machines

MDPI

Article

Runtime Verification for Anomaly Detection of Robotic Systems Security

Yunus Sabri Kirca [1], Elif Degirmenci [1,*], Zekeriyya Demirci [2], Ahmet Yazici [1], Metin Ozkan [1], Salih Ergun [3] and Alper Kanak [3]

[1] Department of Computer Engineering, Eskisehir Osmangazi University, Eskisehir 26048, Turkey
[2] Department of Software Engineering, Eskisehir Osmangazi University, Eskisehir 26048, Turkey
[3] ERARGE—Ergünler Co., Ltd. R&D Center, İstanbul 34768, Turkey
* Correspondence: edegirmenci@ogu.edu.tr

Abstract: Robotic systems are widely used in industry, agriculture, the inspection of infrastructure, and even in our daily lives. The safety and security of robotic systems have become a primary concern as their interaction with humans increases. In this context, attacks on robotic systems have increased for diversified field applications. It is necessary to accurately detect these abnormal events in these systems as soon as possible. However, these systems also need a runtime verification approach on whether they conform to the established specifications. In this study, runtime verification for anomaly detection methods is proposed for the security of the robot operating system (ROS). Firstly, an anomaly detection method is proposed to detect unexpected situations, such as the number of the received packages being decreased under DoS attacks. Then, a holistic runtime verification architecture is proposed for the anomaly detection method. This architecture consists of three major entities: a verification device, an attacker device, and a robotic platform without losing generality. In the verification device, ROSMonitoring and Oracle are used to implement runtime verification. The proposed architecture is verified through an experimental setup. It is shown that the architecture can be used for runtime verification of different anomaly detection algorithms. A discussion on the security of robotic systems is also presented.

Keywords: runtime verification; ROS; anomaly detection; security

check for updates

Citation: Kirca, Y.S.; Degirmenci, E.; Demirci, Z.; Yazici, A.; Ozkan, M.; Ergun, S.; Kanak, A. Runtime Verification for Anomaly Detection of Robotic Systems Security. *Machines* **2023**, *11*, 166. https://doi.org/10.3390/machines11020166

Academic Editor: Dan Zhang

Received: 6 December 2022
Revised: 10 January 2023
Accepted: 14 January 2023
Published: 25 January 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The use of robotic systems in industrial systems increases by an average of 30% every year, while the use in non-industrial systems increases by 52% [1]. However, as the usage areas are diversified, safety and security become primary problems [2]. The number of security attacks is also increasing due to digital accessibility. The robot operating system (ROS) is a critical digital accessibility tool in robotic systems. Many components in a robotic platform are integrated using ROS. However, it has entailed security concerns, and SROS [3] and ROS2 [4] have been presented to handle some security issues. However, current solutions are mostly about encrypting or signing messages. A security countermeasure against a volumetric attack is not included in these new versions of ROS. ROS-based studies ensure security by encryption methods of data or policy creation fields [5–8]. Even these security measurements are evaluated. ROS still has vulnerabilities against various attacks such as DoS, blackhole, and man-in-the-middle. The latest studies are focused on DoS attacks for ROS [9,10]. ROSPloit tool [9] is a security attack analysis tool for ROS. The security assurance of robotics systems' is important in both the in-design phase and runtime phase. Threat modeling is a phase that needs to be constantly checked in the system's runtime life cycle, starting from the in-design phase [11]. However, even if the threat models are evaluated, some situations are omitted due to time constraints. The omitted situations should be detected in runtime using anomaly detection. Anomaly detection

is increasingly used both in robotic systems [12] and in system security independently. Narayanan and Bobba studied anomaly detection on an industrial robotic arm for an ROS-based environment [12]. They determined the anomalous events as repeated movements of the industrial arms. Another study focused on detection of anomalies on an autonomous racer's sensor data built from a ROS [13]. Note that anomaly detection is also an active research topic in IoT systems' security [14]. In the literature, there are studies on anomaly detection and security attacks. However, the anomaly detection approaches have limitations such as training data importance and anomalous-yet-correct behavior [13]. Therefore, anomaly detection methods need to be verified. Runtime verification (RV) can be used to verify the anomaly detection method that works in complex and dynamic systems.

RV is a method of meeting the system's requests and determining whether it violates the specifications during the monitoring phase according to the stated specifications. Since states are defined for each state in formal methods, this causes a state explosion in complex situations. Therefore, RV methods are a very popular method for validating systems that are more unpredictable than formal methods [15]. RV is complementary to verification techniques and also disquoting feature is besides other verification techniques is it acts on runtime, and it is lightweight verification for systems in dynamically changing environments [16]. In recent years, various tools/frameworks have been suggested for RV [17]. The pioneering work ROSRV is an RV framework for ROS [5]. It may verify the system by intercepting and monitoring communication between publisher and subscriber for security and safety commands. Security verification is ensured by checking unencrypted messaging. However, there is no measurement in this study about denial-of service (DoS) attacks on the system. In environments where critical tasks are also carried out, attacks that may prevent data traffic such as DoS or blackhole attacks on ROS nodes may cause autonomous robots operating in industrial systems to deviate from their routes, mission disruptions, and even material damage such as collisions of autonomous robots. Furthermore, recent studies show that detection of these attacks is not as robust as before [18]. In addition, protection or detection systems could be fooled by this attack. Therefore, the necessity of verifying detection systems by another RV arises. Although ROS is widely used middleware for robotic systems, there are few security tools developed for it. Additionally, the increasing interactions of robotic systems with the environment also require using developed RV tools for security. One of the recent studies for RV of security is a ROS-based attack tool [19]. It is focused on a model-based runtime verification approach. To the best of the author's knowledge, RV for security of robotic system is an open research area considering the latest studies above.

In the present study, RV architecture is proposed for an anomaly detection algorithm in robotic systems' security. The proposed architecture for RV consists of a robotic platform, verification system, and attacker system. The robotic platform consists of three devices, namely ROS Master, controller, and simulation or physical robot, without losing generality. In the robotic platform, the robot control node sends the control locations of the robotic arm to GAZEBO in the simulation device. The simulation device is used to evaluate experiments; however, it could be replaced by a real robotic platform. All components of the robotic platform contain a packet loss tracker for detecting unexpected situations. For this setup, anomaly detection is also performed by the ROS Master without losing generality. The verification systems realize the RV of the anomaly detection method using specifications. The verification system consists of an RV data collector and ROSMonitoring. The RV data collector monitors the anomaly detection node for anomaly status and the attacker device for monitoring attack state. ROSMonitoring is used for evaluating the verification by specifications or temporal logic (TL) Oracle in itself [20]. ROSMonitoring logs the alert situations. The attacker system includes various attacks on the robotic platform. In the present study, three types of attacks, namely unauthorized publishing attack, general DoS attack, and SYN flood attack, are evaluated in the experiments. The attacker system could select any device as a target device in the robotic platform. During the experiments, the master device is selected as the target device by the ROS attack tool. This integration of the

both the control system and the verification system results in a reduced time required for the V&V process. This architecture also controls the security requirements with defined methods and security tools.

