Securing Robots: An Integrated Approach for Security Challenges and Monitoring for the Robotic Operating System (ROS)

Sean Rivera, Radu State
SnT, University of Luxembourg, Luxembourg
{sean.rivera, radu.state}@uni.lu

***Abstract*—Robotic systems are becoming an ever-increasing part of everyday life due to their capacity to carry out physical tasks on behalf of human beings. Found in nearly every facet of our lives, robotic systems are used domestically, in small and large-scale factories, for the production and processing of agriculture, for military operations, to name a few. The Robotic Operating System (ROS) is the standard operating system used today for the development of modular robotic systems. However, in its development, ROS has been notorious for the absence of security mechanisms, placing people in danger both physically and digitally. This dissertation summary presents the development of a suite of ROS tools, leading up to the development of a modular, secure framework for ROS. An integrated approach for the security of ROS-enabled robotic systems is described, to set a baseline for the continual development to increase ROS security. The work culminates in the ROS security tool *ROS-Immunity*, combining internal system defense, external system verification, and automated vulnerability detection in an integrated tool that, in conjunction with Secure-ROS, provides a suite of defenses for ROS systems against malicious attackers.**

***Index Terms*—robotics, ROS, security, integrated, dissertation**

I. Introduction

Robotics is a diverse field involving the design, engineering, and use of robots to assist humans with a wide variety of tasks. Robotics is a sub-field of the field of cyber-physical systems, which encompasses all devices with a physical component that are controlled by a digital component. In particular, robots are characterized by a particular degree of movement and autonomy to perform tasks [1], generally consisting of sensors and software that translate the digital world into the physical, and the physical world into the digital. A popular example of the marriage between robot's cyber-physical components is self-driving cars, where large and complex physical components are combined with software to create autonomous tools that can perform tasks equally or more adeptly than humans. These systems utilize both physical cues from the environment observed via sensors, and act upon those cues with adaptations in the functioning of the car as determined by software.

Robots are observed plentifully in everyday life, with applications including, but not limited to, manufacturing, transportation, medicine, military, industrial, agricultural, and domestic products [2]. The penetration of robotics into society has vastly increased in the past 30 years due to the wide variety of tasks that they can perform [3]. In western society, humans interact with a plethora of robots and robotic systems every day, from interacting with domestic robots within their homes, utilizing or interacting with vehicles running robotic systems, or benefiting from services run by robots by purchasing products manufactured or delivered by robots. Robotics held a market-value of \$115 billion in 2019, with an expected CAGR[1] of 25% [4]. Between 2000-2016, 1.6 million jobs worldwide were lost due to automation by robots, with 20 million lost expected by 2030 [5].

All robots must utilize control software to function, the most popular of which is The Robotic Operating System (ROS). ROS enables the development and distribution of robotic systems. There are several security challenges that ROS faces, from securing nodes and ROS middleware from attackers, to securing the sensors themselves from data attacks. However, the security framework of ROS has received much scrutiny in the previous years, as there are major design flaws that have been left unaddressed. As ROS gains popularity, increasing security is vitally important to prevent damages, especially physical damages.

As robotic systems become more common in everyday life, there are growing concerns about the potential impact security vulnerabilities may have on the real-world. This dissertation focuses on security challenges and solutions for ROS systems. The development of an integrated ROS security tool is described in detail, presented with the chronological development of the suite of tools combined to create a full security solution. The solution is broken down into one of three components: internal system defense, external system verification, and automated vulnerability detection, to address vulnerabilities in all sensitive areas of robotic security, even providing a solution for identifying unknown, novel vulnerabilities. *ROS-Immunity*, an integrated approach for the security of ROS-enabled robotic systems, is described, to set a baseline for the continual development to increase ROS security. The work culminates in the ROS security tool that addresses all three areas of defense with an integrated tool that, in conjunction with Secure-ROS, provides a suite of defenses for ROS systems against malicious attackers.

978-3-903176-32-4 © 2021 IFIP

[1]Compound Annual Growth Rate

II. Background

The Robotic Operating System (ROS) is a collection of software libraries that enables communication of both (abstracted) hardware and (pure) software components to develop robotic systems. It is predicted that ROS centered systems will make up the majority of robotic systems within the next five years, both in commercial and academic settings [6]. Even among non-ROS systems, many design similarities exist wherein similar methods can be applied [7]. ROS is an open-source project, and it has a vibrant community with thousands of developers and over nine thousand unique packages available[2]. While the developers of ROS have added security features in recent years, most notably SROS and Secure-ROS, the security solutions in place do not provide adequate protection against all types of vulnerabilities [8].

