EasyChair Preprint

№ 15324

Performance Impact of Strengthening the Accountability and Explainability System in Autonomous Robots

Alejandro González Cantón, Miguel Angel Gonzalez Santamarta, Enrique Soriano Salvador, Gorka Guardiola Muzquiz and Francisco J Rodríguez Lera

October 29, 2024

Performance Impact of Strengthening the Accountability and Explainability System in Autonomous Robots

Alejandro González-Cantón[1], Miguel A. González-Santamarta[1], Enrique Soriano[2], Gorka Guardiola[2], and Francisco J. Rodriguez Lera[1]

[1] Universidad de León, Campus de Vegazana, Spain,
fjrodl@unileon.es,
WWW home page: http://robotica.unileon.es
[2] Universidad Rey Juan Carlos, Campus de Fuenlabrada,
Madrid, Spain

Abstract. Enhancing data management and security in robotics is crucial for ensuring reliable and efficient robotic operations. Effective data management facilitates data auditing, which further improves system reliability and traceability. This paper explores the integration of SealFS, a secure file system, into autonomous robotic systems. The primary problem addressed is the potential impact of SealFS on system performance, particularly write latency, compared to traditional Ext4 file systems. The methodology involves conducting latency tests in audio, video, and navigation contexts, along with analyzing CPU usage, file size consistency, and power efficiency.

Keywords: Cibersecurity, ROS 2, ROS integration, Secure file systems, System reliability, Traceability in robotics

1 Introduction

Autonomous robots function in dynamic and frequently uncontrolled environments, rendering them vulnerable to a range of cybersecurity threats. Protecting the integrity of crucial operational logs is essential to ensure the reliability of forensic analysis[1]. Effective security measures must be implemented to mitigate these risks, safeguarding not only the robots' operational data but also ensuring the overall reliability and safety of their functions.

The integration of off-the-self tools such as SealFS[2], a tamper-evident logging solution, becomes a valuable asset. SealFS allows the system to maintain an immutable record of its activities, detecting and preventing unauthorized alterations to crucial log files. Therefore, implementing SealFS on a robot running ROS 2 (Robot Operating System 2) [3] could have several positive implications:

– Security Enhancement: By ensuring the integrity of log files, the system can enhance the overall security of the robot's operation. In a robotics context, where the system may interact with physical environments and potentially

sensitive data, maintaining tamper-evident logs can be crucial for forensic analysis and identifying security breaches.
- Resource Considerations: ROS 2 systems typically run on resource-constrained devices, especially in embedded or mobile robot applications. While the described system aims for lightweight implementation, it's essential to evaluate its resource requirements (CPU, memory, storage) to ensure compatibility with the robot's hardware capabilities. The system's impact on real-time performance, especially in time-sensitive robotic tasks, should also be considered.
- Integration Complexity: Integrating a Linux kernel module like SealFS into a robot running ROS 2 environment may introduce additional complexity to the system architecture. Some robot manufacturer use their own kernels for improving and guaranteeing robot performance. Besides, ROS 2 operates on top of middleware for communication between components, and integrating a file system-level security mechanism requires careful consideration of compatibility and potential conflicts with existing software stack components.

Implementing the described tamper-evident logging system on a robot with ROS 2 has the potential to enhance security and reliability, but it requires careful consideration of resource constraints, integration complexity, and physical security considerations inherent to robotic environments. Current alternatives such as the ones presented by Khasan[4] based on cryptographic file system can be also implemented as a middleware layer to encrypt individual files or directories using file system filter driver technology in the Windows kernel or a low-level file system layer, operating beneath the real file system, providing encryption for single- or multiple-disk partitions [5, 6]. However, this research faces the unix-like approach[2], which is the most extended solution used by ROS developers. Thus, this paper performs an analysis and discusses the possibilities of integration given its impacts.