The remainder of this paper is organized as follows: Section 2 contains a literature review of verification for system security. Section 3 introduces RV architecture for the anomaly detection approach. Section 4 introduces the experimental environment and discusses the experimental results presented. Section 5 presents the conclusion by identifying the strengths and weaknesses of this paper and future work.

2. Verification for System Security

Formal verification is a process that ensures the correctness of the system by checking predetermined requirements. It is one of the most popular techniques to verify systems such as autonomous systems [21], robotic systems [22–24], or a combination of these [25]. It can be divided into some different subtitles such as model checking, theorem proving, and RV, which is the most widely used [25]. Unlike RV, the other methods have defects when verifying the systems at runtime and these methods are not sufficient to meet the requirements according to the given specifications [26]. Luckcuck et al. (2019) [25] presented a state-of-the-art review on formal specification and verification for autonomous robotics. Specifically, they comprehensively addressed and categorized the challenges. Although one of these verification methods could be sufficient, it is also possible to combine them. Webster (2020) [24] proposed an approach called corroborative verification and validation (V&V) of robot assistants in the context of human–robot interactions that combines several different V&V techniques. This combination enabled V&V methods to be performed at different stages of development. Halder et al. (2017) [23] proposed an approach for modeling and verifying ROS systems using real-time properties, focusing on the communication between nodes and especially the verification of safety and liveness properties. The physical robot Kobuki is used as a complex case study. Defining predetermined states in formal verification methods can make them unidentifiable for complex checks. In such cases, RV methods can be implemented for verifying the systems.

RV is a process that demonstrates whether the system satisfies or violates properties against given specifications at runtime [26]. If there is a violation caused by any component of the system, the user can be informed or predefined actions can be taken [27]. RV demands two inputs for a system to be inspected and a group of properties to be checked [17]. The specifications can be expressed as properties by using formal specification languages such as TLs [17,28]. Luo et al. (2018) [28] drew attention to the importance of an obstacle avoidance system for robotics. They conducted a case study to verify the safety strategy of collision avoidance of robots. They developed their own strategies under two subclasses: pre-contact to consider the avoidance of collision and post-contact to decrease likely damage. They also proposed a new method, namely dynamic parameter selection, to specify the dangerous distance between the robot and the obstacle. They chose JavaMOP as a model checker for the RV process. It allows for specifying properties as several specification languages like finite-state machine (FSM) and TLs. Hu et al. (2019) [26] suggested a new technique based on ROS, namely robots monitor on multilayer (RMoM), on a robot swarm by using RV. They aim to verify whether the system violates given specifications. They used discrete-time metric temporal logic (MTL) to express the properties of the behaviors of robot swarms. Intelligent vehicle safety is becoming an increasingly complex safety-critical system. To achieve cost-effective verification for intelligent vehicles, one study carried out a case study using RV to achieve safety aims [27], they tried to verify the interaction between two advanced driver assistance systems (ADAS) and demonstrated whether the RV can detect unintended behaviors. They used Unity and Simulink, which are simulation tools, to implement these systems and used the breach tool and signal temporal logic (STL) to verify intelligent vehicle safety. An et al. [29] used an approach with RV as an input for a machine learning model to achieve a more intelligent decision. They used the statistical model checker UPPAAL-SMC to verify the system against given safety properties. Moreover,

there are very few studies on DoS RV. Zeller [30] proposed a RV system for DoS attacks. He analyzed the DoS attacks on a hotel booking domain. He took as an argument DoS for the reservation cancellation problem. Adaptive RV is a method where the rules can evolve according to various input depending on changing situations [31]. There are also RV studies in different applications of IoT domains [31,32].

3. Runtime Verification Architecture

RV is necessary for anomaly detection methods since it is not possible to test all situations before going live due to various constraints. In the present study, an RV system is proposed to verify the anomaly detection method at runtime. The proposed RV system architecture consists of three parts: robotic platform, attacker system, and verification system. In the following subsections, the system architecture and details of the RV method are given consecutively.

3.1. Proposed System Architecture

ROS is a widely used robotic middleware in industry and academia. Despite its increasing use, ROS is not yet developed enough in the field of cyber-security. Anomaly detection is a method used to detect various attacks in the ROS environment. However, the anomaly detection methods may not be able to be fully tested before the runtime. The proposed RV system architecture verifies the anomaly detection method at runtime. It consists of three parts: robotic platform, attacker system, and verification system as shown in Figure 1. In this architecture, the verification system monitors both the robotic system and anomaly detection at runtime. The attacker system could generate attacks for any part of the robotic platform.

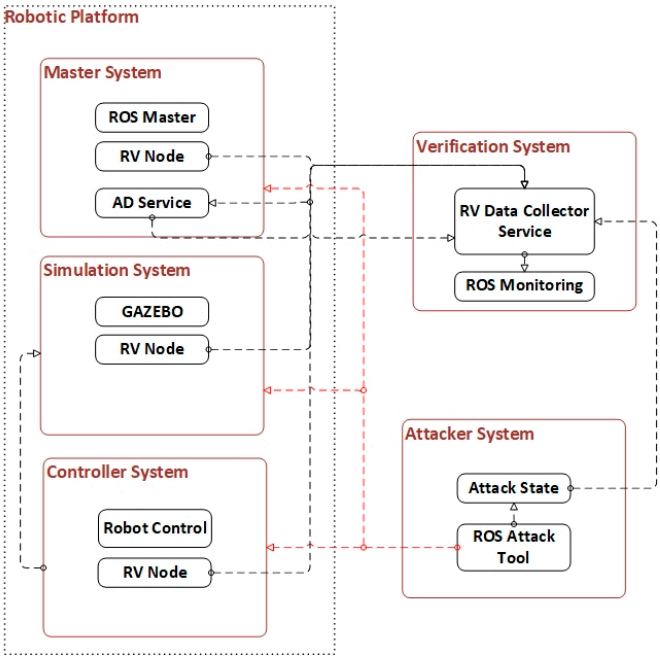


Figure 1. Runtime verification architecture.

