electronics

MDPI

Article

Formal Analysis and Detection for ROS2 Communication Security Vulnerability

Shuo Yang [1], Jian Guo [2,3,*] and Xue Rui [3]

[1] MoE Engineering Research Center for Software/Hardware Co-Design Technology and Application, East China Normal University, Shanghai 200062, China; 51215902140@stu.ecnu.edu.cn
[2] National Trusted Embedded Software Engineering Technology Research Center, East China Normal University, Shanghai 200062, China
[3] School of Information Science and Technology, Xinjiang Teacher's College, Urumgi 830043, China
* Correspondence: jguo@sei.ecnu.edu.cn

Abstract: Robotic systems have been widely used in various industries, so the security of communication between robots and their components has become an issue that needs to be focused on. As a framework for developing robotic systems, the security of ROS2 (Robot Operating System 2) can directly affect the security of the upper-level robotic systems. Therefore, it is a worthwhile research topic to detect and analyze the security of ROS2. In this study, we adopted a formal approach to analyze the security of the communication mechanism of ROS2. First, we used a state transition system to model the potential vulnerabilities of ROS2 based on the ROS2 communication mechanism and the basic process of penetration testing. Secondly, we introduced a CIA model based on the established vulnerability model and used linear temporal logic to define its security properties. Then, we designed and implemented a vulnerability detection tool for ROS2 applications based on the vulnerability model and security properties. Finally, we experimentally tested some ROS2-based applications, and the results show that ROS2 has vulnerabilities without additional protection safeguards.

Keywords: robotic system; ROS2; communication mechanisms; security and safety analysis; formal method

Citation: Yang, S.; Guo, J.; Rui, X. Formal Analysis and Detection for ROS2 Communication Security Vulnerability. *Electronics* **2024**, *13*, 1762. https://doi.org/10.3390/electronics13091762

Academic Editors: Chih-Chieh Chang, Nai-Wei Lo and Jheng-Jia Huang

Received: 30 March 2024
Revised: 29 April 2024
Accepted: 30 April 2024
Published: 2 May 2024

1. Introduction

Robotic systems, as a representative of advanced automation technology, have been widely used in various industries, including but not limited to home service [1–3], healthcare [4–6], public safety [7,8], and many other fields. The Top 5 Robot Trends 2021 [9], published by the International Federation of Robotics, shows that the annual installations of industrial robots more than tripled between 2010 and 2019, and the global sales of professional service robots increased by 32% to $11.2 billion in 2018–2019. These figures fully demonstrate the huge potential of the robotics market. As robotic systems become more and more intelligent and autonomous, the interaction and communication between system nodes and between the system and the external world are rapidly increasing. How to test, analyze, and enhance the security of robotic systems have become critical issues as there may be a large amount of confidential or private data in the communication network, and any failure in this network may also lead to data leakage or system failure [10–12].

In the Robot Operating System (ROS) [13], a widely used framework for robotics application development, communication among various components, known as nodes, is facilitated through a publish–subscribe messaging system. Nodes can publish messages to specific topics, and other nodes can subscribe to those topics to receive the messages. This communication mechanism is powerful for building complex robotic systems but initially lacked robust security features, leaving it susceptible to various vulnerabilities and attacks [14].

Similar to ROS, ROS2 does not have a unified standard for security measurement, so we analyzed its security using the CIA model, which is common in information security. In addition, we formally modeled the communication security vulnerabilities of ROS2, designed and developed a vulnerability scanning tool for ROS2 based on the model we saw, and finally tested and analyzed the security of specific ROS2 applications with this tool.

The main work of this article is as follows:

- For the different communication mechanisms of ROS2, we formally modeled and analyzed the potential vulnerabilities in it and, at the same time, formally expressed the CIA properties as the security properties;
- Based on the established vulnerability model and security properties, we designed and developed an ROS2 vulnerability detection tool. The tool detects vulnerabilities in the ROS2 system by means of a reachability analysis and analyzes which properties in the ROS2 CIA are damaged by the detected vulnerabilities.

The rest of the article is organized as follows. In Section 2, we list some of the related work. In Section 3, we present the possible communication security vulnerabilities in the different communication mechanisms of ROS2. In Section 4, we formally model and analyze the proposed communication security vulnerability model and the CIA properties that ROS2 needs to satisfy. In Section 5, we present the design and implementation of ROS2Tester, a ROS2 vulnerability detection tool. In Section 6, we discuss the implementation environment that we built for ROS2 security testing and tested using ROS2Tester. In Section 7, the work of this article is summarized, and future research directions are presented.

2. Related Work

The Robot Operating System (ROS), as a widely used framework for robot application development, was not designed initially with adequate consideration of its security, making ROS-based robot systems also vulnerable to security attacks, such as information theft, tampering, and denial-of-service attacks [14]. These security issues pose potential threats and risks to both the robot itself and the environment in which the robot is applied, so it becomes critical to protect the security of the ROS.

At the early stage of ROS development, most of researchers' studies on its communication security were focused on a specific system rather than on the ROS itself, and the objects of research were rather fragmented and specific, such as rescue robots [15,16], household robots [17,18], telemedicine robots [19,20], and drones [21]. Risk analyses also focused on remote communication and control security, such as efficient remote authentication [22], telemedicine protocols [23], etc. These studies reflect the lack of a unified underlying framework for robots, and the research on robot security has not focused on the underlying architecture of ROS/ROS2.

Of course, in addition to the security of the ROS itself, the security of the "computer" running the robotic system is equally important. In the official community, researchers have isolated a number of possible attackers from a network and authentication perspective [24]. Benjamin Breiling secured an ROS on a peer-to-peer basis through the direct interaction between publishers and subscribers, thus strengthening the authenticity and integrity of the communication at the application layer [25].

With the gradual maturity of ROS development, more and more researchers have begun to focus on the impact of the communication security of the ROS framework itself on the security of the entire robot system, and a large number of research results on ROS security analysis and security solutions have emerged. In order to achieve an in-depth analysis of ROS communication security, researchers have explored and summarized it from many different directions. Among them, researchers have tested the communication security vulnerabilities of ROSs from the perspective of simulated attacks by means of penetration testing [26], of which the typical testing tools are ROSPenTo [27] and ROSploit [28]. ROSPenTo is the first tool to achieve a vulnerability attack on the communication of ROS nodes, parameters, topics, etc., and ROSploit adds network scanning and other

expansion functions on the basis of the former to make it more suitable for practical scenarios. Based on the ROSPenTo tool, researchers have successfully verified that ROSs have security vulnerabilities such as to unauthorized publication, unauthorized data access, and denial-of-service attacks [29], and they have proposed a security protection solution in the direction of the application layer for solving such problems. In addition, some researchers have also conducted research using formal methods for verifying the security and reliability of the ROS communication [30] using runtime verification to enable the state monitoring of a running robotic system [31].

Along with the security analysis of ROSs, in order to enhance the security of a ROS, a number of researchers have proposed a series of schemes, most of which use encryption and authentication mechanisms. However, these solutions cannot fully protect against sophisticated security attacks, so other researchers have explored many other directions for ROS security. ROS_Immunity [32] integrates internal system defenses, external system authentication, and automated vulnerability detection with Secure–ROS [33] together to provide a set of defenses for ROSs against malicious attackers.

In 2014, the official ROS community introduced a robotic operating system, ROS2, in order to improve the system's real-time performance and security. Compared to the ROS, the ROS2 middleware framework is based on the DDS (Data Distribution Service) protocol, which provides better performance and security, and therefore, the latest robot manufacturers prefer ROS2-based applications. Although ROS2 uses DDS Security and SROS2 [34] to enhance its security and usability, it is not absolutely secure. The official ROS community has posted vulnerabilities in the robotic system from hardware to software [35], and the vulnerabilities regarding the ROS2 level are highly relevant to the work of this article. On the one hand, the security problems existing in ROS may not be fully solved by ROS2, and on the other hand, ROS2 itself may also have new security problems. Some early studies [36,37] have analyzed the security and performance of ROS2 and SROS2 in a comprehensive way, trying to find out the balance between the two. At the same time, some other articles [36,38] have shown that SROS2 still has flaws at present through conducting a security analysis of DDS Security and SROS2.

3. ROS2 Communication Security Vulnerability

In this section, we present potential communication security vulnerabilities based on the ROS2 communication mechanism. And we explain the principle and attack flow of these vulnerabilities.