1.1 Contribution

Measuring the impact of SealFS on a robot involves assessing various factors related to its performance, security, and overall functionality. Given the context of using SealFS in the domain of autonomous robots, the contributions are:

- Integration and deployment in real world robots: Integration and deployment in real-world robots involve several critical steps. These include adapting the system to the specific hardware and software configurations of the robots, ensuring seamless interoperability with existing components, and conducting extensive testing in real-world environments to validate performance and reliability.
- Performance evaluation: It helps determine how well the system or component performs under various conditions, ensuring that it operates efficiently and meets the required standards. The evaluation can reveal performance bottlenecks or limitations that may impede the system's functionality, allowing for targeted improvements.

The remainder of this paper is organized as follows. Section 2 provides a detailed review of SealFS, highlighting the previous empirical findings that underpin our study and its integration in ROS middleware. Section 3 outlines the hardware and software methods employed, including the research design and data collection procedures. In Section 4, we present the results of our analysis, offering a comprehensive examination of the data and its implications. Section 5 discusses the findings in the context of the existing literature, addressing potential limitations and suggesting areas for future research. Finally, Section 6 concludes the paper by summarizing the main contributions and implications of our study, and proposing practical applications of our findings.

2 Architecture

2.1 SealFS

The SealFS prototype originates from the wraps framework, which is a stackable file system that allows multiple file systems to be "stacked" on top of each other, presenting a unified view to the user or other applications. SealFS has undergone progressive refinements to align with kernel version 5.19.0-43, following its initial deployment on versions 4.8.17 and 4.15.0 (referenced in the associated version tags, with substantial bug resolutions incorporated until version 5.4.0-65). Serving as a Linux kernel module, SealFS establishes itself as a stackable file system with a primary objective of authenticating written data, thereby ensuring the creation of log entries that resist tampering. Operating within the paradigm of a forward integrity model, SealFS robustly mitigates the potential for attackers to fabricate logs predating privilege elevation.

Upon SealFS deployment atop an extant file system, comprehensive protection is extended to all files positioned beneath the designated mount point. This system confines write operations exclusively to append-only, rigorously validating the integrity of data written to the underlying files managed by an alternative file system. The extant manifestation, SealFSv2, integrates the principles of ratcheting and storage-based log anti-tamper mechanisms. This innovative approach introduces a degree of adaptability, allowing users to choose between attaining a state of complete theoretical security akin to its predecessor, SealFSv1, or adopting a partially linear degradation reminiscent of classical ratchet schemes. This flexibility empowers users to finely calibrate security parameters, thereby effectively balancing security considerations with computational resource usage.

2.2 ROSSealFS

This component is a ROS 2 node dedicated to logging, designed to integrate with SealFS seamlessly. This node enables the logging of data generated during the robot's operations. When paired with SealFS configuration, it establishes tamper-evident log files, ensuring authentication for forward integrity.

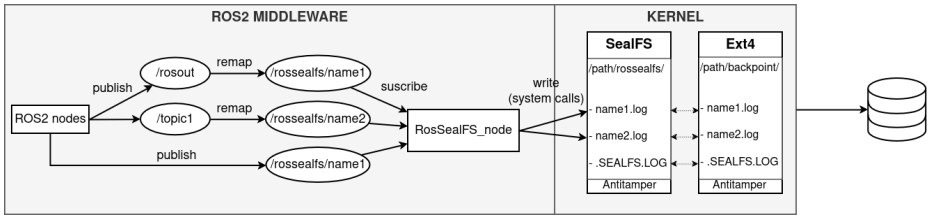

Fig. 1. Usage pipeline for SealFS and RosSealFS.

In this stage, the component is supplied with data from other robot topics and general logs. The goal is to gather all information generated by regular communications in ROS 2, as well as to incorporate standard logs found in specific directories (such as *.ros*). Consequently, the files will be stored in the designated folder prepared for deploying SealFS. Figure 1 illustrates the general overview of the proposed approach.