The robotic platform is designed without loss of generality, and it consists of three subsystems: the master system (ROS Master), simulation system, and controller system. The ROS Master is the most important element required for the operation of the ROS. Every communication management in the ROS architecture is provided through the ROS Master. The simulation system simulates a robotic environment using the physics engine. A real robot could also be used instead of the simulation system. A real robot can also execute the same instructions given by the controller system. However, it may get out of control and crash during cyber-attacks. Therefore, initially it will be safer to perform attacks using

simulation. The controller system is simply the device that generates the commands for the robot using the processed sensor data.

The attacker system consists of an ROS attack tool. It could perform various attacks on the robotic platform. Some of them are as follows: several volumetric attacks, man-in-the-middle attacks, unauthorized publishing, unauthorized data access, active target searches, disabling service nodes, etc. In addition, the attacker system can publish its attack state for operations such as generating a dataset or verifying the system's security by ground truth.

The verification system is designed for the RV of the anomaly detection method. It consists of ROSMonitoring and an RV data collector node. ROSMonitoring verifies the method by using the specifications defined by Oracle. It has two types of Oracle: Register Management Logic (RML) Oracle and TL Oracle. In the present study, TL Oracle is used to specify rules and for checking the processes. Since ROSMonitoring is able to verify only one data source at once, an RV data collector is used. The RV data collector gathers data from multiple sources in synchronization. In the present study, the RV data collector subscribes two sources: the anomaly detection node and attack state node.

3.2. Runtime Verification Method

The proposed RV architecture can be used to implement various RV methods. In the present study, an attack-status-based RV is implemented for anomaly detection in robotic systems. Figure 2 shows the sequence diagram of communication for the proposed RV system architecture components. In this figure, the attacker system is represented by the attacker. The verification system is represented by ROSMonitoring and the robotic platform is represented by four subsystems, i.e., GAZEBO, ROS Master, robot control, and anomaly detection method. In Figure 2, communication of the robotic platform devices (GAZEBO and robot controller) starts with registering ROS Master. GAZEBO is used for the simulation environment. After ROS nodes are registered, the topics are shared between GAZEBO and the robot controller. Then the robotic device in GAZEBO starts realizing the task from the robot controller. Meanwhile, ROSMonitoring and anomaly detection start to check or verify the network from the beginning of communication. ROSMonitoring obtains information about both the attacker system and the robotic platform. The ROSMonitoring node verifies the anomaly detection method by obtaining anomaly detection results and attack state information. Meanwhile, the attacker starts a DoS attack against the robotic platform devices. It is assumed that there is no delay in the attack's effect on the system.

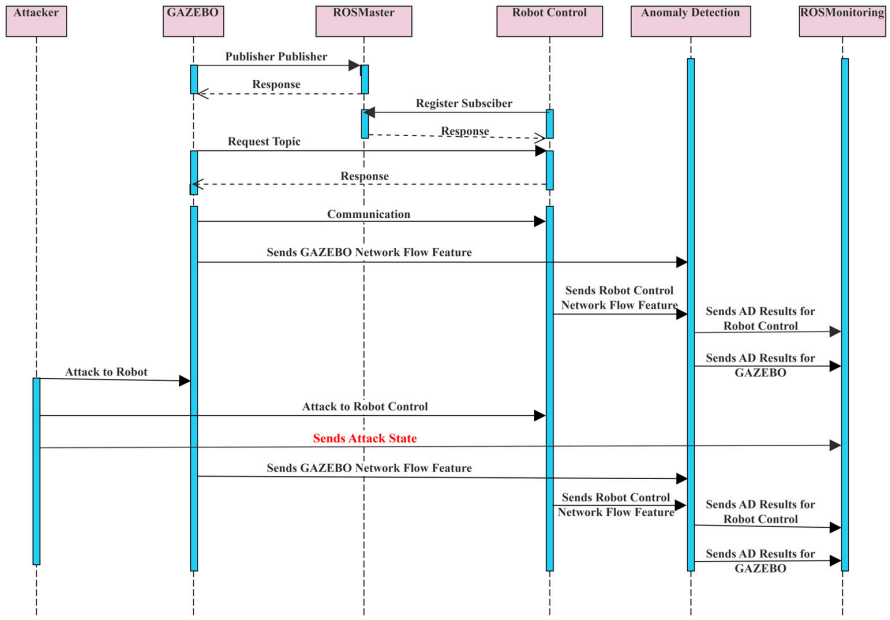


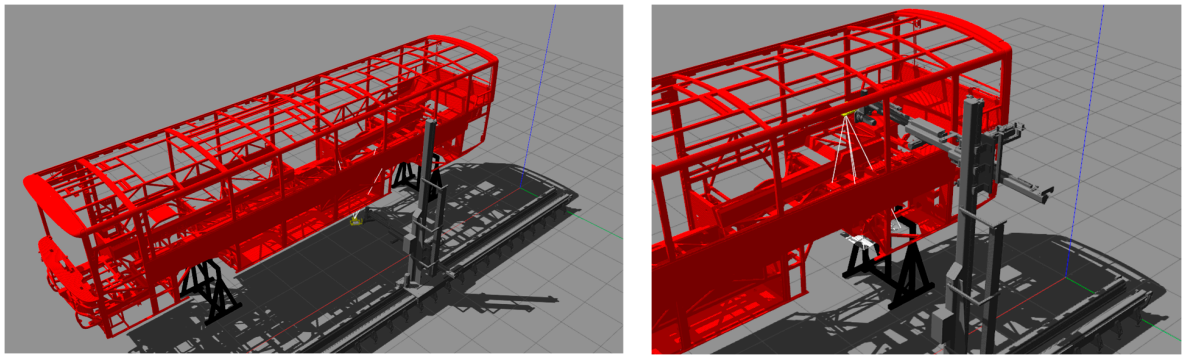
Figure 2. A sequence diagram of runtime verification architecture communication flow.

In our study, the anomaly detection method detects abnormal behavior in network traffic. This method applies statistical approaches and sends the AD result to the verification system. Statistical calculations are applied to find the upper limit and the lower limit for defining the last normal behavior. If abnormal behavior is detected for that time interval, the limits are not updated, and the method preserves the latest normal limits in the memory. The anomaly detection method gives two types of result: (a) normal condition, (b) warning condition. Anomalies are detected in univariate time-series data from the ROS environment. Every robotic system device in the robotic platform publishes data that includes two pieces of information: timestamp and frequency (for our study, the frequency is 100 for each RV node). The verification system's RV data collector node collects all data published in the robotic platform nodes and processes these data for ROSMonitoring. The processed data become ready for ROSMonitoring. The anomaly detection method obtains a slice of data for each interval and produces an AD result at the end of each check. If the anomaly detection method gives a warning about the system and attack state information does not, ROSMonitoring reports the anomaly detection method verification result as not verified. Further, if the AD method gives the normal status about the system and attack-state information is not supported, the anomaly detection method is again reported as not verified. If the anomaly detection method gives a warning message and ROSMonitoring verifies this, the verification result is reported as verified. The anomaly detection method is verified true-positively or false-positively by detecting attacks. The anomaly detection method keeps running unless the system is closed.