ROS enables the development of robotic systems at a fine-grained scale. Briefly, the design paradigm behind ROS is that of a Publish-Subscribe model, where a master keeps track of the state of the system while applications called *nodes* directly interact with each other through *topics*. A node is a process that performs a specific computation and controls a part of the robot's operation. There can be a node computing a trajectory, a node moving the wheels, a node controlling a camera, *etc.*. A robotic system usually consists of many nodes that communicate with each other by passing messages. Messages define clean and consistent interfaces. A **ROS topic** is an implementation of a channel in which messages are passed in a publish-subscribe model. Topics act as a named bus where nodes can join as either a publisher, a subscriber, or both. When a publisher node sends a message over a topic, every subscriber node to that topic receives a copy. A particular node, namely the master node, is always present in a ROS-enabled system.

Vulnerabilities against the robotic operating systems can be broadly broken down into three separate categories: vulnerabilities in the nodes that were written by developers, vulnerabilities in the ROS middleware itself, and vulnerabilities inherent to the robot system. Vulnerabilities in the nodes are ubiquitous, as with most software systems, as they are usually maintained by small groups or individuals who are more focused on functionality then security [9]. As such it is unwise for any defensive security system to depend on the integrity of all nodes since they could be compromised at any moment. Instead, the defenses should focus on maintaining protection for the whole system, while limiting attackers opportunities to exploit individual nodes. One of the most critical vulnerabilities in the ROS middleware is that the ROS master is a singular point of failure within ROS. Without the master, nodes would not be able to find each other, exchange messages, or invoke services. If the master node is compromised, it is functionally identical to the whole system being compromised. That said, the master node has no mechanism for enforcing good behavior among nodes, and in fact, cannot even verify that its internal model of the robot's software[3] matches the actual executing software on the robot. This manipulation of the actual software, without the master becoming aware of it, forms the basis of many different attacks against ROS, as once an attacker can successfully manipulate the external ROS graph, they can take control of the robot, even in the presence of many security features [8]. Vulnerabilities inherent to the robot system take many forms, from power vulnerabilities to sensor attacks, and are the hardest to defend against directly.

ROS was initially created as a pure research project, by academics for other academics, without any security considerations. As it escaped from academic confinement into 'real-world' use cases, concerns about its security began to grow. Dieber *et al.* [10] [11] was the first serious security analysis of ROS, however, its findings were dismal. The core ROS system had effectively no security infrastructure, and it was trivial for an attacker to take control of a robot and use it to inflict damage.

Several researchers noted these issues as well and analyzed the security challenges in robotics security, particularly among ROS systems. Recent work highlighted a number of security threats against ROS [12] [13] [14]. The lack of proper security tools in ROS exposes systems to a variety of security concerns, including but not limited to, sensor spoofing(the action of disguising a communication from an unknown source as being from a known one [15]), node impersonation, denial of service, man-in-the-middle, and privilege escalation attacks. It was found that robots targeted by sensor spoofing attacks were able to force an incorrect behavior and undermine the success and safety of critical operations [16], potentially causing physical accidents in factories or other physical spaces. It became clear that an insecure robot could irreversibly damage the physical environment in which it is operating, including being harmful to human beings, and few security tools were available to prevent large breaches. The Open Robotics Foundation, creators of ROS, acknowledges the security concerns of ROS.

The ROS group also began work on SROS [13], a security suite for ROS systems. However, this framework is still under development and is not fully implemented. Noticing these issues, Dieber *et al.* developed ROSPenTo [17], an automated testing tool for ROS to demonstrate how insecure the system is. Additionally, SRI developed SecureROS, which provides crypto and limits nodes to a whitelist for connections, but does not protect against external attackers connecting to nodes [18]. The release of ROS2 mitigated some security concerns, introducing a DDS security standard, but the system is still largely insecure with many areas of vulnerabilities left unaddressed [19]. In particular, current solutions for the security of ROS and ROS2 do not protect compromised nodes or denial of service, do not support reactive policies that are updated dynamically at runtime, and cannot enforce low-level granularity network policies.