3 Materials and Methods

This section provides an overview of integrating SealFS into an autonomous robot. While this research focuses on two specific robots used by the authors in recent years, the process can be adapted for any other robot.

3.1 Hardware Description

This research utilizes two commercially available robotic platforms, whose configurations are presented in Table 1: the Summit XL from Robotnik Automation and the TIAGo from PAL Robotics. These platforms are employed in both outdoor and indoor contexts to comprehensively evaluate their performance and versatility:

- Summit XL: It offers a versatile and customizable outdoor platform [7] for research, education and industrial applications in robotics, with its modular design, mobility, sensor suite, and compatibility with ROS. Regarding energy specifications, it features a nominal voltage of 56.5 V and a current ranging from 1.5 A at rest to 4.4 A in motion, resulting in a power consumption of 84.75 W to 248.6 W.
- TIAGo : It is a versatile and customizable indoor robot platform [8] designed to support research, education, and development in various domains of robotics. It has manipulation capabilities, mobility, perception sensors, HRI features, and ROS 2 compatibility. Concerning the battery, it boasts a nominal energy of 720 Wh, a maximum discharge of 20 A, and a nominal voltage of 19.2V for the computer.

Table 1. Configuration of the robots including the hardware and software setups.

	OS	Kernel	ROS Version	RAM	CPU	Storage
TiAGO	Ubuntu 20.04	5.10.0-rt	Noetic	16 GB	i7-7700	SATA 3.3 SSD
Summit	Ubuntu 20.04	5.15.0-generic	Noetic	32 GB	i7-10700	NVMe 1.3 SSD

3.2 Experimental Description

To conduct the experimental phase, firstly, the different data streams obtained from a robot have been identified, with two variables observed within this framework: message size and frequency. Based on this observation, three data source scenarios are proposed, largely framing the current paradigm:

- Text message streams with low and irregular frequencies and small content, such as relevant system event logs.
- Data streams with high and regular frequency and small content, such as measurements from a laser sensor.
- Data streams with high and regular frequency and large content, such as RGB images from a camera.

Within the hardening framework in explainability methods, the most common scenario would be the first scenario of storing system logs, with the remaining scenarios being feasible but impractical in a real-world setting due to the challenges surrounding storage size on these mobile platforms.

The proposed methodology for practical experiments involves a systematic approach to integrating and evaluating SealFS within autonomous robots. Under these circumstances, it is possible to avoid the comparison between actuators characteristics, which would be analyzed in a future work. Three scenarios are proposed associated to three data flows that will be stored:

1. System logs will be stored during an autonomous navigation exercise using Nav2.
2. Artificial messages of intermediate size will be generated and sent at a high frequency through the middleware to simulate an auditory sensor, such as a microphone.
3. Frames captured by the robot's vision system will be stored independently.

These data flows will be stored in the file system of the robotic platform itself using SealFS file system, employing the default parameters of $nratchet = 5120$ and $size = 1000000$. For each scenario, five experiments will be conducted, each limited to a duration of 3 minutes, in order to obtain a sufficiently large data sample for analysis.

To measure the power consumption of the module, a different machine than the robotic platform will be utilized, as the battery does not provide a useful interface for employing this tool. To obtain a measurement as close to the actual one produced by the robot, adjustments have been made by modifying elements (such as screen, wheels, etc.) between measurements.

3.3 Metrics

To evaluate the integration of SealFS into the autonomous robots, we will analyze several key metrics.

- Throughput: Speed at which the robot can write log data to the file system with and without SealFS. This comparison will be necessary in order to ensure that SealFS does not significantly degrade performance. This value will be measured with iotop. This tool can produce the current write speeds for processes.
- Latency: Time it takes for the system to process write operations with and without SealFS. Lower latency is generally preferred. This value will be measured with ftrace[9]. With these tools, it is possible to extract the time it takes to make the right call.
- CPU Usage: Impact of SealFS on the robot's CPU usage during normal operation and intensive logging scenarios. This value will be measured with perf. This tool outputs the CPU usage of every process and thread.
- Power Efficiency: Robot's power consumption, especially in scenarios where the robot operates on battery power. This will be measured with Powertop[10]. This tool can output a power consumption estimation for every running process.