The proposed RV architecture integrates robotic platform, attacker system and verification system. There are some studies that focus on the various verification problem for ROS [5,20,33,34]. HAROS [34] and ROSDiscover [33] are focused on good coding and verification of software bugs in ROS software development. On the other hand, the proposed architecture focuses on runtime verification of anomaly detection. ROSRV is a pioneer work that checks the safety requirements in case of unauthorized access of the sensors' data [5]. In the proposed study, anomaly detection of volumetric attacks (unauthorized publishing/subscribing attack, DoS attack, SYN flood attack) that targets to the robotic platform are evaluated. ROSMonitoring is a recently developed framework that checks the safety requirements at runtime [20]. It was originally focused on safety. In this study, the ROSMonitoring library is also used in an integrated way. The proposed holistic RV architecture includes a robotic platform, attacker system and verification system in one architecture and focus on security test cases besides the others.

4. Experimental Study

The proposed RV architecture that is given in Figure 1 is validated using an experimental setup that is given in both Figures 3 and 4. To test the RV of anomaly detection, various volumetric attacks are also realized in this setup. An overall discussion is also included concerning the contribution of the proposed approach to robotic security.



(**a**) (**b**)

Figure 3. SRVT simulation environment. (**a**) A general perspective. (**b**) A close scene for robotic arm.

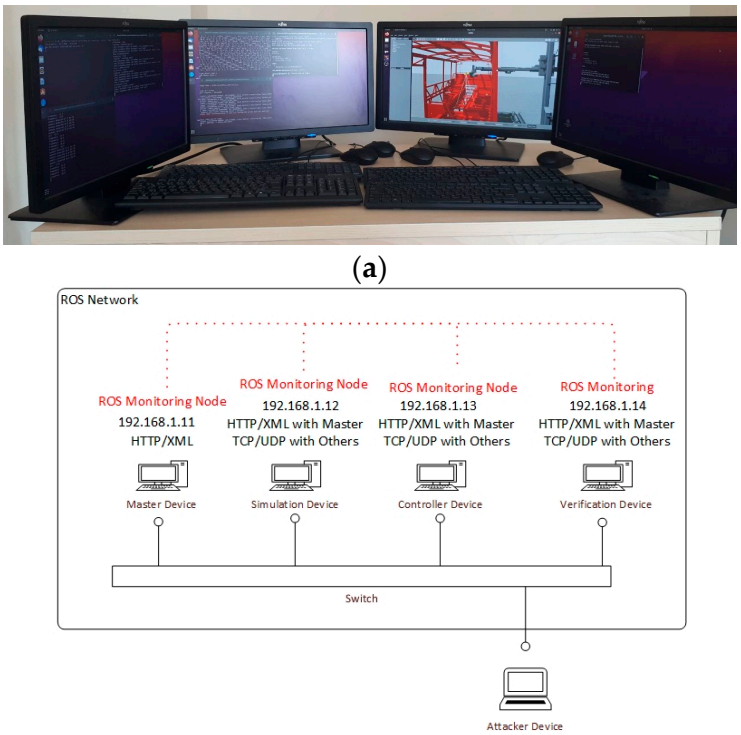


Figure 4. The experiment environment: (**a**) laboratory environment; (**b**) test-bed devices explanations.

4.1. Experimental Setup

An experimental setup is built in a laboratory environment for the proposed RV architecture as given in Figure 4a. We use an SRVT simulation environment [35] for robotic platform simulation. The SRVT simulates the "An Automated Robot Inspection Cell for Quality Control of Automotive Body in White (ROKOS)". ROKOS (robotic quality control system) is proposed for evaluation of V&V methods [36]. In the following subsection, the test-bed specifications, environment details, and devices used in the RV architecture are given. After that, RV for anomaly detection in the experimental test results is given.

The simulation environment consists of a bus chassis and a robotic arm with a depth camera (Figure 3). In the real-world experiment, the robotic arm receives missions from the controller system and the arm moves into the specified object to evaluate the missions. According to the mission plan, the robotic arm goes around the bus. The robotic arm checks incorrect or missing parts on the chassis using a depth camera. This experimental setup can also be changed with a physical ROKOS.

The experimental setup is shown in Figure 4. The figures include both the laboratory environment and the test-bed device's details. To simulate the real system, five devices are used (Figure 4a).

These are the ROS Master device, GAZEBO device, controller device, verifying device, and attacker device. The ROS Master device takes over the management of the ROS. The GAZEBO device is a computer that only runs simulation software (SRVT). The controller device manages the robot's path plan and movements. The verifying device collects data from the other devices and checks the package loss ratio. The attacker device performs volumetric attacks on the system at programmed periods. The attacker device is included in the ROS architecture. The attacker device has various attacks and other mechanisms seasuch asrching and analyzing. It targets any ROS node in the robotic platform. Moreover, the attacker device can select multiple robotic platform nodes as a target.

The specifications of all devices used in the experimental setup are given in Table 1. All of the devices are homogeneous.

Table 1. PCs' specifications.

Device Name	Specification Details
CPU	Intel Core i7 6700 @ 3.40 GHz Skylake 14 nm
RAM	8.00 GB Dual-Channel DDR3 @ 797 MHz
Motherboard	Dell Inc. 0FTVXT
Graphics	2 GB NVIDIA GeForce GTX 1050
Storage	250 GB KINGSTON SA400S37240G SATA SSD
Ethernet	Onboard
OS	Ubuntu 20.04.1 LTS 64-bit 3.36.8
ROS Version	Noetic Ninjemys
Gazebo Version	11.0.0
Network Switch	TP-Link SF-1008D 8-Port 10/100 Mbps
Graphics	2 GB NVIDIA GeForce GTX 1050
Storage	250 GB KINGSTON SA400S37240G SATA SSD
Ethernet	Onboard
OS	Ubuntu 20.04.1 LTS 64-bit 3.36.8
ROS Version	Noetic Ninjemys

ROS has various vulnerabilities; however, the main purpose of the present study is not to test the vulnerabilities of ROS. Instead, we perform tests to verify the anomaly detection method. For this reason, attack methods that completely stop the system are not used. Conversely, as some attacks did not reach sufficient volume, the expected results could not be obtained in the second experimental study. Inclusion of elements in the system, such as applications using high network traffic and physical factors that will affect communication, could also be considered as an attack. All the attacks are carried out by the attacker device. Three attack types are carried out in the experimental tests; these are (1) multiple unauthorized publishing/subscribing (different attack volumes), (2) DoS, and (3) SYN flood. In the multiple unauthorized publishing/subscribing (different attack volumes) attack, due to the lack of an authorization and authentication feature, an attacker or a user can create many publishers or subscribers. The attacker does not even have to listen to or publish data, but ROS tries to respond to the attacker's request. That creates massive data traffic, packet loss, and delay. In the second attack, the DoS attack, the aim is to shut down a machine or network, making it inaccessible to its intended users. The DoS attack accomplishes this by flooding the target with traffic or sending it information that triggers a crash. In the SYN flood attack, the attacker sends a request to connect to a server, but never completes the handshake and this attack occupies all open ports, and none are available for legitimate users to connect to.