3.1. Security Vulnerability of Topic Communication

In topic communication, nodes can act as publishers or subscribers. Publisher nodes are responsible for publishing specific types of messages to topics, while subscriber nodes receive messages by subscribing to the same topics. The communication between publishers and subscribers of a topic is asynchronous. Publishers can publish messages to multiple subscribers, and subscribers can receive messages from multiple publishers at the same time. During the communication process, the publisher nodes serialize the message data into binary format and send it to the topic through the communication layer of ROS2. Subscriber nodes obtain the information sent by publishers by receiving and deserializing the message data.

(1) Stealing basic data of the topic
In addition to the sensitive data contained within the message, the topic itself has information with some data used for publication or subscription, such as topic name, topic type, and so on. According to the DDS discovery protocol, any other node in the same communication domain can access the data information of the topic without additional protection for ROS2 communication. Therefore, as long as the intruder node is able to join the communication domain where the target is located, it is able to steal the information of the target topic. Then, the intruder can steal the data like the ROS2

network structure, and may carry out further intrusion into the ROS2 communication based on the obtained data.

(2) Unauthorized subscription
In the ROS2 topic communication mechanism, any node can subscribe to any topic without authorization to obtain the message data. An intruder can use this vulnerability to steal messages from an application, resulting in the disclosure of important system data or user's privacy data. Since any node can subscribe to any topic without authorization, an intruder can create a malicious node to impersonate a subscriber after obtaining the data related to the target topic so as to obtain the data in the topic.

(3) Unauthorized publication
Similar to unauthorized subscription, nodes in ROS2 are able to publish messages to any topic without authorization, which may be used by intruders to inject false data or commands into applications, thus interfering with their normal operation and causing undesirable consequences. Before carrying out the attack, the intruder first needs to obtain the relevant parameters of the target topic, such as topic name and topic type. Then, the intruder creates a malicious node in the domain where the target topic is located and creates a publisher on that node. Finally, based on the obtained topic name and topic type, the intruder can forge false messages recognizable to the target topic, which the target topic will publish to all the nodes subscribed to the topic.

3.2. Security Vulnerability of Service Communication

Service communication is a communication mechanism based on a request–response model. In service communication, a node can provide a service or invoke a service. The service server node registers a particular service and defines the data types of the request and response. When another node invokes the service, the request is sent to the server-side node, which performs the appropriate computation or operation and sends the result back to the client node as a response.

(1) Stealing basic service data
In addition to the sensitive data contained within the ROS2 service, the service itself has some data information, such as the service name, service type, etc., based on which the client node can send a request to the specified service. According to the DDS discovery protocol, any other node in the same communication domain can access the data information of the service without the additional protection of ROS2 communication. Therefore, as long as the intruder node joins the communication domain where the target is located, it can steal the information of the target service. Then, the intruder may realize the theft of data, such as of the ROS2 network structure, and carry out further intrusion into the service communication.

(2) Unauthorized service call
In ROS2 communication, any node in the same communication domain can make calls to services in the domain and receive responses. Therefore, on the basis of stealing the basic data of a service, the intruder can also communicate with the target service based on these data and send malicious requests to the target service, thus successfully stealing the sensitive ROS2 data or issuing malicious commands to the ROS2 nodes, which can cause serious consequences.

3.3. Security Vulnerability of Action Communication

Action communication combines the characteristics of topics and services. In action communication, a node can act as a target node, a feedback node, or a result node of an action. The target node sends a goal to the action server and waits for the server to execute the relevant action. During execution, the action server sends periodic feedback to the feedback node so that the target node can understand the execution progress of the action. When the action is completed, the result node receives the final result.

(1) Stealing basic action data

In addition to the sensitive data contained within the ROS2 service, the action itself has some data information, such as the action name, action type, etc., based on which the client node can send a request to the specified service. According to the DDS discovery protocol, any other node in the same communication domain can access the information of the action without the additional protection. Therefore, the intruder node is able to join the communication domain and steal the data of the target's actions, thus realizing the stealing of basic data, and may carry out further intrusion into the action communication.

(2) Unauthorized action call
In ROS2 communication, any node in the same communication domain can make calls to actions in the domain and receive responses and feedback. Therefore, the intruder can also communicate with the target action based on these data and send malicious requests to the target action based on stealing the basic data of the action, successfully stealing sensitive ROS2 data or issuing malicious commands to the ROS2 nodes.

4. Modeling of Security Vulnerability

In this section, we formally model seven ROS2 communication security vulnerabilities using a transition system. Based on the ROS2 communication process and the attack flow of an intruder node, the model $M = \{S, \Sigma, \delta, I\}$ can be built based on the following:

- S is the set of all states in the system;
- Σ is the set of actions, representing all the actions in the system;
- $\delta = S \times \Sigma \times S$ denotes the transition relationship between the states in the system;
- I denotes the set of initial states of the system.

In addition, based on the model developed, we use linear temporal logic (LTL) [39] to represent the confidentiality, integrity, and availability of the CIA security properties. LTL formulates over the set AP of atomic propositions are formed according to the following grammar:

$$\varphi ::= true \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid X\varphi \mid \varphi_1 U \varphi_2$$

where $a \in AP$, X means "next", and U means "until". LTL's explanations are all given in an infinite trajectory as follows:

- a: a holds at the current time, and in the trajectory, it behaves as if it holds at the first position.
- $X\varphi$: φ holds at the next time point and holds at the second position in the trajectory.
- $\varphi_1 U \varphi_2$: φ_1 holds until φ_2 holds.
- $F\varphi$: $F\varphi = true\ U\ a$, which means that φ holds sometime in the future.
- $G\varphi$: $G\varphi = \neg\ F\neg\varphi$, which means that φ always holds in the future.

LTL is particularly well suited to modeling the dynamic behavior of systems, especially those properties that involve temporal order and persistence. The following are some of the types of properties that can be modeled with LTL: liveness, responsiveness, fairness, etc.

4.1. Modeling of Topic Security Vulnerability

4.1.1. Vulnerability Model

In Section 2, we introduced three vulnerabilities in the ROS2 topic communication mechanism, namely, stealing the basic data of the topic, unauthorized subscription, and unauthorized publication. And in the following, we will model each of these three vulnerabilities as M_1, M_2, and M_3.

In our work, we built the model for stealing basic data of the topic as follows:

$$M_1 = \{S_1, \Sigma_1, \delta_1, I_1\}$$

where the following are considered:

- $S_1 = \{idle, check1, auth, comm, wait\_t, check2, success, fail\}$.
There are eight states in S_1. In these states, *idle* denotes the initial state of the vulner-

ability model; *check*1 denotes the state to check about whether the intruder can be authorized; *auth* denotes that the intruder node can be an authorized node; *comm* denotes that the intruder node is already able to communicate with the target; *wait_t* indicates that the intruder node waits to receive information about the target topic; *check*2 denotes the state to check about whether the intruder can obtain the basic data of the topic; and *success* and $fail$ represent whether the topic information has been successfully obtained or not, respectively.

- $\Sigma_1 = \{authorized, get\_domain, get\_topic, return\_topic\}$.
  authorized denotes to determine whether the node is an authorized node; *get_domain* denotes to get the communication domain where the target topic is located; and *get_topic* denotes to get the information of the target topic. *authorized*, *get_domain*, *get_topic*, and *return_topic* are channels that are used to communicate with the ROS2 communication model.
- The initial state set $I_1 = \{idle\}$.
- The transition relationship δ_1 between the states is shown in Figure 1.

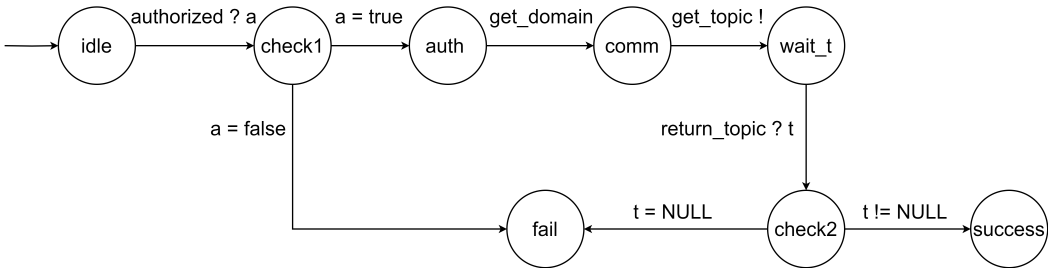

Figure 1. Stealing the basic data of the topic.