4 Results

Next, we will describe the obtained results from the integration and testing of SealFS in the autonomous robots.

Table 2. Descriptive Statistics - Throughput in the proposed scenarios.

	Audio bandwidth (kB/s)				Video bandwidth (kB/s)				Navigation bandwidth (kB/s)			
	summit_ext4	summit_sealfs	tiago_ext4	tiago_sealfs	summit_ext4	summit_sealfs	tiago_ext4	tiago_sealfs	summit_ext4	summit_sealfs	tiago_ext4	tiago_sealfs
Valid	896	896	886	872	890	895	880	860	331	332	360	359
Missing	0	0	0	0	0	0	0	0	0	0	0	0
Mode	119.870	119.870	138.310	138.750	7662.150	7565.330	10670.080	10711.040	7.720	7.720	7.540	7.520
Mean	118.101	120.306	135.645	136.896	9719.360	9307.038	10691.642	10419.843	11.649	11.690	7.840	8.266
Std. Deviation	3.555	5.053	6.283	6.806	1017.225	1226.905	907.653	795.380	3.932	3.976	1.385	2.459
Minimum	93.730	91.740	122.100	120.700	7506.720	7274.770	7680.000	7772.160	3.840	3.830	3.690	3.730
Maximum	127.720	135.470	147.020	154.220	10334.540	10378.170	13137.920	13178.880	19.310	15.490	11.330	15.150
25th percentile	115.960	116.010	128.043	131.037	10102.090	7749.100	10434.560	10096.640	7.720	7.720	7.500	7.500
50th percentile	119.760	119.790	138.510	138.805	10210.390	10094.500	10670.080	10659.840	11.580	15.320	7.520	7.520
75th percentile	119.890	123.630	139.420	142.433	10274.325	10317.040	10792.960	10762.240	15.440	15.440	7.540	7.540
Sum	105818.320	107793.970	120181.610	119373.720	$8.650 \times 10^{+6}$	$8.330 \times 10^{+6}$	$9.409 \times 10^{+6}$	$8.961 \times 10^{+6}$	3855.690	3881.110	2822.520	2967.650

The analysis of throughput data across different bandwidth categories provides valuable insights into the performance of SealFS in comparison to Ext4 on two distinct robotic platforms: Summit and Tiago. In the audio bandwidth category, SealFS demonstrates a slight improvement in mean throughput for both robots compared to Ext4. However, this enhancement is accompanied by an increase in variability, suggesting potential trade-offs between performance and consistency. For video bandwidth, SealFS exhibits mixed results, with lower

mean throughput observed for Summit but higher for Tiago compared to Ext4. Notably, SealFS introduces greater variability in performance, particularly evident in Summit. In the navigation bandwidth category, SealFS generally presents marginal improvements in mean throughput for both robots, with comparable variability to Ext4. These findings underscore the nuanced impact of SealFS integration on data throughput across diverse operational contexts, highlighting the need for careful consideration of trade-offs between performance enhancements and stability in autonomous robotic systems. Table 2 shows the results for three scenarios.

Next, we will describe the obtained results from the integration and testing of SealFS in the autonomous robots.