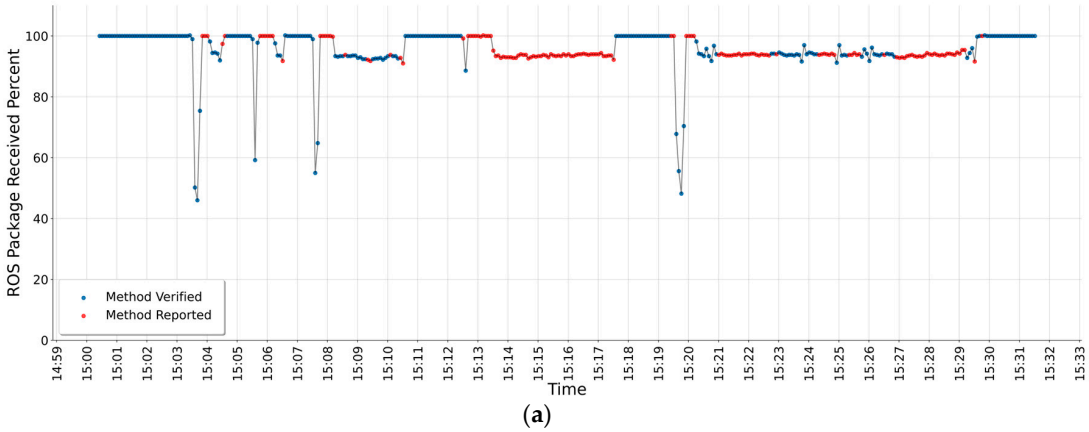
4.2. Experimental Results

We present the test results of the three different attack types: (1) multiple unauthorized publishing/subscribing attack, (2) DoS attack, (3) SYN flood attack. Therefore, three tests are evaluated in the test results. Each test follows the same timeline as in Table 2 which third and fourth columns show the duration of the attacks and the start time schedule respectively. During one test, the same attack type is used through the experiments. The duration of the attacks is changed as in Table 2. Each test initially starts with normal traffic. Then the attacks with different durations are tested. Some time gaps are also applied to eliminate the effect of previous attacks. Each test takes about 31 min. The verification system gives an alert at runtime and saves these results as the verification result (verified or non-verified method). ROSMonitoring logs the verification result approximately every five seconds. In these experiments ROSMonitoring checks the anomaly detection method by attack state information and anomaly detection results for frequency. In these experiments, the RV method and the anomaly detection method run simultaneously.

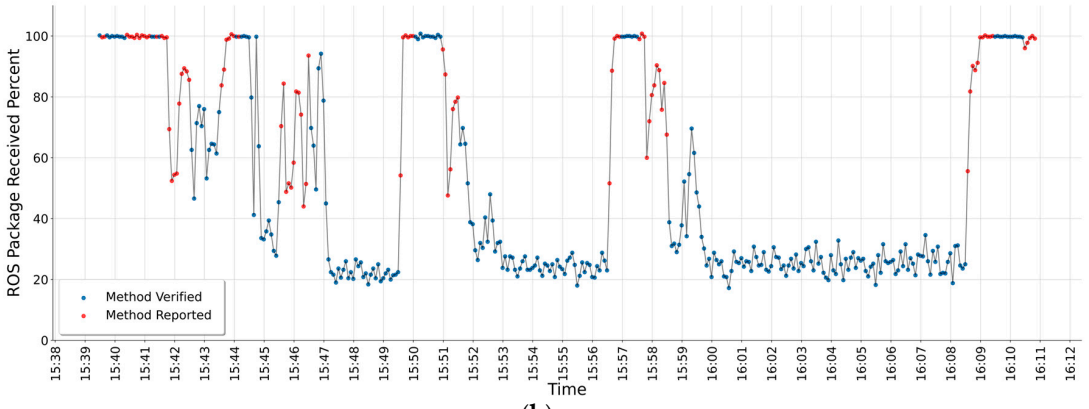
Table 2. Experimental attack schedule for the tests.

ID	Attack or Normal Traffic	Duration	Timeline of the Experimental Study
1	normal	3 min.	0 min.
2	attack	1 min.	3 min.
3	normal	1 min.	4 min.
4	attack	1 min.	5 min.
5	normal	1 min.	6 min.
6	attack	3 min.	7 min.
7	normal	2 min.	10 min.
8	attack	5 min.	12 min.

The experimental results are given in Figures 5 and 6. In all experimental results, red dots illustrate that the interval corresponding to the anomaly detection method is not verified, i.e., the method could not detect attacks. Blue dots indicate that the anomaly detection method correctly detected attacks. The x and y axes have the same meaning in each test result graph. The y-axis corresponds to the ROS package received percentage and the x-axis corresponds to real time in hours and minutes such as 3.00 pm or 3.32 pm. All experiments are evaluated in the same environment. After each experiment, the system reboots completely so that one experimental result does not affect the other one. Further, in our experiments, we left gaps between the same attack types (see in Table 2).



(**a**)



(**b**)

Figure 5. Test results of runtime verification method where ROS running in different volume of traffic: (**a**) 500 nodes traffic; (**b**) 1000 nodes traffic.

