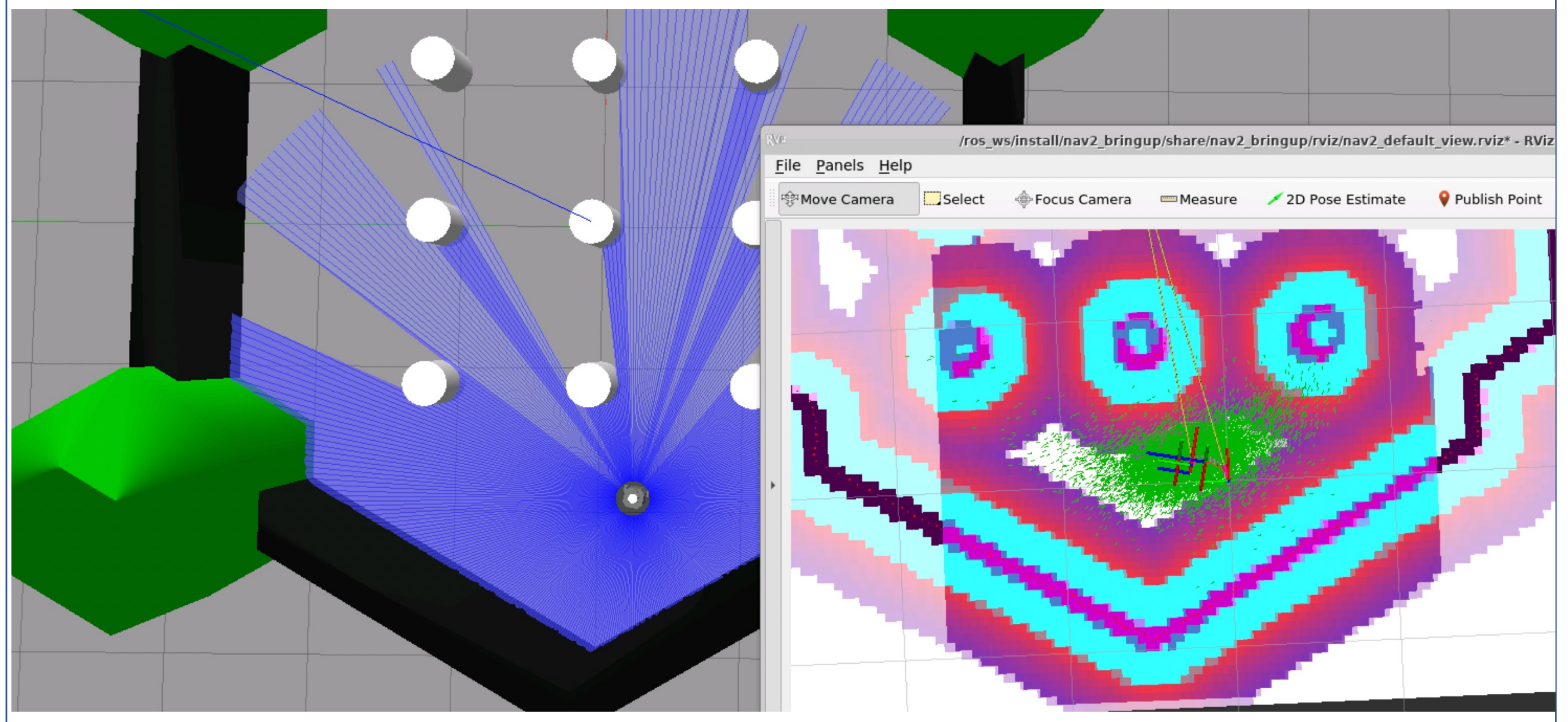
Measuring the effect of security enforcement on functional performance of a ROS 2 TurtleBot

Paul Smith[1,2], Simon Chu[1], Changjian Zhang[1], Eunsuk Kang[1], Christian Kästner[1], Bradley Schmerl[1]

1. Institute for Software Research, Carnegie Mellon University, Pittsburgh, PA 15213
2. St. Mary's University, San Antonio, TX 78228

Problem

Many cyber attacks targeting robots can be mitigated using the built-in security features introduced by ROS 2 that implement encryption, authentication, and access control through the DDS middleware[1]. Previous studies on ROS 2 have focused on measuring changes in communication latency and throughput through simple talker-listener systems, rather than robot behavior[3,4,5] . Here, we measure the impact of security on functional performance metrics like navigation time, final position accuracy, and path smoothness, using a simulated TurtleBot3 running ROS 2 Foxy.