Table 3. Descriptive Statistics - Latency in Audio test

	time (μs)			
	summit_ext4_write	summit_sealfs_write	tiago_ext4_write	tiago_sealfs_write
Valid	19450	18627	17040	16496
Mode	18.166	66.664	11.115	36.643
Mean	17.589	71.513	15.277	71.506
Std. Deviation	3.982	5.430	8.008	32.827
Minimum	9.001	58.409	8.003	26.227
Maximum	45.337	149.267	277.732	293.878
25th percentile	15.418	68.346	11.135	41.079
50th percentile	17.562	70.803	13.479	74.326
75th percentile	19.543	73.463	17.543	89.801
Sum	342100.189	$1.332 \times 10^{+6}$	260311.746	$1.180 \times 10^{+6}$

The latency data across the three tests, namely Audio, Video, and Navigation, provides a comprehensive insight into the performance of SealFS and Ext4 file systems in various operational contexts within autonomous robotics. In the Audio test, both Summit and Tiago robots exhibit a notable increase in write latency with SealFS integration compared to Ext4. The mean write latency for Summit is 17.589μs with Ext4 and 71.513μs with SealFS, indicating a substantial increase in latency with SealFS integration. This trend is consistent across the mode, median, and 75th percentile measurements.SealFS also exhibits higher variability in write latency, with a larger standard deviation (32.827μs) compared to Ext4 (8.008μs). This increase in latency is consistent across all measured parameters, indicating a significant impact on data processing speed (Table3).

Similarly, in the Video test (Table 4), SealFS integration results in a substantial rise in write latency for both Summit and Tiago robots. The variability in write latency is particularly pronounced, suggesting potential challenges in maintaining consistent performance levels, especially in video-intensive tasks.

However, the Navigation test (Table 5 presents a more nuanced picture, with SealFS demonstrating higher write latency compared to Ext4 in Tiago, while showing comparable performance to Ext4 in Summit. This disparity underscores the importance of evaluating the performance impact of SealFS integration across different robotic tasks, as the implications may vary depending on the specific operational requirements and hardware configurations.

Table 4. Descriptive Statistics - Latency in Video Test

	time (μ s)			
	summit_ext4_write	summit_sealfs_write	tiago_ext4_write	tiago_sealfs_write
Valid	2733	3272	3902	4240
Mode	1387.584	14853.500	1035.439	10518.330
Mean	1513.815	15520.996	1318.313	11412.904
Std. Deviation	547.953	723.789	1235.535	1106.017
Minimum	1363.485	14603.020	625.503	9701.663
Maximum	6531.803	22898.950	16741.160	20321.590
25th percentile	1404.792	15059.908	1081.766	10805.368
50th percentile	1421.612	15325.030	1146.722	11169.010
75th percentile	1451.222	15488.157	1232.809	11692.485
Sum	$4.137 \times 10^{+6}$	$5.078 \times 10^{+7}$	$5.144 \times 10^{+6}$	$4.839 \times 10^{+7}$

Table 5. Descriptive Statistics - Latency in Navigation Test

	time (μs)	
	summit_sealfs_write	tiago_sealfs_write
Valid	327	346
Missing	0	0
Mode	94.353	75.937
Mean	93.356	90.867
Std. Deviation	11.862	18.511
Minimum	72.875	67.443
Maximum	148.174	159.787
25th percentile	86.987	77.046
50th percentile	90.028	82.279
75th percentile	96.704	105.529
Sum	30527.356	31439.834

Besides it also analyzed various parameters associated with SealFS integration and impact within autonomous robotic systems.

– CPU Usage: Through perf monitoring, a consistent CPU usage of 0.6% has been observed across different scenarios, indicating stability in CPU usage.

Table 6. Descriptive Statistics for Power Consumption

	Usage (μs) [sealfs]	Pw Estimate (μW) [sealfs]
Valid	8	8
Missing	0	0
Mode	2.400	171.000
Mean	4.025	364.375
Std. Deviation	0.870	125.518
Minimum	2.400	171.000
Maximum	5.200	501.000
25th percentile	3.700	285.000
50th percentile	4.050	396.000
75th percentile	4.575	456.250
Sum	32.200	2915.000

– Power Efficiency: The SealFS process demonstrates an estimated average power consumption of 3.1 mW under scenarios with high message transmission frequencies. Additionally, the tool estimates PC power consumption at 3.2W, indicating energy-efficient operation. Table6 overviews overall data.