[2]https://metrics.ros.org/

[3]This is called a rosgraph.

III. Challenges and Justifications

To effectively monitor such robotic systems at run-time, special monitoring software is required. However, not many concepts and solutions have been proposed to monitor the components of robotic systems, especially in ROS. Security in ROS is an active concern, with few fully implemented solutions [10]. Any such monitoring systems have to be flexible, scalable, and secure, without sacrificing run-time. Currently, most monitoring software is processing intensive and slow, a double disadvantage on cyber-physical systems, particularly robotic systems. There are no current solutions that offer both monitoring and security without a severe processing impact. A consistent set of security tools is crucial for security research as it provides a foundation for further development. Additionally, the existence of such tools allows a developer to test their systems against similar real-world attacks and more effectively develop secure reliable systems.

This work describes the development of a fully integrated, secure security solution for ROS systems. The goal of this dissertation was in pursuit of such a system that would provide a security framework for ROS with the following design guidelines in mind: 1) the configurations should 'speak' the same language as ROS (i.e. developers of ROS should not have to learn a new system to add security features), 2) the system should be able to efficiently communicate issues that it could not address to users, such as newly discovered or unaddressed vulnerabilities, and 3) the system should have a highly modular design (a key feature of ROS functionality). Additionally, in this dissertation, the design paradigm addressed to ROS systems is broken down into two: centralized and decentralized systems. It is assumed that these are two main design paradigms that are applied to implement a multi-robot system with ROS. The primary difference is the number of master nodes in the system, with centralized systems utilizing only one master node while decentralized systems more than one. Both design paradigms allow for unique functionality while carrying their challenges, and thus it is vital to create a system that can address both types of systems.

It was also of interest to account for three core areas of security concerns with ROS systems: internal attacks, external attacks, and vulnerabilities (known and unknown). A significant goal of this work was to create a system that could account for these three areas in one tool, without affecting the users' system.

IV. Development of the Solution

To design a security solution for ROS systems, the logical first step was to create a reconnaissance tool. Such a tool would allow researchers to evaluate the potential risks exposed by the vulnerabilities in ROS. Additionally, it would serve as an excellent testing tool for any security solution. As such, research began with the development of *ROSploit*, an automatic vulnerability scanner and testing framework for ROS [20]. Similar to ROSPenTo, *ROSploit* contains fingerprints for many different exploits that can be executed on a ROS system and the ability to automatically scan a network for vulnerable ROS systems. Unlike ROSPenTo however, *ROSploit* includes the ability to identify specific nodes and store vulnerability fingerprints for those nodes.

Once *ROSploit* was successfully implemented and tested, the research could begin on a defensive solution to address ROS's issues. Looking at the heavily networked nature of ROS, efforts were focused on securing ROS's network infrastructure as an initial step, and then extended reach from there. ROS-Defender, a tool built to take advantage of the flexibility of Software Defined Networking (SDN), was the initial result of these explorations [21]. ROS-Defender addressed many of the vulnerabilities discussed in Section II, including enforcing the Master's rosgraph, anomaly detection, and user-specified traffic filtering.

While ROS-Defender was an effective security system for ROS, it did come with some drawbacks that would limit its adaption. Chiefly among those was that ROS-Defender required specialized SDN capable hardware and that the added overhead could interfere with some ROS system's operations. As such, the novel eBPF/XDP functionality in the Linux kernel [22] was leveraged to create *ROS-FM* [23]. *ROS-FM* is a dramatic improvement on ROS-Defender in terms of both performance and functionality. Additionally, *ROS-FM* is far more modular in its design, allowing it to be customized to match the target ROS-system with both security and monitoring functionalities available to users.

After testing *ROS-FM*, it was discovered that it was possible to address many different types of ROS vulnerabilities, provided that the component under attack was running on the robot. That said, there is a whole class of vulnerabilities that do not leave traces on the targeted system, such as sensor attacks. To address these, a new approach was required. Because sensor attacks involve distorting the robot's view of the world, through subtle or not so subtle means, any security solution needed to be able to maintain an accurate internal representation to detect them. To do so, focus turned to machine learning [24]. After experimentation, a system was devised that protected against sensor attacks. It involved the use of autoencoders to maintain a consistent historical model of the world using reconstruction error to detect anomalies [25]. This solution was integrated into *ROS-FM* as a plug-in to provide a novel security solution for sensor attacks.