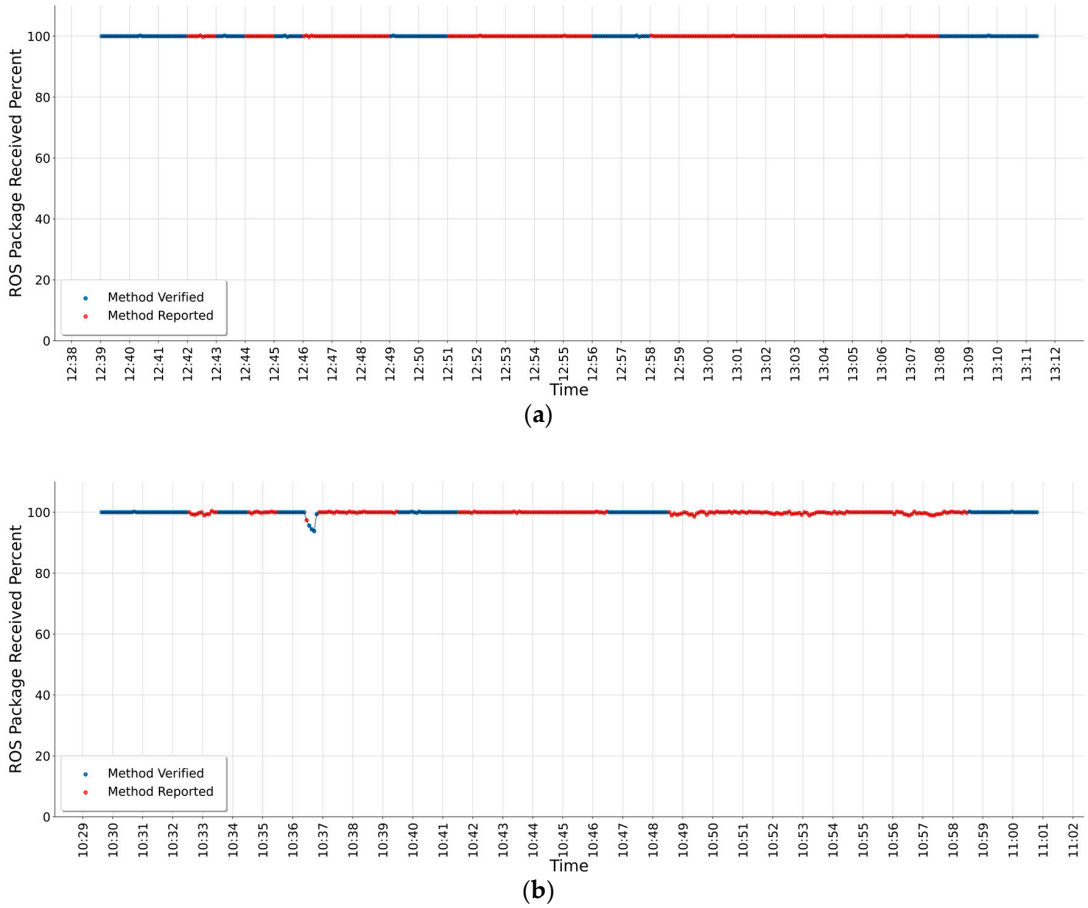


Figure 6. Test results of runtime verification method for different attack types: (**a**) DoS attack; (**b**) SYN Flood attack.

The runtime verification result for the anomaly detection method with the multiple volume of unauthorized publishing/subscribing attack is given in Figure 5. It shows the test results of the RV method for two different volumes of network traffic. The attack schedule given in Table 2 is the same for both. The RV results of scenarios': (1) 500 nodes' communication in the system, (2) 1000 nodes' communication in the system are given in Figure 5a,b, respectively. Figure 5a has five frequency decreases that are detected as anomaly and also verified with system. In Figure 5a, the first one-minute attack can be seen between 15.03 and 15.04 min. In Figure 5a, the results with red dot show that the anomaly detection algorithm is not sufficient in the stable attack moments especially seen around 15.15 min. The anomaly detection method has adapted to the changing situations and started to accept this new situation as normal. On the other hand, the anomaly detection method detect the anomaly due to increased volume of network traffic (1000 nodes). Therefore, in Figure 5b, the runtime verification results more blue dots which means the method is verified.

Figure 6 shows the RV results of the anomaly detection method for two different attacks. These are as follows: the DoS attack results in Figure 6a and the SYN flood attack results in Figure 6b. The graph results show that both attack types do not cause packets received percentage decreases. In Figure 6a,b, attack moments can be seen clearly as red dots in both figures. However, as ROSMonitoring verification receives the attack status from the attacker device and the anomaly detection method only checks the packets received percentage from the network, the methods are not verified in all attack moments. In Figure 6a, the DoS attacks test is started at 00.39 pm and lasts until around 1.12 pm. In Figure 6a, the attacks mentioned in Table 2 are evaluated as seen; the first attack is between

00.42 pm and 12.43 pm, the second attack between 00.44 pm and 00.45 pm, the third attack between 00.51 pm and 00.56 pm, and the last attack between 00.58 pm and 1.08 pm.

In Figure 6b, the SYN flood attacks test is started around 10.30 am and lasts until around 11.01 am. In Figure 6b, the attacks mentioned in Table 2 are evaluated as seen; the first attack is around 10.33 am, the second attack around 10.35 am, the third attack around 10.38 am, and the last attack around 10.55 am.

The RV approach clearly shows that the anomaly detection method has some unresolved issues. The test results show that the anomaly detection method is not sufficient when the packets' received percentage is not affected directly. As can be seen in Figure 6a,b, the anomaly detection method needs an improvement such as not only checking the system at one point; it also needs to check the system at multiple points for monitoring the whole system.

Overall, the experimental results show that RV architecture is effective in runtime verification of anomaly detection under different security attacks. This architecture has a modular structure to integrate new attack types, anomaly detection methods and various verification methods. For example, in the pioneer work [10], the effect of DoS attacks is analyzed on both ROS and network traffic. Additionally, formal verification for security attacks is presented at a conference [19]. In the literature, there is no benchmark dataset for ROS security. This RV architecture can be used to create various benchmark datasets for ROS middleware.

5. Discussion about Positioning the Outcomes with Robotic Security

No matter how the robotic system is deployed, there is a strong need to build systems based on formal threat models. In recent years, security, privacy, and safety threats are starting to be tackled coherently as systems are evolving from cyber-only or physical-only to cyber-physical. The current state-of-the-art threat models can be classified as attacker-centric, asset-centric, vulnerability- and threat-centric, and software-centric [37]. The applied RV method can contribute to the development of countermeasures against various security, privacy, and safety attacks. Attacker-centric, asset-centric, vulnerability- and threat-centric, and even software-centric threat models can be applied for identifying the potential of the proposed method for better resilience against cyber-physical risks.

It is noteworthy that the proposed RV method improves the resilience of robotic systems [38]. In the present paper, we focus on the following robotic security requirements: (i) system security, privacy threat modeling, (ii) intrusion detection, and (iii) scalability, user, and system constraints.