Upon analyzing the performance between the evaluated robots, it becomes evident that the performance of the TiAGo robot surpasses that of the Summit model. Furthermore, a significant similarity in the behaviors and trends of the file system evaluated on both platforms is observed.

5 Discussion

Based on the results obtained, several key conclusions have been identified. In environments where disk stress is low, the tool's impact is negligible. The integration of SealFS results in a notable increase in write latency for both Summit and Tiago robots compared to Ext4. SealFS introduces higher variability in write latency, potentially impacting the predictability and responsiveness of robotic operations. Thus, SealFS's impact on latency warrants further consideration in real-time robotic applications. However, in high-load scenarios, although there is an increase in latency, this added time is measured in microseconds and does not present any significant issues for real-world deployment.

The tool does not result in any significant increase in CPU usage, suggesting that it operates efficiently without imposing additional processing burdens on the system. Additionally, there is no detectable increase in file size when using the tool, indicating that it does not add any overhead in terms of storage requirements. Furthermore, the tool incurs minimal additional energy consumption in percentage and nominal terms. This slight increase in energy usage is minor and unlikely to significantly impact overall system performance or efficiency.

6 Conclusions

In summary, the tool integration onboard of robots demonstrates minimal impact on system performance and resource usage under various conditions, with some minor considerations for functionality and energy consumption. Further work could focus on addressing the limitation related to folder creation to enhance organizational capabilities. Additionally, exploring optimization opportunities to reduce even the minor latency and energy consumption observed could further improve the tool's efficiency and user experience.

Acknowledgements

We gratefully acknowledge the financial support of Grant DMARCE Project: EDMAR PID2021-126592OB-C21 – CASCAR PID2021-126592OB-C22 funded by MCIN/AEI/10.13039/501100011033 and by ERDF A way of making Europe; and of the Recovery, Transformation, and Resilience Plan, financed by the European Union (Next Generation) thanks to the TESCAC project (Traceability and Explainability in Autonomous Systems for improved Cybersecurity) granted by INCIBE to the University of León.

References

1. A. Botta, S. Rotbei, S. Zinno, and G. Ventre, "Cyber security of robots: A comprehensive survey," *Intelligent Systems with Applications*, p. 200237, 2023.
2. E. Soriano-Salvador and G. Guardiola-Múzquiz, "Sealfs: Storage-based tamper-evident logging," *Computers & Security*, vol. 108, p. 102325, 2021.
3. 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.
4. O. A. Khashan, N. M. Khafajah, W. Alomoush, M. Alshinwan, S. Alamri, S. Atawneh, and M. K. Alsmadi, "Dynamic multimedia encryption using a parallel file system based on multi-core processors," *Cryptography*, vol. 7, no. 1, 2023.
5. C. Cho, Y. Seong, and Y. Won, "Mandatory access control method for windows embedded os security," *Electronics*, vol. 10, no. 20, 2021.
6. Y. S. Seong, C. Cho, Y. P. Jun, and Y. Won, "Security improvement of file system filter driver in windows embedded os.," *Journal of Information Processing Systems*, vol. 17, no. 4, 2021.
7. M. O. Arregi and E. L. Secco, "Operative guide for 4-wheel summit xl mobile robot set-up," *Acta Scientific COMPUTER SCIENCES Volume*, vol. 5, no. 4, 2023.
8. J. Pages, L. Marchionni, and F. Ferro, "Tiago: the modular robot that adapts to different research needs," in *International workshop on robot modularity, IROS*, vol. 290, 2016.
9. S. Rostedt, "Finding origins of latencies using ftrace," *Proc. RT Linux WS*, 2009.
10. S. Benedict, "Energy-aware performance analysis methodologies for hpc architectures—an exploratory study," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1709–1719, 2012.