The previous solutions provided a full suite of security tools to defend against known attacks, however, the area of unknown attacks was unaccounted for. This gap in ROS security research influenced the development of *DiscoFuzzer*, a novel technique for finding novel vulnerabilities in ROS systems [26]. Building on the research on robot state analysis discussed in the previous paragraph, it was discovered that robots were dependent upon the continuity of their physical space. This was utilized to hunt for vulnerabilities in the areas where the space became discontinuous. Using several methods of numeric analysis and feature space exploring, *DiscoFuzzer* was developed as a tool that was able to find several novel vulnerabilities that could have led to the malicious manipulation of robots. Applying fuzzing technology, *DiscoFuzzer*

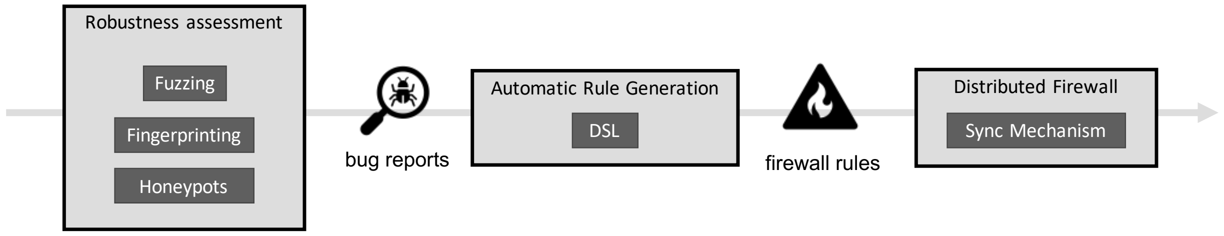

Fig. 1. *ROS-Immunity*'s three components to implement an integrated security mechanism in ROS system addressing the security gaps in current ROS systems.

was developed to address the security gaps in vulnerability detection, a very under-researched area in ROS security.

The development of the above-mentioned tools demonstrated the need for ROS security tools, especially those that limit the impact on the system and user implementation of them. Development continued to combine these tools into an integrated solution, described in the next section.

V. ROS-IMMUNITY

The pieces of work discussed in the previous section led to unique insights into the requirements of developing an adequate security solution for ROS systems. These insights were used to design *ROS-Immunity*, a comprehensive security system for ROS. *ROS-Immunity* provides a light security solution for ROS that is capable of covering the whole chain from discovering vulnerabilities to protecting from attacks [27]. *ROS-Immunity* consists of three components: robustness assessment, automatic rule generation, and distributed defense with a firewall (Figure 1). The robustness assessment discovers new vulnerabilities of the target through the combination of different testing and analysis techniques. Then, these vulnerabilities are encoded in a domain-specific language and automatically fed into the distributed firewall. Finally, the firewall runs on the robots of the target system and can detect and block malicious messages or other malicious behavior. By addressing these areas, *ROS-Immunity* provides an all-encompassing security solution that addresses internal system attacks, external system verification, and automatic vulnerability detection.

To account for all vulnerabilities that could affect a ROS system, a fully integrated tool was needed. The most critical components of such a security tool is the ability to discover new vulnerabilities in a system, and the ability to determine if any known vulnerabilities exist within the system. One example of this is in the robustness assessment component of *ROS-Immunity*, which combines traditional fuzzing [28], our ROS specific fuzzing [26], and a database of other ROS vulnerabilities [29] to collect and store vulnerabilities of the ROS system. It then disseminates these vulnerabilities to the firewall components of each robot, in order to apply for protection. Additionally, it regularly leverages *ROSploit* to ensure that the firewall is correctly blocking all known vulnerabilities.

The firewall is the foundation upon which all other security features are built. Presented in Figure 2, its design was developed based on the type of system is protected. In both cases, its primary task is to enforce the master node's rosgraph on the internal state of the robot, i.e. ensure that all communications within the robot only take place on channels the master is aware of. It is based on the previous work of *ROS-FM*, and is an eBPF/XDP based firewall, which efficiently filters out all traffic that matches known vulnerabilities. As it runs on each robot, it looks for anomalies in the communications and behavior of the robot, and it can even raise an alert if it detects a suspected attack. This anomaly detection can even be extended to protect against external sensor attacks.

