodes and channels that could come into contact with personal data. This leads to a situation where the path from the source node *safety camera* via the sensitive topic *safety video* to the leak node *archive server* does not contain any uncategorized elements and is still vulnerable. At the same time, the other source node, *robot camera* is no longer included in any vulnerable paths. This means that while the *robot camera* node does not need a filter to keep personal data from leaking, the node *safety camera* does. For the placement of the filter, there are two options: before or after the topic *safety video*. The main difference is that the filter before the topic *safety video* would publish the filtered video to the other subscriber *emergency stop* as well. The filter after the topic *safety video* would only filter the video for the *archive server* node. Since the execution of the *emergency stop*'s service *stop arm* is time-critical, the filter was placed after the topic *safety video* in step 3. This left another possible vulnerable path from *safety camera* to *archive server* via the service *save footage* which upon inspection in step 4 was categorized as mundane as well. In the end, the ROS 2 system was found to not be vulnerable anymore. The goal of the user was therefore achieved.

D. Example; Turtlebot

We searched for public examples of a complex ROS 2 system that included policy files for the whole system. The only one that was found was the TurtleBot3[23] with SROS policy files as described in [24]. The version of TurtleBot3 the policy was written for does not have a video camera but instead a laser-based navigation system. Still, as the laser system could be used to eavesdrop on humans[25], it is treated as the source of personal data in this scenario. In a self-study, the node that introduces this laser data was classified as a *source* node. Further in any step run by our program, various nodes and topics are inspected and user-classified into various categories. After 6 steps and the identification of a side channel between to mapping nodes in the ROS 2 system, we are left with a connection to the so-called recoveries server which is responsible for robot recovery and was classified as a leak. This was due to the server storing data without ensuring that no personal data was stored or that storing such data was warranted. As this node is responsible for robot recovery, storing of most recent data is probably warranted but one would need to reasonably ensure that it can not be externally read. We will publish the full analysis of the TurtleBot3 alongside the release of our software.

III. Discussion

First, we discuss the limitations of our work. Then we discuss possible future work. Finally, we will recapitulate the central thoughts and findings made.

One limitation of the tool is that it only considers the policy files SROS works with and not the governance and permission files of DDS. Although it still covers the core information, this makes the tool less flexible as we cannot apply our work to general DDS systems. Another issue is that for big ROS 2 systems without existing policy files, it is impractical or very costly to create a policy without in-depth knowledge. Workarounds are using multiple policies for the whole system or dividing it up into subsystems and analyzing them separately. Further, the tool is limited to ROS 2, while real systems often do not solely rely on ROS 2 but include communication channels and software that is not using the ROS 2 framework as seen in the Turtlebot case. Furthermore, there was a lack of real-world examples of complex ROS 2 systems that already had a complete SROS policy file describing their access controls. Because there were no experts in law or privacy protection involved in creating this tool, a more in-depth legal perspective on the usefulness of the tool would be interesting. Still, the tool helps to make a full analysis of the flow of private sensitive data in ROS 2 easier and assists in providing reasonable assurance that the controller or a third party has a legitimate interest (in the sense of the GDPR) in processing the data in the ROS 2 system.

For potential further work, we have the following suggestions. One way to improve the tool could be the incorporation of the DDS governance and permission files into the process. Both are files that belong to the DDS security system. They are created in the process of applying SROS methods to ROS 2 systems. Governance files document the general settings for a security enclave. While the tool so far assumes default governance settings as provided by SROS commands, reading the governance files can improve the flexibility of the tool for different environments. Permission files each contain the access privileges of one DDS node. While they can be created from policy files, and as such, all their information would be contained in a concentrated form in these policy files, reading the permission files directly can have its benefits. For one, the permission files directly set the conditions in the security layer of DDS and are signed by the CA. As such, they would be better suited for stronger threat models than the one used.

So far there is no modeling of the ROS 2 graph at runtime, which limits the applicability to dynamic ROS 2 systems. Creating a policy file based on the current ROS 2 graph uses the built-in functionality of SROS. A custom implementation could detect elements of the ROS 2 graph at runtime and directly add them to the graph, simplifying the process for bigger systems. This would make the security system more flexible and would allow certain nodes to be added while still keeping the privacy analysis valid.