Gazebo simulation of TurtleBot3 Burger alongside the Rviz2 navigation GUI. Here the particle count for localization has been greatly increased to affect the performance of the TurtleBot.

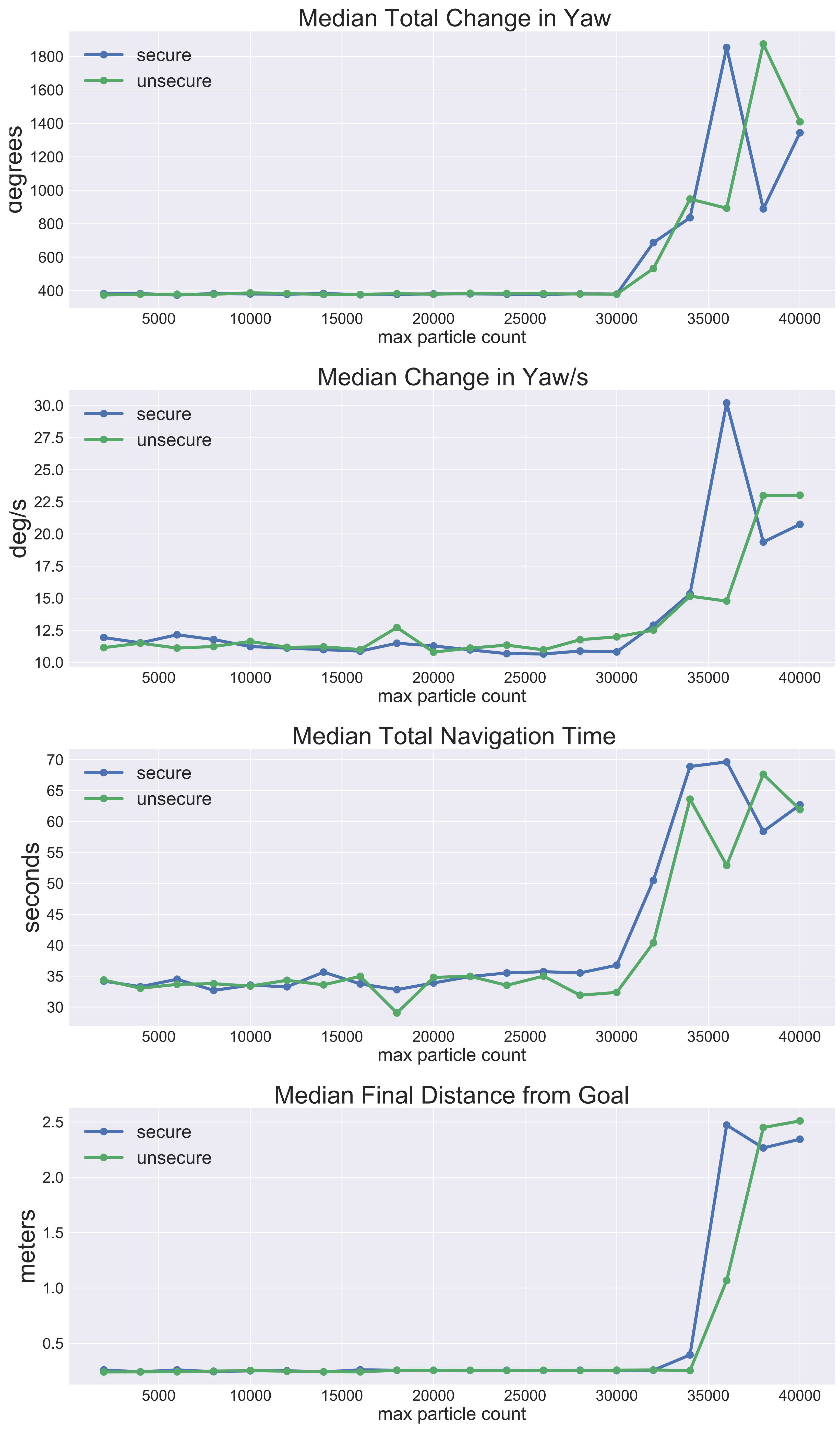
Approach

To answer the question "Does enabling security configurations add overhead to the performance of the TurtleBot?" we needed to strain the performance of the TurtleBot to the point where additional overhead could have a measurable impact on performance. To accomplish this, we simulated the TurtleBot attempting to navigate a set of waypoints with increasing particle counts, which are used to localize the robot's position. After each simulation, behavioral metrics such as navigation time and robot jitter were recorded from log files. Particle counts were incremented in steps of 2000 particles, and each configuration was simulated 10 times with both security enabled and disabled.