A. Results

ROS-Immunity was demonstrated with four use-cases, addressing robotic systems from both centralized and decentralized systems to demonstrate the effectiveness of the solution. The goal of *ROS-Immunity* for centralized systems is to ensure that the master node is always protected and aware of its robots, to quickly identify compromised robots to restore them to a functioning state or a fail-safe state until the user can intervene. On the contrary, the main concern with decentralized ROS systems is the difficulty with applying any cryptographic key-based system, as ROS1 relies on packages and workarounds and does not have direct support for these systems, while ROS2 includes limited support. In this case, the goal is to limit the damage that anyone robot can do and maintain a record of all activity. *ROS-Immunity* is tested on two centralized systems, a self-driving car and a centralized factory, and two decentralized systems, a robotic swarm and decentralized factory.

In all four cases, low average CPU, memory, and network overhead was observed, except in the case of small swarms and large centralized factories. In the case of the small swarm, this was due to the cost of sharing information between different robots and reconstructing the trusted network as the underlying mesh network shifts. As the size of the swarm increased, the mesh network became more stable and the overhead of *ROS-Immunity* was vastly reduced. Meanwhile, in the case of the large centralized factory, the central master node created a bottleneck leading to a noticeable increase in the overhead. In all other cases, *ROS-Immunity* was demonstrated to provide more benefit than harm, increasing the security of the systems without impacting their functionality.

Owing to the nature of robotic systems, power usage is the most important metric for calculating overhead. In Figure

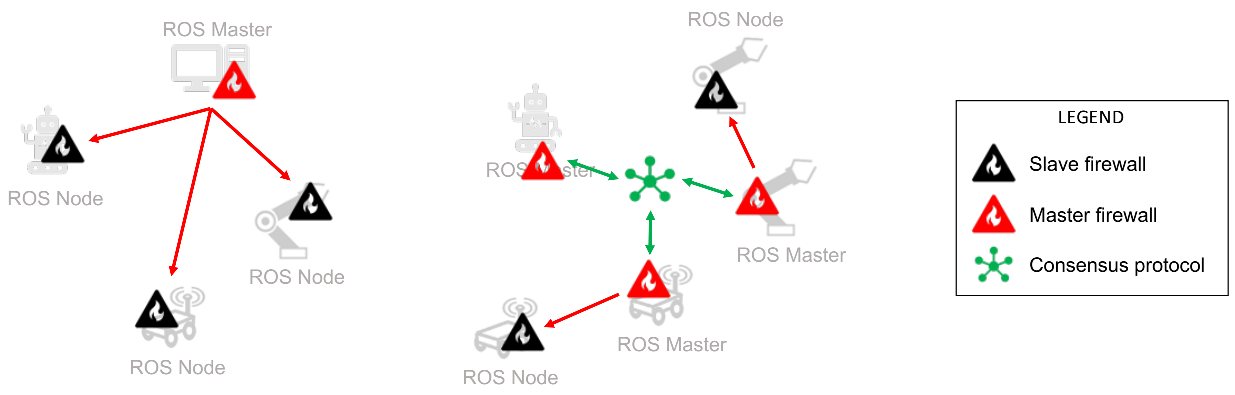

Fig. 2. Integrated Firewall Architecture for centralized (on the left) and decentralized (on the right) ROS system designs.

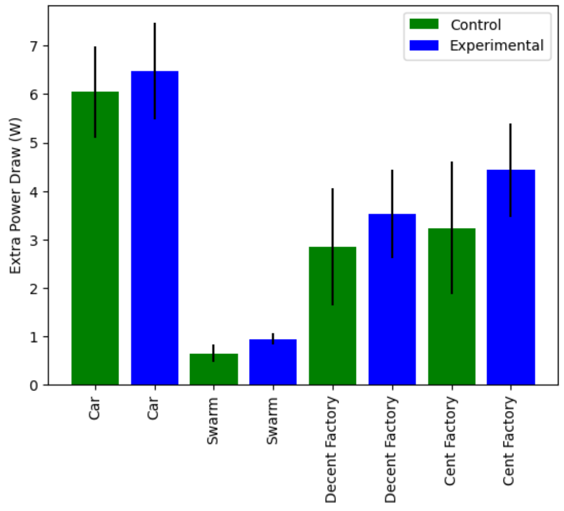

Fig. 3. Normalized Power Overhead

3 the per second power requirements of *ROS-Immunity* were calculated versus the control system using the ARM embedded power formula found in Mao *et al.* [30][4]. The power draw of each robot in the system was normalized. Overall, *ROS-Immunity* had an average power cost of 7% for the car, 13% for the Decentralized Factory, 18% for the Centralized factory, and 16% for the swarm. This additional power overhead is very low compared to the power requirements inherent in hardening the network with cryptography [31].