The model for the unauthorized subscription is

$$M_2 = \{S_2, \Sigma_2, \delta_2, I_2\}$$

where the following are considered:

- $S_2 = \{idle, check1, auth, comm, wait\_t, check2, sub, wait\_m, success, fail\}$.
  There are ten states in S_2. In these states, *sub* indicates that the intruder node can subscribe to the target topic; *wait_m* indicates that the intruder node waits for the topic that has been subscribed to; and *success* and $fail$ represent whether or not it successfully subscribed to and received the messages in the topic, respectively. The rest of the states have the same meaning as in S_1.
- $\Sigma_2 = \{authorized, get\_domain, get\_topic, return\_topic, subscribe, publish, timeout\}$.
  subscribe and *publish* are channels. The intruder node sends a request to the subscribe topic through channel *subscribe* and receives the topic through channel *publish*. If it cannot receive a message for a long time, it will receive a timeout response through channel *timeout*. The rest of the states have the same meaning as in Σ_1.
- The initial state set $I_2 = \{idle\}$.
- The transition relationship δ_2 between the states is shown in Figure 2.

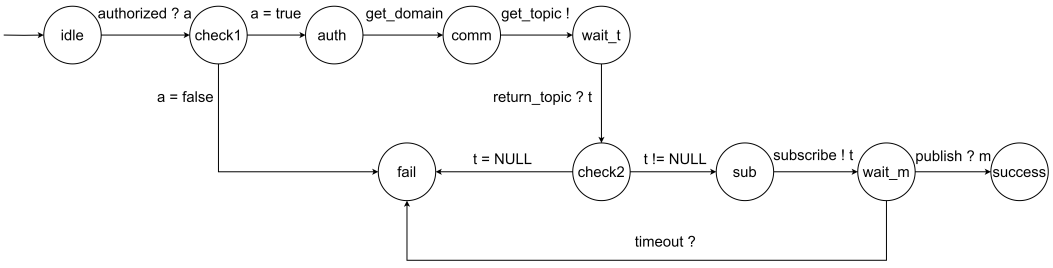

Figure 2. Unauthorized subscription.

The model for unauthorized publication is

$$M_3 = \{S_3, \Sigma_3, \delta_3, I_3\}$$

where the following are considered:

- $S_3 = \{idle, check1, auth, comm, wait\_t, check2, pub, wait\_m, success, fail\}$. There are ten states in S_3. In these states, pub indicates that the intruder node publishes a message to the target topic; $success$ and $fail$ represent whether it has successfully published a fake message to the topic or not, respectively. The meanings of the remaining states are the same as in S_1.
- $\Sigma_3 = \{authorized, get\_domain, get\_topic, return\_topic, fakemsg, publish\}$. $fakemsg$ means to fake a fake message based on the topic information obtained, and $publish$ means to publish a message to the target topic. The rest of the actions have the same meaning as Σ_2.
- The initial state set $I_3 = \{idle\}$.
- The transition relationship δ_3 between states is shown in Figure 3.

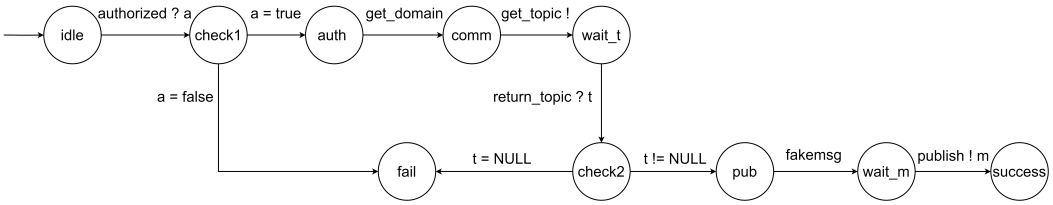

Figure 3. Unauthorized publication.

4.1.2. Security Specification

ROS2 topic confidentiality requires that only authorized nodes have access to the topic data, which can be broken down into the following two points:

- Only authorized nodes can access the information (topic name, topic type) of topics in the communication domain. The ROS2 node should satisfy

$$G(\neg((auth \rightarrow F(wait\_t)) \rightarrow F(success)))$$

- Only authorized subscriber nodes can successfully subscribe to the target topic. The ROS2 subscriber node should satisfy

$$G(\neg((auth \rightarrow F(sub)) \rightarrow F(success)))$$

Thus, ROS2 topic confidentiality can be expressed as

$$\begin{aligned} &G(\neg((auth \rightarrow F(wait\_t)) \rightarrow F(success)) \wedge \\ &\neg((auth \rightarrow F(sub)) \rightarrow F(success))) \end{aligned} \tag{1}$$

ROS2's topic integrity requires that only authorized nodes can modify the topic's data, and only authorized publisher nodes can post messages to the topic, which means there is no unauthorized publisher node that can post messages to the topic. Thus, ROS2 topic integrity can be expressed as

$$G(\neg((auth \rightarrow F(pub)) \rightarrow F(success))) \tag{2}$$

Topic availability in ROS2 requires that authorized subscriber nodes must be able to subscribe to messages in the target topic, and authorized publishers must be able to publish messages to the target topic, which can be formulated as follows:

$$\begin{aligned} &G(\neg((auth \rightarrow F(sub)) \rightarrow F(success)) \wedge \\ &\neg((auth \rightarrow F(pub)) \rightarrow F(success)) \end{aligned} \tag{3}$$

4.2. Modeling of Service Security Vulnerability

In Section 2, we introduced two vulnerabilities in the ROS2 Service communication mechanism: stealing the basic data of a service and unauthorized service calls. And in the following, we will model M_4 and M_5 for each of these two vulnerabilities.

4.2.1. Vulnerability Model

The model for stealing basic data of service is

$$M_4 = \{S_4, \Sigma_4, \delta_4, I_4\}$$

where the following are considered:

- $S_4 = \{idle, check1, auth, comm, wait\_s, check2, success, fail\}$.
  There are eight states in S_4. In these states, *idle* denotes the initial state of the vulnerability model; *auth* denotes that the intruder node used to detect the vulnerability is an authorized node; *comm* denotes that the intruder node has been able to communicate with the target; *wait_s* denotes that the intruder node waits to receive a message from the target's service; and *success* and $fail$ denote, respectively, whether the message in the service is successfully acquired or not.
- $\Sigma_4 = \{authorized, get\_domain, get\_service, return\_service\}$.
  authorized denotes to make the intruder node be an authorized node; *get_domain* denotes to get the communication domain where the target service is located; *get_service* denotes to get the information of the target service, and the intruder can obtain the data of the service through channel *return_service*.
- The initial state set $I_4 = \{idle\}$.
- The transition relationship between the states δ is shown in Figure 4.

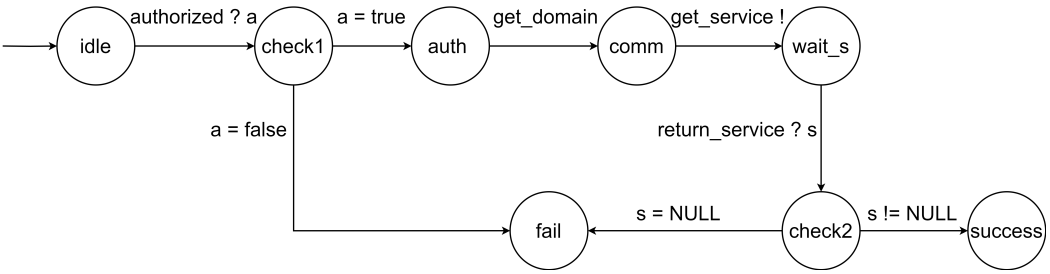

Figure 4. Stealing basic data of service.

The model for unauthorized service call is

$$M_5 = \{S_5, \Sigma_5, \delta_5, I_5\}$$

where the following are considered:

- $S_5 = \{idle, check1, auth, comm, wait\_s, check2, service, call, success, fail\}$.
  There are ten states in S_5. In these states, *service* denotes that the intruder has obtained the data of the service; *call* denotes that the intruder has sent a service request to the ROS2 system; and *success* and $fail$ represent whether the response was successfully received or not, respectively.
- $\Sigma_5 = \{authorized, get\_domain, get\_service, request\_s, return\_service, response\_s, timeout\}$.
  request_s means to send the service request, and *response_s* means to receive the service response. The rest of the actions have the same meaning as in S_5;
- The initial state set $I_5 = \{idle\}$.
- The transition relationship between the states δ_5 is shown in Figure 5.