Similar to established methods[26], code comments could be one way to increase automation in this tool. Users could annotate nodes and the data they transmit via topics, services, and actions while developing them. From these comments, the lists for source, conduit, and leak nodes could be generated automatically. After changes to the system, only the labels of the edited sections of nodes would have to be reevaluated. With access to the code of running nodes, the system could even evaluate the privacy status of ROS 2 at runtime.

Another addition that might make our tool more practical and versatile is the addition of non-ROS 2 graph elements. As for now, it is limited to ROS2 nodes interacting with each other. These elements could, for example, model external servers, storage devices, or non-ROS 2 communication channels between nodes like shared memory. That way, things that are right now only indirectly expressed via the categorization of ROS 2 elements can be made explicit in the graph. This would mean extending the categorization system to include information like the duration of storage for storage devices. It could help analyze a wider range of threats.

The examples presented are each limited to one kind of personal data, but larger and more complex systems are likely computing different types simultaneously. In that case, a user could benefit from the separation and visualization of different personal data categories within the tool. The differentiation could be based on the computational data type (e.g., string, integer, or visual data), the GDPR category (e.g., physiological, genetic, or economic data), the sensitivity of the data as described in Article 9 of the GDPR (e.g., especially protected data like religious affiliation and biometric data have to be handled differently than an online identifier), or on a combination of those options. At this point, one limitation is that the user would have to run the tool for each kind of data separately to get individual results for multiple categories. A differentiation could, for example, mark a node as a sanitizer for one kind of personal data and a potential leak for a different kind because it only transmits one type of data to an untrusted outside system but computes the other kind internally.

ROS 2 does not support the optional DDS security data tagging in the Service Plugin Interface (SPI). If that changed or workarounds based on DDS implementations supporting data tagging were created, this could offer a different approach to tracing the flow of personal data. Instead of building a model of the ROS 2 graph, inspecting it from the outside, and intervening when the structure allows data to leak, data could directly be tagged as sensitive and limited in the nodes that could handle it. This seems like a promising way to cut the necessary effort users have to expend to secure personal data and could be one way to extend the ROS 2 framework.

One project that could decrease the workload for this tool user could be an automated categorization of nodes according to their relationship to personal data using large language models. Current code generation tools like GitHub's Copilot are already capable of suggesting comments based on the code context. Similar models could be trained to recognize typical structures in nodes that collect or potentially leak personal data and either directly suggest privacy labels or provide code comments.

Aside from the suggestions for improvements to the work we just made, we still achieved the following. We modeled

data flow and protection in ROS 2 and created a tool that can assist data controllers and data processors in describing the processing of personal data and finding vulnerabilities in the privacy of ROS 2 systems. A model was presented that described different privacy roles for elements of the ROS 2 graph. This was implemented in a tool that used SROS features to map the graph, apply user-generated privacy labels to its vertices, and analyze the result for possible paths that might lead personal data out of the system. This tool was tested both on a synthetic and a real-world example and worked well in our self-study. Hence the tool can play a crucial role in helping guarantee compliance of robotic systems with GDPR rules or similar privacy laws.

REFERENCES

[1] "ROS 2 private data flow analysis tool and experiments," Feb. 2024. [Online]. Available: https://github.com/JonasWoh/ROS2PrivacyGraph

[2] C. Lutz and A. Tamo-Larrieux, "The robot privacy paradox: Understanding how privacy concerns shape intentions to use social robots," *Human-Machine Communication*, vol. 1, p. 87–111, 2020. [Online]. Available: https://search.informit.org/doi/10.3316/INFORMIT.097053479720281

[3] G. V. Cervi, "Why and how does the eu rule global digital policy: an empirical analysis of eu regulatory influence in data protection laws," *Digital Society*, vol. 1, no. 2, Sep. 2022. [Online]. Available: http://dx.doi.org/10.1007/s44206-022-00005-3

[4] S. Wachter, B. Mittelstadt, and L. Floridi, "Transparent, explainable, and accountable ai for robotics," *Science Robotics*, vol. 2, no. 6, p. eaan6080, 2017. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.aan6080

[5] H. Felzmann, E. Fosch-Villaronga, C. Lutz, and A. Tamo-Larrieux, "Robots and transparency: The multiple dimensions of transparency in the context of robot technologies," *IEEE Robotics & Automation Magazine*, vol. 26, no. 2, pp. 71–78, 2019.