Additionally, tests were conducted to validate *ROS-Immunity* against unknown attacks. Experiments showed that both the centralized systems were able to issue a reset of the system within 1.31 seconds ($\mu = 1.16$, $\sigma = .29$ seconds) in the worst case. Additionally, the decentralized systems were able to exclude a compromised system within 2.4 seconds in the worst case ($\mu = 1.71$, $\sigma = 1.39$ seconds). While the decentralized was slower than the centralized, it was still able to react and exclude a compromised system before any system connected to it could be compromised. These results indicated that the system was readily able to stop compromised systems in both cases, preventing an attacker from gaining ample access to either type of system.

[4]GPU power was excluded from the calculations

VI. Discussion and Conclusions

With the rapidly increasing presence of robotic systems in everyday life, it is vitally important for such systems to be secure to prevent damage in the real-world. Of particular importance is the security of The Robotic Operating System, one of the most popular operating systems in use today. This dissertation focuses on security concerns, challenges, and solutions for ROS systems. This work culminated into *ROS-Immunity*, an integrated, comprehensive security system for ROS capable of addressing the entire chain of security gaps in current research. Specifically, this tool addressed ROS security by providing a full security system addressing internal system defense, external system verification, and automatic vulnerability detection with low-overhead. *ROS-Immunity* provides the ability to discover new vulnerabilities, harden the system against active attacks, and provide reliable security for users.

Acknowledgment

Research supported by EC H2020 Project CONCORDIA GA 830927 and the Luxembourg Directorate of Defence.

References

[1] “Robots and robotic devices — Vocabulary,” International Organization for Standardization, Geneva, CH, Standard, Mar. 2012.
[2] M. Ben-Ari and F. Mondada, “Robots and their applications,” in *Elements of Robotics*. Springer, 2018, pp. 1–20.
[3] M. Shukla and A. N. Shukla, “Growth of robotics industry early in 21st century,” *International Journal of Computational Engineering Research*, vol. 2, no. 5, pp. 1554–1558, 2012.
[4] Robotics Business Review, “Global spending on robots, drones to top $115b in 2019, says idc),” https://www.roboticsbusinessreview.com/manufacturing/global-spending-on-robots-drones-to-top-115b-in-2019-says-idc/, Tech. Rep., [Online; accessed 30-August-2019].
[5] E. Cone and J. Lambert, “How robots change the world - what automation really means for jobs, productivity and regions,” Jun 2019. [Online]. Available: https://www.oxfordeconomics.com/recent-releases/how-robots-change-the-world

[6] D. Petrara, "The rise of ros: Nearly 55% of total commercial robots shipped in 2024 will have at least one robot operating system package," 2019. [Online]. Available: https://www.bloomberg.com/press-releases/2019-05-16/the-rise-of-ros-nearly-55-of-total-commercial-robots-shipped-in-2024-will -have -at - least -one - robot- operating- system- package

[7] A. Elkady and T. Sobh, "Robotics middleware: A comprehensive literature survey and attribute-based bibliography," *Journal of Robotics*, vol. 2012, 2012.

[8] B. Dieber, R. White, S. Taurer, B. Breiling, G. Caiazza, H. Christensen, and A. Cortesi, "Penetration testing ros," in *Robot Operating System (ROS)*. Springer, 2020, pp. 183–225.

[9] M. Pichler, B. Dieber, and M. Pinzger, "Can i depend on you? Mapping the dependency and quality landscape or ROS packages," in *Proceedings of the 3rd International Conference on Robotic Computing*. IEEE, Feb. 2019.

[10] B. Dieber, S. Kacianka, S. Rass, and P. Schartner, "Application-level security for ros-based applications," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 4477–4482.