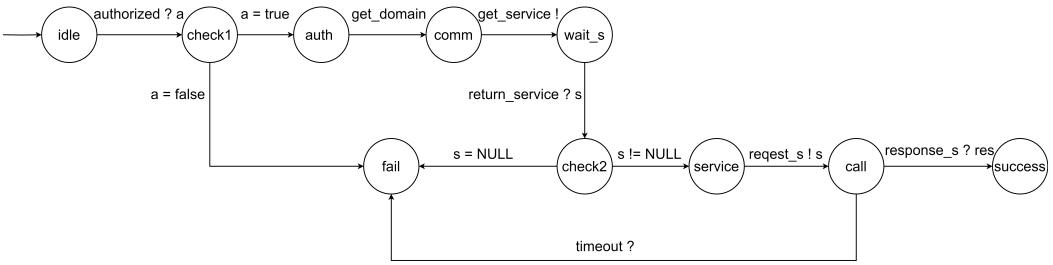

Figure 5. Unauthorized service call.

4.2.2. Security Specification

Service confidentiality in ROS2 requires that only authorized nodes have access to data related to the service, which can be broken down into the following two points:

- Only authorized nodes can access the basic information (service name, service type) of the service in the communication domain. So, the ROS2 node should satisfy

$$G(\neg((auth \rightarrow F(wait\_s)) \rightarrow F(success)))$$

- Only authorized client nodes can receive service responses sent by the server, and only authorized server nodes can receive service requests sent by the client. So, the ROS2 service client should satisfy

$$G(\neg((auth \rightarrow F(service)) \rightarrow F(success)))$$

Thus, ROS2 service confidentiality can be expressed as

$$\begin{aligned} &G(\neg((auth \rightarrow F(wait\_s)) \rightarrow F(success)) \wedge \\ &\neg((auth \rightarrow F(service)) \rightarrow F(success))) \end{aligned} \tag{4}$$

The service integrity of ROS2 requires that only authorized nodes can modify the data of the service, and only authorized client nodes can send requests to the server node. Thus, ROS2 service integrity can be expressed as

$$G(\neg((auth \rightarrow F(service)) \rightarrow F(call))) \tag{5}$$

Service availability in ROS2 requires that authorized service client nodes must be able to send requests to the server, which can be formulated as follows:

$$G(\neg((auth \rightarrow F(service)) \rightarrow F(success))) \tag{6}$$

4.3. Modeling of Action Security Vulnerability

In Section 2, we introduced two vulnerabilities in the ROS2 action communication mechanism: stealing basic the data of an action and unauthorized action calls. We will model M_6 and M_7 for each of these two vulnerabilities in the following.

4.3.1. Vulnerability Model

The model for stealing basic data of action is

$$M_6 = \{S_6, \Sigma_6, \delta_6, I_6\}$$

where the following are considered:

- $S_6 = \{idle, check1, auth, comm, wait\_a, check2, success, fail\}$.
  There are eight states in S_6. In these states, *idle* denotes the initial state of the vulnerability model; *check*1 denotes to check whether the intruder is authorized; *auth* denotes that the intruder node used to detect the vulnerability is an authorized node;

comm denotes that the intruder node is ready to communicate with the target; *wait_a* denotes that the intruder node waits to receive the message from the target's action; *check*2 denotes whether the message of action is null; and *success* and $fail$ denote, respectively, whether the response is successfully received or not.

- $\Sigma_6 = \{authorized, get\_domain, get\_action, return\_act\}$.
  authorized indicates whether the node is an authorized node or not; *get_domain* indicates the communication domain where the target action is located; *get_action* indicates that the intruder requests to get the information of the target action; and *return_act* indicates that the intruder has received the information.
- The initial state set $I_6 = \{idle\}$.
- The transition relationship between the states δ is shown in Figure 6.

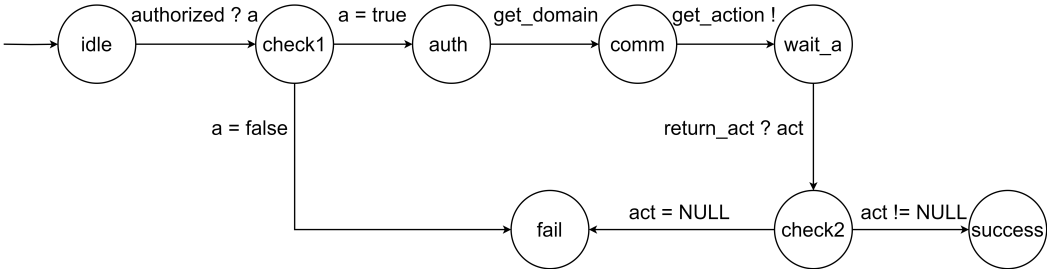

Figure 6. Stealing basic data of action.

The model for unauthorized action call is

$$M_7 = \{S_7, \Sigma_7, \delta_7, I_7\}$$

where the following are considered:

- $S_7 = \{idle, check1, auth, comm, wait\_a, check2, action, call, success, fail\}$.
  There are ten states in S_7. In these states, *action* denotes that the intruder node is ready to send a request; *call* indicates that the intruder has finished sending the request and is waiting for a response from the server; and *success* and $fail$ represent whether or not the response was successfully received, respectively. And the other states are the same as in S_6.
- $\Sigma_7 = \{authorized, get\_domain, get\_action, return\_act, request\_a, response\_a, timeout\}$.
  request_a means to send a request for the action, and *response_a* means to receive the response of the action.
- The initial state set $I_7 = \{idle\}$.
- The transition relationship between the states δ is shown in Figure 7.

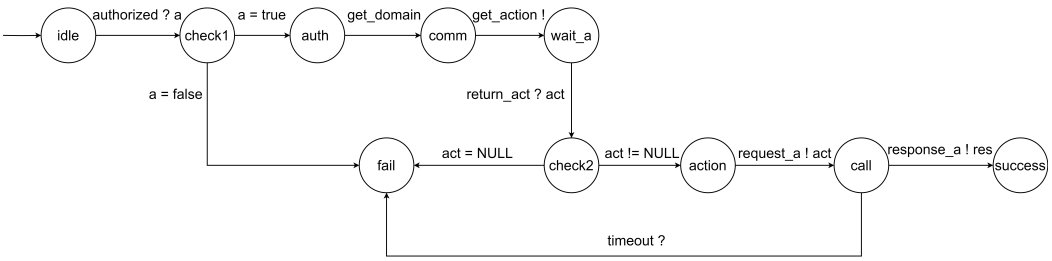

Figure 7. Unauthorized action call.

4.3.2. Security Specification

ROS2 action confidentiality requires that only authorized nodes have access to data related to the action, which can be broken down into the following two points:

- Only authorized nodes can access the basic information (action name, action type) of actions in the communication domain, so the ROS2 node should satisfy

$$G(\neg((auth \rightarrow F(wait\_a)) \rightarrow F(success)))$$

- Only authorized client nodes can receive action responses and feedback sent by the server, and only authorized server nodes can receive action requests sent by the client. So, the action client should satisfy

$$G(\neg((auth \rightarrow F(action)) \rightarrow F(success)))$$

Thus, the action confidentiality can be expressed as

$$\begin{aligned} &G(\neg((auth \rightarrow F(wait\_a)) \rightarrow F(success)) \wedge \\ &\neg((auth \rightarrow F(action)) \rightarrow F(success))) \end{aligned} \tag{7}$$

ROS2's action integrity requires that only authorized nodes can modify the action's data, and only authorized client nodes can send requests to server nodes, which can be formulated as follows:

$$G(\neg((auth \rightarrow F(action)) \rightarrow F(call))) \tag{8}$$

Action availability in ROS2 requires that authorized service client nodes must be able to send requests to the server, which can be formulated as follows:

$$G(\neg((auth \rightarrow F(action)) \rightarrow F(success))) \tag{9}$$

5. ROS2 Communication Security Vulnerability Detection Method

In order to analyze ROS2 security, we designed and developed ROS2Tester, a tool for the vulnerability detection of the ROS2 system. In this section, we describe and illustrate the design and implementation of the tool.

5.1. Framework of Method

In order to test and analyze the impact of the above attacks on ROS2 applications, we designed a vulnerability detection method and tool for ROS2 in which these attacks have been implemented and integrated. Our method was designed to detect potential security vulnerabilities in ROS2 systems by conducting penetration testing [40]. This method comprises two modules: a communication domain scanning module and a vulnerability detection module, as shown in Figure 8.