Tools

- 2017 Macbook Pro (specs: 2.2 GHz Core i7, 16GB RAM)
- Ubuntu 20.04 Docker container (8gb RAM , 6 CPUs)
- ROS2 Foxy
- Eclipse Cyclone DDS
- Gazebo TurtleBot3 Burger simulation
- RViz2
- Rosbag2
- Byobu
- Python, Pandas, Matplotlib, Scipy, Numpy

Experimental Data



Based on various performance metrics, we can see that the additional overhead from encrypting all internal communications causes the TurtleBot to become overloaded at a lower particle count in comparison to the unsecure TurtleBot.

Performance Statistics

particle count	32000	34000	36000	38000	40000
total yaw	0.2284	0.8735	0.0089	0.0009	0.3360
yaw rate	0.3884	0.4454	0.0005	0.0277	0.8966
nav time	0.5023	0.3375	0.1136	0.0228	0.2119
final distance	0.2641	0.0365	0.0046	0.0354	0.8335

P-values from t-tests for upper range of particle counts. Highlighted cells show p-values < 0.05

Though 10 runs at each particle count does not meet the requirement of the central limit theorem, outliers were removed, and the resulting boxplots appeared approximately normal, therefore a t test is appropriate for comparing the means of the secure and unsecure data. We can see there is a significant difference between the two datasets where they diverge at higher particle counts.

Challenges

ROS 2 adds the ability to automatically generate a security policy based on the running node graph, and can generate x.509 certificates and a set of keys for every node based on this policy or a manually defined access control policy[2]. However, the security configurations did not work with the default ROS middleware implementation (RMW), Fast RTPS, and needed to be changed to Eclipse Cyclone DDS. The generated access control policy also needed to be manually modified to allow for logging from Rosbag2.

Conclusion

While the addition of security did not impact the functional performance of the TurtleBot with default configurations, the earlier degradation in performance when secured is evidence of the additional overhead from security. Though this overhead may not have had a significant impact, that may not be the case for more complex systems or resource-constrained robots. This underscores the importance of measuring the impact of security on distributed systems, especially cyber-physical ones, to ensure they operate both safely and securely. Future work could involve the creation of a framework for measuring the functional performance of different robots based on user defined behavioral metrics.

Acknowledgements

References

1. Bonaci Tamara, Jeffrey Herron, Tariq Yusuf, Junjie Yan, Tadayoshi Kohno, and Howard Jay Chizeck. "To Make a Robot Secure: An Experimental Analysis of Cyber Security Threats Against Teleoperated Surgical Robots". arXiv:1504.04339 [Cs], April 16, 2015. http://arxiv.org/abs/1504.04339.
2. ROS2 Design: DDS Security, url: https://design.ros2.org/articles/ros2_dds_security.html
3. Kim, J., Smereka, J.M., Cheung, C., Nepal, S., Grobler, M.: Security and performance considerations in ROS 2: a balancing act. arXiv preprint arXiv:1809.09566v1 (2018)
4. Matellán, Vicente & Balsa, Jesús & Casado García, Fernando & Fernández, Camino & Martín, Francisco & Rodríguez Lera, Francisco. (2016). Cybersecurity in Autonomous Systems: Evaluating the performance of hardening ROS.
5. Pemmaiah, A., Pangercic, D., Aggarwal, D., Neumann, K., & Marcey, K. (2020). Performance Testing in ROS 2. https://drive.google.com/file/d/15nX80RK6aS8abZvQAOnMNUEgh7px9V5S/view.
6. DiLuoffo, V., Michalson, W. R., & Sunar, B. (2018). Robot operating System 2: The need for a holistic security approach to robotic architectures. *International Journal of Advanced Robotic Systems*, *15*(3). https://doi.org/https://doi.org/10.1177/1729881418770011