For the first security requirement, an architecture is needed to analyze the methods and means of attacks and to protect any critical system data or preserve privacy. In the present study, an architecture is proposed for RV of robotic systems' security. This architecture consists of an attacker device, controller device, and robotic device. Unauthorized nodes connect to the system as an attacker and ROSMonitoring verifies whether the system is correctly catching attacks or not. With the help of this study, the methods that can be used are analyzed and necessary warnings are given in this regard. The second security requirement, intrusion detection, is also another important scope of the proposed RV approach. Intrusion detection systems could use various methods for warning the system, such as signature-based, anomaly detection-based, host-based, and network-based. Anomaly detection methods can be considered a subfield of intrusion detection systems.

In the present study, our architecture shows that the intrusion detection method's RV improves the reliability of the system. The study also contributes to the third requirement, since the robotic system can serve multiple users without any significant degradation in performance or any unmanageable overhead. Our study uses the robotic platform without losing generality. Each part of the architecture has been established as extensible in this platform. For example, new robotic devices can be added to the robotic platform. Verification devices can also monitor a newly added device and the attacker system can also attack that device. In the present study, online performance monitoring and frequency

checks aligned with RV are performed. In this architecture, the robotic platform could be scalable by enlarging or shrinking.

Advantages: The runtime verification is selected due to the dynamic nature of anomaly detection since it is not facing a state-explosion problem. In RV systems, there is no need to record trace data, and the results of RV can be seen in runtime. The proposed architecture ensures the integration of RV processes into robotic system's security. Therefore, it reduced both the human effort and the time required to complete the V&V process by automating the attacks synchronized with RV system. On the other hand, the RV system and attacker platform are located separately from the robotic platform. Therefore, each part of the architecture is open to development individually.

Disadvantages: These experimental results showed that the anomaly detection method is not sufficient in some attacks that especially do not directly affect the network flow. Although it is a disadvantage in terms of anomaly detection, it validates the RV architecture. The improvements in anomaly detection methods can be easily tested in RV architecture. Additionally, both configurations of the anomaly detection and the RV take time and need expertise in this area.

6. Conclusions

A runtime verification architecture is proposed and validated for anomaly detection of robotic system's security. The runtime verification architecture has three major entities: a verification device, attacker device, and robotic platform. The verification system includes ROSMonitoring for runtime verification and a data collector for collecting data from the robotic platform. The robotic platform is set up to include the basic ROS and anomaly detection system experiments are carried out in a laboratory environment. In the experimental study, three types of attacks such as multiple unauthorized publishing/subscribing attack, DoS attack, and SYN flood attack are tested. The runtime verification architecture verifies the anomaly detection method's using attack state information. The test results showed that the runtime verification architecture could be used for verification of anomaly detection methods. Therefore, a runtime verification method for anomaly detection methods could be used in such attacks. The proposed runtime verification architecture reduces both the human effort and the time required to complete the V&V process.

In future studies, the RV architecture could be tested with different attacks in the security attack tool. Moreover, machine learning methods can be used for anomaly detection and these methods could be studied in RV architecture. Future attempts with different attack types that have cascading effects on the system can be considered for RV.

Author Contributions: Conceptualization, E.D., Y.S.K., Z.D. and A.Y.; methodology, E.D., Y.S.K., Z.D. and A.Y.; software, E.D., Y.S.K. and Z.D.; validation, A.Y. and M.O.; formal analysis, A.Y.; investigation, E.D., Y.S.K. and Z.D.; resources, A.Y.; data curation, E.D. and Y.S.K.; writing—original draft preparation, E.D., Y.S.K., Z.D., A.Y., M.O., S.E. and A.K.; writing—review and editing, E.D., Y.S.K. and A.Y.; visualization, E.D. and Y.S.K.; supervision, A.Y.; project administration, A.Y. and S.E.; funding acquisition, A.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by theVALU3S project that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey (TUBITAK, under contract no: 119N356). The views expressed in this work are the sole responsibility of the authors and do not necessarily reflect the views or position of the European Commission. The authors, the VALU3S Consortium, and the ECSEL JU are not responsible for the use which might be made of the information contained in here. This work is supported by the Scientific and Technical Research Council of Turkey (TUBITAK), Contract No 120N800, project title: "Verification and Validation of Automated Systems' Safety and Security".

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to ongoing project restrictions.