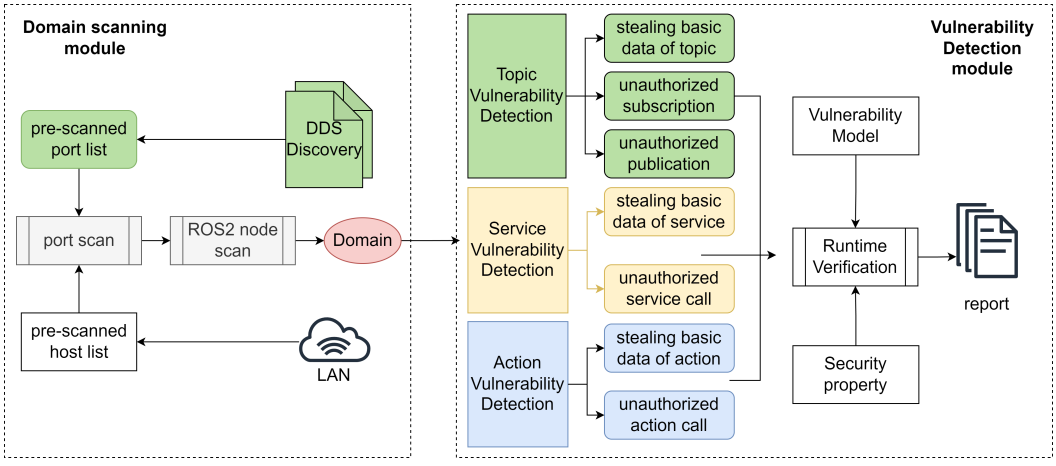

Figure 8. Framework of method.

The domain scanning module uses network port scanning to locate ROS2 systems running in the LAN. The vulnerability detection module scans for vulnerabilities according to a formal method and generates a vulnerability detection report. This report includes details of the detection target, information about the vulnerabilities detected in the target, and the results of vulnerability attacks. It provides a comprehensive analysis of the security issues identified in the scanned ROS2 system.

5.2. Domain Scanning Module

The communication domain scanning module is responsible for detecting active ROS2 systems in the local area network (LAN) and passing their communication domain IDs to the vulnerability detection module. The module is composed of three parts: pre-scan port calculation, port scanning, and ROS2 node scanning. The overall flow of the module is shown in Figure 9.

As per the DDS protocol, ROS2 nodes in the same communication domain can discover and communicate with each other. To determine the communication domain ID of the ROS2 system, this module performs a UDP port scan in reverse. To optimize the scanning efficiency, ROS2Tester calculates all UDP ports that may be utilized for ROS2 communication using the DDS discovery protocol before conducting port scanning, resulting in a pre-scanned port list.

To obtain the pre-scanned port list, the module employs the DDS discovery protocol, which has a discovery broadcast port in each DDS communication domain for mutual discovery between nodes in the communication domain. The port and the domain ID have a corresponding transformation relationship, which is determined by the following formula:

$$DiscoveryMulticastPort = PB + DG * DomainID \tag{10}$$

Here, PB is a constant value of 7400, which indicates the starting port number, i.e., the discovery broadcast port of the domain with DomainID 0, and DG is a constant value of 250, which indicates the maximum number of ports that can be included in a domain.

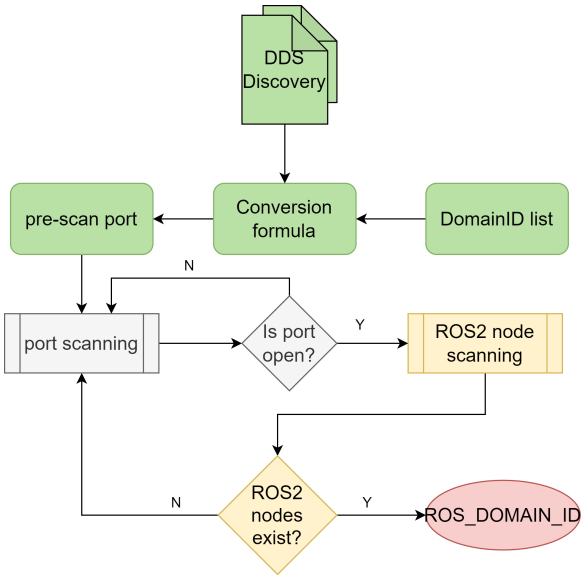

Figure 9. Domain scanning module.

As the communication domain ID of ROS2 ranges from 0 to 232, the pre-scanned port list can be calculated accordingly. Subsequently, ROS2Tester employs the network scanning tool Nmap [41] to scan the ports sequentially in the pre-scanned port list. If the port is active, it conducts a node scan to confirm whether a ROS2 node is running on it. If a ROS2 node is found, the module returns the corresponding domain ID. If not, it continues to scan the next port.

5.3. Vulnerability Detection Module

The vulnerability detection module is used to simulate an intruder and perform network attacks on the ROS2 application to test its security. According to the type of ROS2 communication mechanism. The module is divided into three sub-modules: a topic vulnerability detection module, service vulnerability detection module, and action vulnerability detection module.

5.3.1. Topic Vulnerability Detection Module

In the Topic Vulnerability Detection Module, the tool mainly carries out the three attacks of stealing the basic data of the topic, unauthorized subscription, and unauthorized publication to detect vulnerabilities in the target system. And the tool also performs runtime verification in the process of vulnerability detection. The implementation flow of this module is shown in Figure 10.

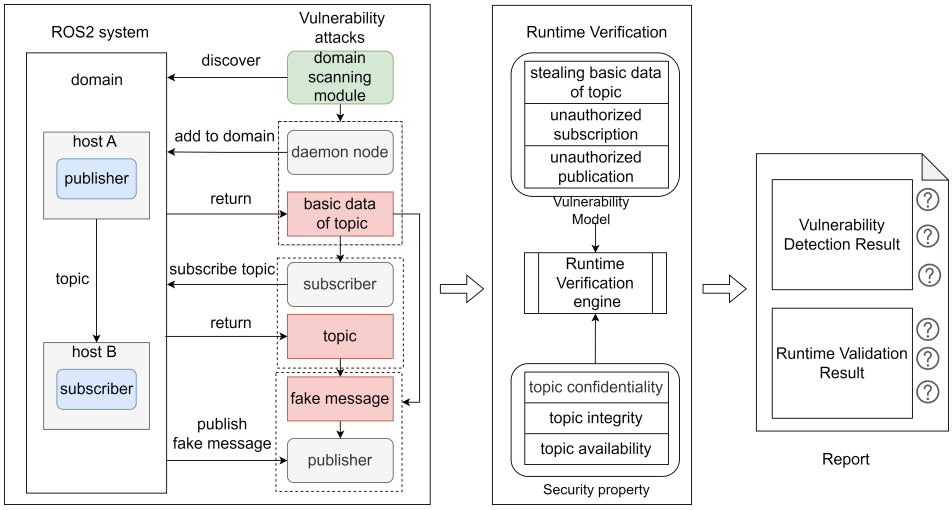

Figure 10. Topic vulnerability detection module.

The left side of the figure shows the vulnerability detection function for a running ROS2 system. First, the tool will discover the target system based on the aforementioned communication domain scanning module. Then, the tool will establish an intruder node and link it into the communication of that system. After successfully joining the communication network, the intruder node will create a daemon node, which will be used to continuously obtain the information of other nodes. Finally, the intruder will obtain the type of message in the topic by calling the ROS2 API to detect the vulnerability of stealing the basic data of the topic.

Based on the topic name and topic type, the module first creates an attacker node in the target domain and then creates subscribers on this node for subscribing to the target topic. Finally, the attacker node is launched to subscribe to the target topic and obtain the data in the target topic. Similarly, the module can also send false data to the target topic by creating a publisher, and all the subscribers of the topic will be subjected to the false data to detect unauthorized publication vulnerability.

During the entire vulnerability detection process, the tool will record the behavior and state changes of the intruder and the system at the same time. Then, this information will correspond to the vulnerability models. Based on this, the tool will perform a runtime verification of the system, to verify whether it meets the confidentiality, integrity, and availability under different vulnerability attacks. The verification results will be included in the inspection report together with the vulnerability detection results. The specific implementation interface is shown in Figure 11.