[6] S. Wachter, B. Mittelstadt, and L. Floridi, "Transparent, explainable, and accountable ai for robotics," *Science robotics*, vol. 2, no. 6, p. eaan6080, 2017.

[7] T. Hoffmann and G. Prause, "On the regulatory framework for last-mile delivery robots," *Machines*, vol. 6, no. 3, p. 33, 2018.

[8] K. Wuyts, L. Sion, and W. Joosen, "Linddun go: A lightweight approach to privacy threat modeling," in *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2020, pp. 302–309.

[9] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[10] G. Spilere Nandi, D. Pereira, J. Proença, E. Tovar, A. Rodriguez, and P. Garrido, "Secure integration of extremely resource-constrained nodes on distributed ros2 applications," 2023. [Online]. Available: https://doi.org/10.12688/openreseurope.16108.1

[11] R. White, G. Caiazza, H. Christensen, and A. Cortesi, "Procedurally provisioned access control for robotic systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. [Online]. Available: https://doi.org/10.1109/IROS.2018.8594462

[12] K. Hjerppe, J. Ruohonen, and V. Leppänen, *Annotation-Based Static Analysis for Personal Data Protection*. Cham: Springer International Publishing, 2020, pp. 343–358. [Online]. Available: https://doi.org/10.1007/978-3-030-42504-3-22

[13] M. Dowd, J. McDonald, and J. Schuh, *The art of software security assessment: Identifying and preventing software vulnerabilities*. Pearson Education, 2006.

[14] Y. Patel, P. H. Rughani, and D. Desai, "Analyzing security vulnerability and forensic investigation of ros2: A case study," in *Proceedings of the 8th International Conference on Robotics and Artificial Intelligence*, ser. ICRAI '22. New York, NY, USA: Association for Computing Machinery, 2023, p. 6–12. [Online]. Available: https://doi.org/10.1145/3573910.3573912

[15] V. DiLuoffo, W. Michalson, and B. Sunar, "Robot operating system 2: The need for a holistic security approach to robotic architectures," *International Journal of Advanced Robotic Systems*, vol. 15, p. 172988141877001, 05 2018.

[16] V. Mayoral-Vilches, R. White, G. Caiazza, and M. Arguedas, "Sros2: Usable cyber security tools for ros 2," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 11 253–11 259.

[17] R. R. Beck, A. Vijeev, and V. Ganapathy, "Privaros: A framework for privacy-compliant delivery drones," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 181–194. [Online]. Available: https://doi.org/10.1145/3372297.3417858

[18] D. J. Butler, J. Huang, F. Roesner, and M. Cakmak, "The privacy-utility tradeoff for remotely teleoperated robots," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 27–34. [Online]. Available: https://doi.org/10.1145/2696454.2696484

[19] S. Jana, A. Narayanan, and V. Shmatikov, "A scanner darkly: Protecting user privacy from perceptual applications," in *2013 IEEE Symposium on Security and Privacy*, 2013, pp. 349–363.

[20] M. H. Abbasi, B. Majidi, M. Eshghi, and E. H. Abbasi, "Deep visual privacy preserving for internet of robotic things," in *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, 2019, pp. 292–296.

[21] S. Behnel, M. Faassen, and I. Bicking, "lxml: Xml and html with python," 2005.

[22] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.

[23] R. Amsters and P. Slaets, *Turtlebot 3 as a Robotics Education Platform*. Cham: Springer International Publishing, 2020, pp. 170–181.

[24] V. Mayoral-Vilches, R. White, G. Caiazza, and M. Arguedas, "Sros2: Usable cyber security tools for ros 2," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 11 253–11 259.

[25] S. Sami, Y. Dai, S. R. X. Tan, N. Roy, and J. Han, "Spying with your robot vacuum cleaner: eavesdropping via lidar sensors," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, ser. SenSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 354–367. [Online]. Available: https://doi.org/10.1145/3384419.3430781

[26] K. Hjerppe, J. Ruohonen, and V. Leppänen, *Annotation-Based Static Analysis for Personal Data Protection*. Cham: Springer International Publishing, 2020, pp. 343–358. [Online]. Available: https://doi.org/10.1007/978-3-030-42504-3_22