Acknowledgments: This work was supported by theVALU3S project that has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey (TUBITAK, under contract no: 119N356).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Siriweera, A.; Naruse, K. Survey on Cloud Robotics Architecture and Model-Driven Reference Architecture for Decentralized Multicloud Heterogeneous-Robotics Platform. *IEEE Access* **2021**, *9*, 40521–40539. [CrossRef]
2. Plósz, S.; Schmittner, C.; Varga, P. *Combining Safety and Security Analysis for Industrial Collaborative Automation Systems*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 187–198.
3. White, R.; Christensen, D.; Henrik, I.; Quigley, D. SROS: Securing ROS over the Wire, in the Graph, and through the Kernel. *arXiv* **2016**, arXiv:161107060.
4. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the Performance of ROS2. In Proceedings of the 13th International Conference on Embedded Software, Pittsburgh, PA, USA; 2016; pp. 1–10.
5. Huang, J.; Erdogan, C.; Zhang, Y.; Moore, B.; Luo, Q.; Sundaresan, A.; Rosu, G. *ROSRV: Runtime Verification for Robots*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 247–254.
6. Balsa-Comerón, J.; Guerrero-Higueras, Á.M.; Rodríguez-Lera, F.J.; Fernández-Llamas, C.; Matellán-Olivera, V. *Cybersecurity in Autonomous Systems: Hardening ROS Using Encrypted Communications and Semantic Rulesa*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 67–78.
7. Fernández Muro, B. Securing Communications in Surgery Robots. Phd thesis, Politecnico di Torino, Torino, Italy, 2018.
8. Staffa, M.; Mazzeo, G.; Sgaglione, L. *Hardening ROS via Hardware-Assisted Trusted Execution Environment*; IEEE: Piscataway, NJ, USA, 2018; pp. 491–494.
9. Rivera, S.; Lagraa, S.; State, R. ROSploit: Cybersecurity Tool for ROS. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 415–416.
10. Degirmenci, E.; Kirca, Y.S.; Yolacan, E.N.; Yazici, A. An Analysis of DoS Attack on Robot Operating System. *GAZI Univ. J. Sci.* **2024**, *1*, 1.
11. Tseng, T.W.; Wu, C.T.; Lai, F. Threat Analysis for Wearable Health Devices and Environment Monitoring Internet of Things Integration System. *IEEE Access* **2019**, *7*, 144983–144994. [CrossRef]
12. Narayanan, V.; Bobba, R.B. Learning Based Anomaly Detection for Industrial Arm Applications. In Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, Toronto, ON, Canada, 15–19 October 2018; pp. 13–23.
13. Afzal, A.; Le Goues, C.; Timperley, C.S. Mithra: Anomaly Detection as an Oracle for Cyberphysical Systems. *IEEE Trans. Softw. Eng.* **2022**, *48*, 4535–4552. [CrossRef]
14. Chatterjee, A.; Ahmed, B.S. IoT Anomaly Detection Methods and Applications: A Survey. *Internet Things* **2022**, *19*, 100568. [CrossRef]
15. Bartocci, E.; Falcone, Y. *Lectures on Runtime Verification*; Springer: Berlin/Heidelberg, Germany, 2018; ISBN 3-319-75632-X.
16. Leucker, M.; Schallhart, C. A Brief Account of Runtime Verification. *J. Log. Algebr. Program.* **2009**, *78*, 293–303. [CrossRef]
17. Falcone, Y.; Havelund, K.; Reger, G. A Tutorial on Runtime Verification. *Eng. Dependable Softw. Syst.* **2013**, 141–175.
18. Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; Roli, F. *Evasion Attacks against Machine Learning at Test Time*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 387–402.
19. Kïrca, Y.S.; Değirmenci, E.; Yazïcï, A.; Özkan, M. ROS Based Attack Tool for Verification of Robotic System Security. In Proceedings of the 23rd Turkish Automatic Control National Conference (TOK 2022), Elazig, Turkey, 15–18 September 2022; pp. 557–561.
20. Ferrando, A.; Cardoso, R.C.; Fisher, M.; Ancona, D.; Franceschini, L.; Mascardi, V. *ROSMonitoring: A Runtime Verification Framework for ROS*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 387–399.
21. Sinha, R.; Patil, S.; Gomes, L.; Vyatkin, V. A Survey of Static Formal Methods for Building Dependable Industrial Automation Systems. *IEEE Trans. Ind. Inform.* **2019**, *15*, 3772–3783. [CrossRef]
22. Gjondrekaj, E.; Loreti, M.; Pugliese, R.; Tiezzi, F.; Pinciroli, C.; Brambilla, M.; Birattari, M.; Dorigo, M. *Towards a Formal Verification Methodology for Collective Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 54–70.
23. Halder, R.; Proença, J.; Macedo, N.; Santos, A. *Formal Verification of ROS-Based Robotic Applications Using Timed-Automata*; IEEE: Piscataway, NJ, USA, 2017; pp. 44–50.
24. Webster, M.; Western, D.; Araiza-Illan, D.; Dixon, C.; Eder, K.; Fisher, M.; Pipe, A.G. A Corroborative Approach to Verification and Validation of Human–Robot Teams. *Int. J. Robot. Res.* **2020**, *39*, 73–99. [CrossRef]
25. Luckcuck, M.; Farrell, M.; Dennis, L.A.; Dixon, C.; Fisher, M. Formal Specification and Verification of Autonomous Robotic Systems: A Survey. *ACM Comput. Surv. CSUR* **2019**, *52*, 1–41. [CrossRef]
26. Hu, C.; Dong, W.; Yang, Y.; Shi, H.; Zhou, G. Runtime Verification on Hierarchical Properties of ROS-Based Robot Swarms. *IEEE Trans. Reliab.* **2019**, *69*, 674–689. [CrossRef]

27. Watanabe, K.; Kang, E.; Lin, C.-W.; Shiraishi, S. Runtime Monitoring for Safety of Intelligent Vehicles. In Proceedings of the 55th Annual Design Automation Conference, San Francisco, CA, USA, 2018; pp. 1–6.
28. Luo, C.; Wang, R.; Jiang, Y.; Yang, K.; Guan, Y.; Li, X.; Shi, Z. *Runtime Verification of Robots Collision Avoidance Case Study*; IEEE: Piscataway, NJ, USA, 2018; Volume 1, pp. 204–212.
29. An, D.; Liu, J.; Zhang, M.; Chen, X.; Chen, M.; Sun, H. Uncertainty Modeling and Runtime Verification for Autonomous Vehicles Driving Control: A Machine Learning-Based Approach. *J. Syst. Softw.* **2020**, *167*, 110617. [CrossRef]
30. Zeller, S. Secure Self-Reconfiguring Services to Mitigate DoS Attacks. Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2019.
31. Torjusen, A.B.; Abie, H.; Paintsil, E.; Trcek, D.; Skomedal, Å. Towards Run-Time Verification of Adaptive Security for IoT in EHealth. In Proceedings of the 2014 European Conference on Software Architecture Workshops, Vienna, Austria, 25–29 August 2014; pp. 1–8.
32. Pradhan, M.; Noll, J. Security, Privacy, and Dependability Evaluation in Verification and Validation Life Cycles for Military IoT Systems. *IEEE Commun. Mag.* **2020**, *58*, 14–20. [CrossRef]
33. Timperley, C.S.; Dürschmid, T.; Schmerl, B.; Garlan, D.; Le Goues, C. ROSDiscover: Statically Detecting Run-Time Architecture Misconfigurations in Robotics Systems. In Proceedings of the 2022 IEEE 19th International Conference on Software Architecture (ICSA), Honolulu, HI, USA, 12 March 2022; pp. 112–123.
34. Dal Zilio, S.; Hladik, P.-E.; Ingrand, F.; Mallet, A. A Formal Toolchain for Offline and Run-Time Verification of Robotic Systems. *Robot. Auton. Syst.* **2023**, *159*, 104301. [CrossRef]
35. Yayan, U.; Erdogmus, A. Endüstriyel Robot Hareket Planlama Algoritmaları Performans Karşılaştırması. *J. Sci. Technol. Eng. Res.* **2021**, *2*, 31–45.
36. Kanak, A.; Ergun, S.; Ozkan, M.; Çokünlü, G.; Yayan, U.; Karaca, M.; Arslan, A.T. Verification and Validation of an Automated Robot Inspection Cell for Automotive Body-in-White: A Use Case for the VALU3S ECSEL Project. *Open Res. Eur.* **2021**, *1*, 115. [CrossRef]
37. Hamad, M.; Prevelakis, V. SAVTA: A Hybrid Vehicular Threat Model: Overview and Case Study. *Information* **2020**, *11*, 273. [CrossRef]
38. Fosch-Villaronga, E.; Mahler, T. Cybersecurity, Safety and Robots: Strengthening the Link between Cybersecurity and Safety in the Context of Care Robots. *Comput. Law Secur. Rev.* **2021**, *41*, 105528. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.