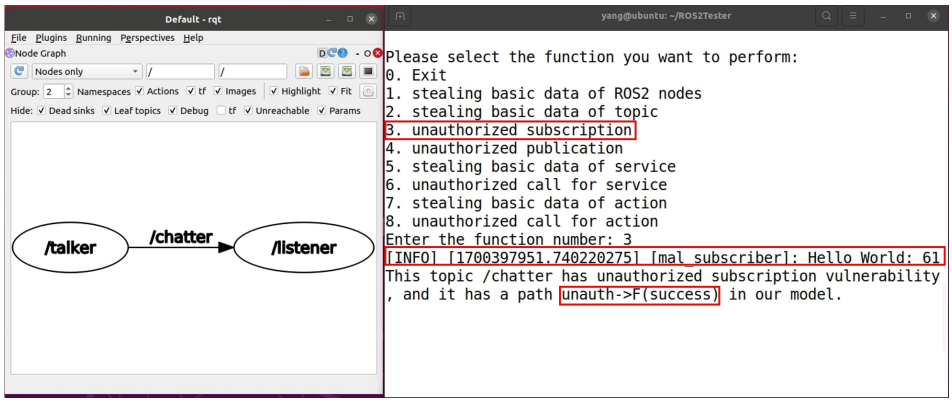

Figure 11. Topic vulnerability detection module.

5.3.2. Service Vulnerability Detection Module

In this module, we use two types of attacks to detect vulnerabilities: stealing the basic data of a service and unauthorized service calls. If successful, these attacks indicate the existence of vulnerabilities in the target system. The implementation flow of this module is shown in Figure 12.

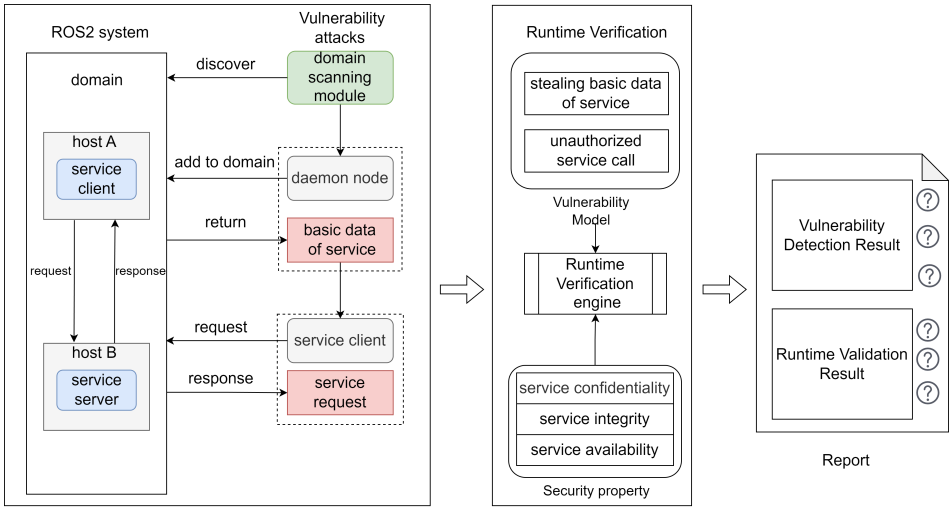

Figure 12. Service vulnerability detection module.

To carry out an unauthorized service call attack, an attacker can obtain the basic data of a service, including its name, type, and interface information, by creating daemon nodes. With this information, the attacker can create an intruder node and service client on the target domain, assign parameters, and issue a request to the server, waiting for a response.

The impact of the attack on the ROS2 application depends on the service function. If the function is related to data access, the attack may compromise the data confidentiality of the ROS2 application. However, if the function is related to data or command writing, the attack may compromise the data integrity of the ROS2 application.

As with the topic vulnerability detection module, during the entire vulnerability detection process, the tool will record the intruder and the system at the times of behavior and state changes, and then the stealing-of-basic-service-data vulnerability model and unauthorized service call model will be verified. Finally, the results of the verification will be written into the detection report.

5.3.3. Action Vulnerability Detection Module

In the action vulnerability detection module, we mainly detect vulnerabilities in the target system using two kinds of attacks, namely, stealing the basic data of an action and

unauthorized action calls, and if an attack is successful, the corresponding vulnerability exists in the target system. The implementation flow of this module is shown in Figure 13.

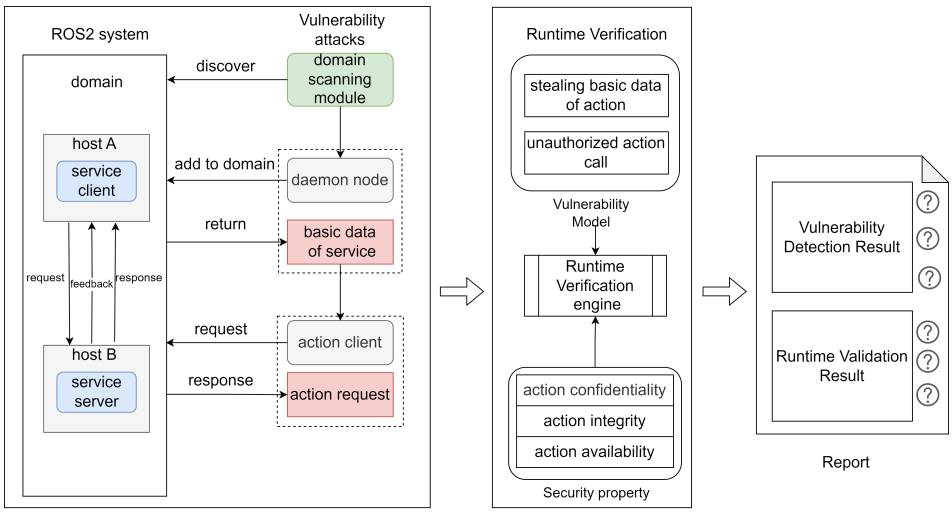

Figure 13. Action vulnerability detection module.

The basic data of an action include its name, type, and interface information. With these data obtained using a daemon node, an attacker can make unauthorized action calls. The attacker can create an intruder node in the target domain, call the relevant API to create an action client on this node, assign parameters required for the action request, and send the request to the action server, waiting for feedback and a response. Depending on the function of the invoked action, this attack can be classified as either a confidentiality or integrity attack, similar to an unauthorized service attack.

Similarly, during the entire vulnerability detection process, the tool will record the intruder and the system at the times of behavior and state changes. And then the stealing-of-basic-action-data vulnerability model and unauthorized action call model will be verified. Finally, the results of the verification will be written into the detection report.

In addition to implementing target-specific functionality, the tool also automates the detection of vulnerabilities across the entire ROS2 application throughout the vulnerability detection module. When choosing to perform an attack manually, the user can select a specific target, such as a topic or service, and then manually assign values to the parameters required for the attack so that the appropriate parameter values can make the results of the attack more obvious. The goal of automated vulnerability detection, on the other hand, is to quickly detect vulnerabilities in an entire ROS2 application in order to find the parts of it that could be attacked.

6. Experiment

In order to test the usability of ROS2Tester and the security vulnerabilities of the ROS2 system, we set up a series of experiments. In this section, we describe the environment setup and the results of the experiments.

6.1. Experiment Environment

In order to test the usability of the tool and the communication security of the ROS2 system, we used three hosts in the same LAN to build the required experimental network environment, with the same hardware and software configurations for the three hosts.

The experimental network environment is shown in Figure 14. Host A and host B form an ROS2 system in which ROS2 nodes will be created for communication during the experiment; host C acts as an intruder and attacks the ROS2 system of host A and B components.

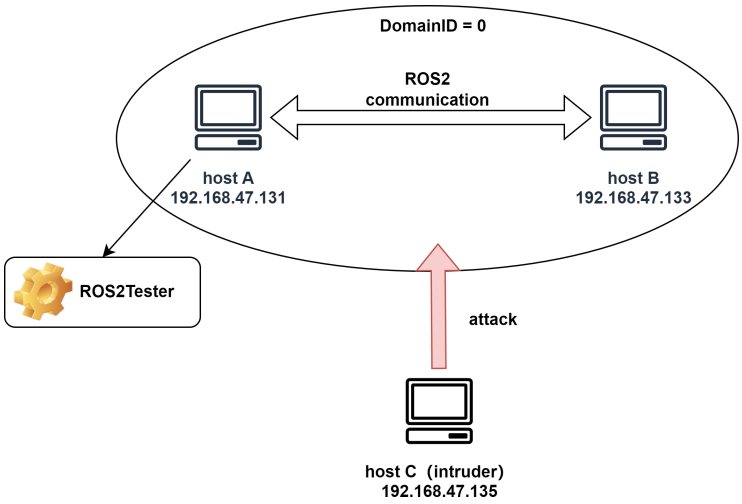

Figure 14. Experimental environment.