[11] B. Dieber, B. Breiling, S. Taurer, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Robotics and Autonomous Systems*, vol. 98, pp. 192 – 203, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889017302762

[12] S.-Y. Jeong, I.-J. Choi, Y.-J. Kim, Y.-M. Shin, J.-H. Han, G.-H. Jung, and K.-G. Kim, "A study on ros vulnerabilities and countermeasure," in *Proceedings of the Companion of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '17. New York, NY, USA: ACM, 2017, pp. 147–148. [Online]. Available: http://doi.acm.org/10.1145/3029798.3038437

[13] R. White, H. I. Christensen, and M. Quigley, "SROS: securing ROS over the wire, in the graph, and through the kernel," *CoRR*, vol. abs/1611.07060, 2016. [Online]. Available: http://arxiv.org/abs/1611.07060

[14] G. W. Clark, M. V. Doran, and T. R. Andel, "Cybersecurity issues in robotics," in *2017 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, March 2017, pp. 1–5.

[15] N. Hubballi and N. Tripathi, "An event based technique for detecting spoofed ip packets," *Journal of Information Security and Applications*, vol. 35, pp. 32 – 43, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2214212617301692

[16] D. Davidson, H. Wu, R. Jellinek, V. Singh, and T. Ristenpart, "Controlling uavs with sensor input spoofing attacks," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, 2016. [Online]. Available: https://www.usenix.org/conference/woot16/workshop-program/presentation/davidson

[17] "Hacking a mir robot with rospento." [Online]. Available: https://bernharddieber.com/post/mir-hacking-video/

[18] "Secure ros," http://secure-ros.csl.sri.com/, accessed: 2019-03-01.

[19] ROS, "ROS 2 Overview," https://index.ros.org/doc/ros2/, [Online; accessed 1-June-2020].

[20] S. Rivera, S. Lagraa, and R. State, "Rosploit: Cybersecurity tool for ROS," in *3rd IEEE International Conference on Robotic Computing, IRC 2019, Naples, Italy, February 25-27, 2019*, 2019, pp. 415–416.

[21] S. Rivera, S. Lagraa, C. Nita-Rotaru, S. Becker, and R. State, "Ros-defender: Sdn-based security policy enforcement for robotic applications," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 114–119.

[22] A. Deepak, R. Huang, and P. Mehra, "ebpf/xdp based firewall and packet filtering," in *Linux Plumbers Conference*, 2018.

[23] S. Rivera, A. K. Iannillo, S. Lagraa, C. Joly, and R. State, "Fast monitoring for the robotic operating system," in *Proceedings of the 25th International Conference on Engineering of Complex Computer Systems*, 2021.

[24] S. Lagraa, M. Cailac, S. Rivera, F. Beck, and R. State, "Real-time attack detection on robot cameras: A self-driving car application," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 102–109.

[25] S. Rivera, S. Lagraa, A. K. Iannillo, and R. State, "Auto-encoding robot state against sensor spoofing attacks," in *2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. IEEE, 2019, pp. 252–257.

[26] S. Rivera, A. K. Iannillo, and R. State, "Discofuzzer: Discontinuity-based vulnerability detector for robotic systems," 2020.

[27] S. Rivera, A. K. Iannillo *et al.*, "Ros-immunity: Integrated approach for the security of ros-enabled robotic systems," 2020.

[28] M. Zalewski, "american fuzzy lop technical "whitepaper"","," *URL: http://lcamtuf. coredump. cx/afl/technical_details. txt*, 2015.

[29] V. M. Vilches, L. A. Kirschgens, A. B. Calvo, A. H. Cordero, R. I. Pisón, D. M. Vilches, A. M. Rosas, G. O. Mendia, L. U. S. Juan, I. Z. Ugarte *et al.*, "Introducing the robot security framework (rsf), a standardized methodology to perform security assessments in robotics," *arXiv preprint arXiv:1806.04042*, 2018.

[30] Y. Mao, "Detailed power measurement with arm embedded boards," 2018.

[31] F. J. Rodríguez-Lera, V. Matellán-Olivera, J. Balsa-Comerón, M. Guerrero-Higueras, and C. Fernández-Llamas, "Message encryption in robot operating system: Collateral effects of hardening mobile robots," *Frontiers in ICT*, vol. 5, p. 2, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fict.2018.00002