In this experiment, in order to detect the security vulnerabilities of the three communication mechanisms of ROS2 topics, services, and actions in a complete way, we ran the systems that carry out these three types of communication in hosts A and B. The specific information is shown in Figure 15.

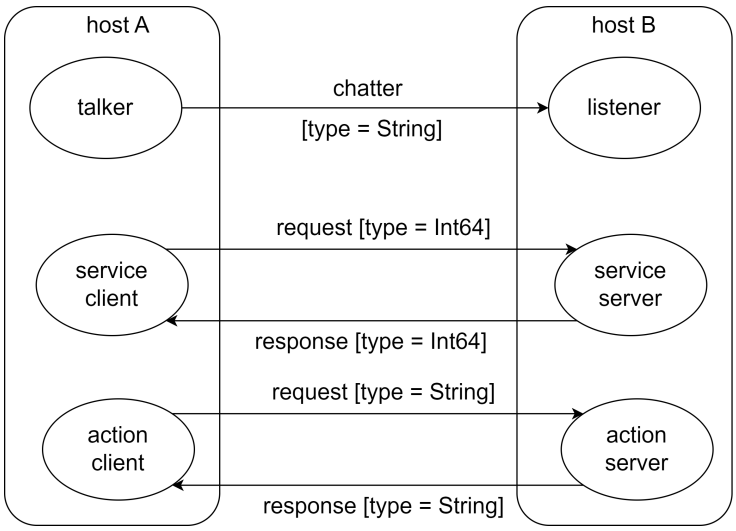

Figure 15. ROS2 system communication structure.

In this system, topic communication is carried out between a node talker and node listener. The name of the topic is chatter, and the data type of the transmitted message in the topic is String. Service communication is carried out between the node service client and node service server. The service client sends a service request to the server and transmits two integer values of type Int64. The service server performs an addition operation on these two integers and transmits an integer value of type Int64 as a response. Action communication is performed between the node action client and node action server. The action client sends a target request to the server and transmits a data of type String. The action server processes the data, controls the drawing of a circle on its own canvas, and sends a response of type String.

Through the above experimental environment, we can achieve the testing and analysis of the security of ROS2 in its default state, but in order to further analyze the security of SROS2, we can protect all the nodes in the above hosts A and B with SROS2, and then we can use ROS2Tester to carry out the same vulnerability detection, comparing the security of SROS2 in the use of SROS2 and the inapplicability of SROS2 to the security changes.

6.2. Results and Analysis

6.2.1. Runtime Verification Results

In this experiment, we performed a runtime verification of the ROS2 system to verify that it can satisfy the communication security properties in case of some type of vulnerability attack, as shown in Table 1.

Table 1. Runtime verification results.

Type of Vulnerability \ Security Property	Confidentiality	Integrity	Availability
stealing basic data of the topic	×	✓	✓
unauthorized subscription	×	✓	×
unauthorized publication	×	×	×
stealing basic data of service	×	✓	✓
unauthorized service call	×	×	×
stealing basic data of action	×	✓	✓
unauthorized action service	×	×	×

In the table, "✓" indicates that the system still satisfies the corresponding security attribute under the vulnerability attack corresponding to that row, and "×" indicates that the system does not satisfy the corresponding security attribute. For the stealing of the basic data of the topic, unauthorized subscription, and unauthorized publication, confidentiality, integrity, and availability in the table denote the relevant security attributes of the topic. Similarly, stealing the basic data of a service and unauthorized service calls denote the relevant security attributes of the service, and the action vulnerabilities denote the relevant security attributes of the action.

From the experimental result data, it can be seen that for topic communication, the stealing of basic data through the topic vulnerability attack only destroys the confidentiality of the system. The unauthorized subscription destroys confidentiality and availability. The unauthorized publication is capable of destroying all the three security attributes. Analyzed in principle, the attacker's theft of some basic information may only lead to the risk of data leakage, and they are unable to inject malicious data into the system or affect the normal communication between the system nodes. Therefore, these vulnerabilities cannot damage the integrity or availability of the system.

Similarly, for service communication and action communication, the theft of basic information cannot serve the purpose of data injection or the disruption of system communication and, therefore, cannot compromise the integrity and availability of the system. For unauthorized service calls and unauthorized action calls, they can be used both by an attacker to steal confidential data from the system and to inject false data or malicious commands into the system, so they have the capability of compromising all of the three security properties.

6.2.2. Vulnerability Detection Result

In this experiment, we performed vulnerability testing on the system for all seven vulnerabilities as a means of identifying the communication security vulnerabilities that exist in the system. In addition, in order to test the actual effect of SROS2 on the security enhancement of ROS2, as well as to contrast with ROS2, we built the system on the framework of SROS2, which is consistent with the original experiment, and divided the attackers into unauthorized nodes and authorized nodes to conduct the experiment to test the degree of security protection of SROS2, and the results of the experiments are shown in Table 2.

Table 2. Detection results.

Attacker Node	Type of Attack	ROS2	SROS2
unauthorized node	stealing basic data of the topic	✓	×
	unauthorized subscription	✓	×
	unauthorized publication	✓	×
	stealing basic data of service	✓	×
	unauthorized service call	✓	×
	stealing basic data of action	✓	×
	unauthorized action call	✓	×
authorized node	stealing basic data of the topic	✓	✓
	unauthorized subscription	✓	✓
	unauthorized publication	✓	✓
	stealing basic data of service	✓	✓
	unauthorized service call	✓	✓
	stealing basic data of action	✓	✓
	unauthorized action call	✓	✓

In the table, "✓" indicates that the system has the corresponding vulnerability. On the contrary, "×" indicates that the system does not have the corresponding vulnerability. From the experimental results, we can see that regardless of whether the attacker is an authorized node or not, the system built based on ROS2 has seven communication security vulnerabilities. While the system built based on SROS2 can withstand these seven vulnerability attacks from unauthorized nodes, but cannot withstand vulnerability attacks from authorized attacker nodes. This shows that the authentication mechanism of SROS2 can indeed play a security role, but due to the lack of a reliable scheme for the remote transmission of security profiles of SROS2 at present, there is a risk of being intercepted, and the attacker can use the intercepted profiles to achieve identity authentication, thus achieving the purpose of vulnerability attacks.

6.2.3. Tool Performance

Our tool, ROS2Tester, can currently detect seven types of security vulnerabilities, and we can subsequently consider expanding the vulnerabilities detected, such as authentication and other types of vulnerabilities. ROS2Tester can be run in an environment with various versions of ROS2; the version of this experiment was ROS2 Foxy, and its performance is shown in Figure 16.

To ensure that the vulnerability detection time for ROS2-based applications with a large number of nodes is acceptable, we tested ROS2-based applications with a large number of nodes. As shown in Figure 16, in order to fully evaluate the performance of the tool, we evaluated systems consisting of 100, 500 and 1000 communicating entities such as nodes or topics, respectively. The results show that the performance of the tool can be affected by the communication frequency settings and the size of the communication data in the system itself. In our tests, vulnerability scans for 1000 actions that manage common tasks were completed in 15 min. These results suggest that while scanning time increases linearly with the number of nodes or topics, the tool's ability to quickly complete vulnerability detection tasks is feasible even for larger systems when the system's own communication performance is good.

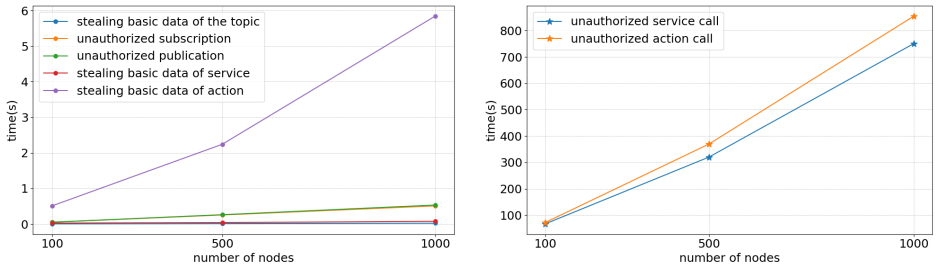

Figure 16. Tool Performance.

6.3. Comparison of Related Tools

Since ROS2 is still in the development stage, there is no vulnerability detection tool for ROS2 like ROS2Tester, but ROS and ROS2 have many similarities in terms of vulnerability detection. Therefore, in order to better reflect the perfect function of ROS2Tester, we compared ROS2Tester with ROSPenTo [27] and ROSploit [28].

ROSPenTo and ROSploit are both tools that use penetration testing to detect vulnerabilities in ROS, and their implementation principles are basically the same. Since ROS communication relies on the master node as the central node for communication, both tools further control other nodes in the system by gaining control of the master node. Table 3 compares the vulnerabilities that can be detected by ROS2Tester, ROSPenTo, and ROSploit, and it can be seen from the table that each of the three tools has its own focus.

Table 3. Comparison of tool detection vulnerability coverage.

Tool Name / Vulnerability Type	ROS2Tester	ROSPenTo	ROSploit
stealing basic data of node	✓	✓	✓
Impersonating node identity		✓	
stealing basic data of the topic	✓	✓	✓
unauthorized subscription	✓	✓	✓
unauthorized publication	✓	✓	✓
stealing basic data of service	✓	✓	
unauthorized service call	✓		
stealing basic data of action	✓		
unauthorized action call	✓		
stealing basic data of parameter		✓	✓
modify node parameter information		✓	✓

As can be seen from Table 3, ROSploit and ROSPenTo focus more on vulnerabilities related to accessing and modifying information on the ROS nodes themselves, as well as security vulnerabilities related to parameter servers, while ROS2Tester focuses on security vulnerabilities related to the three types of communication mechanisms: ROS2 topics, services, and actions. Since publish–subscribe is the most commonly used communication mechanism in ROS and ROS2, all three tools focus heavily on vulnerability detection in this area. The difference is that ROS2Tester is not able to detect the vulnerabilities of node impersonation and node parameter modification like the other two tools, which is mainly due to the difference in the underlying communication architectures of ROS and ROS2. ROS adopts a centralized communication model, where all the nodes need to be registered and logged out of the master node, and all the nodes need to obtain parameters from the global parameter server. All nodes need to subscribe to a global parameter server to obtain parameters. As for ROS2, it adopts a distributed communication model. There is no central node controlling all nodes, so it is impossible to control other nodes through malicious

nodes and achieve the purpose of impersonating other nodes. Meanwhile, compared with the global parameter server in ROS, ROS2 places more emphasis on the flexibility of the distributed system, so the implementation of parameter service may be decentralized. Therefore, it is also difficult for intruders to achieve the modification of node parameters.

7. Conclusions

In this study, we formally modeled common communication security vulnerabilities in ROS2 applications and used LTL to represent the CIA security properties that ROS2 needs to satisfy. In addition, we designed and developed a communication security vulnerability detection tool for ROS2 based on a reachability analysis, through which we can detect the existence of relevant security vulnerabilities in specific ROS2 applications and analyze which property of the CIA security properties is broken by the detected vulnerabilities.

ROS2 is still in rapid development, but there is a lack of tools for security testing. The work in this article explored some of these aspects, which are very favorable to enhancing the security of ROS2. In the future, we will consider validating additional security specifications such as authentication, identity authorization, etc. Also, we will consider the use of runtime verification for vulnerability detection to ensure that the purpose of security detection is achieved without interfering with the normal operation of the system. In this study, we built formal models for ROS2 vulnerabilities and used these models to verify security specifications. In future work, we will consider incorporating these models into the runtime ROS2 system and verifying the security of the system at runtime using the collected data information as input for the models. In fact, in this study, we already integrated a runtime verification module into the tool. Our primary aim is to isolate this module so that it can operate independently within the ROS2 system, serving a function analogous to that of a monitor, The brief process of this is shown in Figure 17.

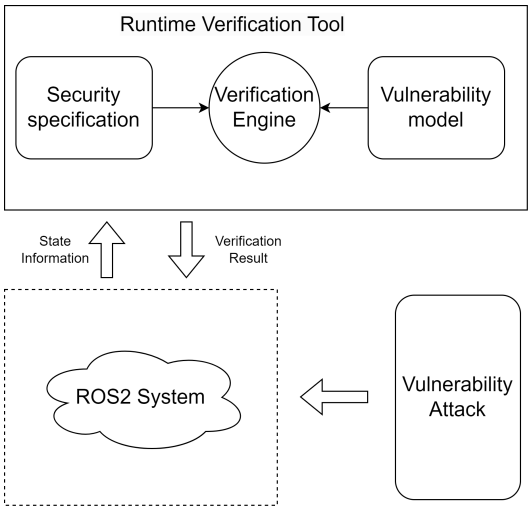

Figure 17. Overflow of future work.

Author Contributions: Conceptualization, S.Y. and J.G.; methodology, S.Y., J.G. and X.R.; software, S.Y.; validation, S.Y. and J.G.; formal analysis, S.Y.; investigation, S.Y. and X.R.; resources, S.Y. and J.G.; writing—original draft preparation, S.Y.; writing—review and editing, S.Y., J.G. and X.R.; supervision, S.Y. and J.G.; project administration, J.G.; funding acquisition, J.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Key Research and Development Program (Grant 2022YFB3104002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

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

References

1. Gonzalez-Aguirre, J.A.; Osorio-Oliveros, R.; Rodríguez-Hernández, K.L.; Lizárraga-Iturralde, J.; Morales Menendez, R.; Ramírez-Mendoza, R.A.; Ramírez-Moreno, M.A.; Lozoya-Santos, J.d.J. Service robots: Trends and technology. *Appl. Sci.* **2021**, *11*, 10702. [CrossRef]
2. Wang, Z.; Tian, G.; Shao, X. Home service robot task planning using semantic knowledge and probabilistic inference. *Knowl.-Based Syst.* **2020**, *204*, 106174. [CrossRef]
3. Belanche, D.; Casaló, L.V.; Flavián, C.; Schepers, J. Service robot implementation: A theoretical framework and research agenda. *Serv. Ind. J.* **2020**, *40*, 203–225. [CrossRef]
4. Kyrarini, M.; Lygerakis, F.; Rajavenkatanarayanan, A.; Sevastopoulos, C.; Nambiappan, H.R.; Chaitanya, K.K.; Babu, A.R.; Mathew, J.; Makedon, F. A survey of robots in healthcare. *Technologies* **2021**, *9*, 8. [CrossRef]
5. Kazanzides, P.; Chen, Z.; Deguet, A.; Fischer, G.S.; Taylor, R.H.; DiMaio, S.P. An open-source research kit for the da Vinci® Surgical System. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6434–6439.
6. He, W.; Ge, S.S.; Li, Y.; Chew, E.; Ng, Y.S. Neural network control of a rehabilitation robot by state and output feedback. *J. Intell. Robot. Syst.* **2015**, *80*, 15–31. [CrossRef]
7. Mintrom, M.; Sumartojo, S.; Kulić, D.; Tian, L.; Carreno-Medrano, P.; Allen, A. Robots in public spaces: Implications for policy design. *Policy Des. Pract.* **2022**, *5*, 123–139. [CrossRef]
8. Luo, R.C.; Chou, Y.T.; Liao, C.T.; Lai, C.C.; Tsai, A.C. NCCU security warrior: An intelligent security robot system. In Proceedings of the IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society, Taipei, Taiwan, 5–8 November 2007; pp. 2960–2965.
9. International Federation of Robotics. Top 5 Robot Trends 2021. 2022. Available online: https://ifr.org/ifr-press-releases/news/top-5-robot-trends-2021 (accessed on 29 April 2024).
10. Plósz, S.; Schmittner, C.; Varga, P. Combining safety and security analysis for industrial collaborative automation systems. In Proceedings of the Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, 12 September 2017; Proceedings 36; Springer: Berlin/Heidelberg, Germany, 2017; pp. 187–198.
11. Kirschgens, L.A.; Ugarte, I.Z.; Uriarte, E.G.; Rosas, A.M.; Vilches, V.M. Robot hazards: From safety to security. *arXiv* **2018**, arXiv:1806.06681.
12. Lacava, G.; Marotta, A.; Martinelli, F.; Saracino, A.; La Marra, A.; Gil-Uriarte, E.; Vilches, V.M. Cybsersecurity Issues in Robotics. *J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl.* **2021**, *12*, 1–28.
13. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: An open-source Robot Operating System. In Proceedings of the ICRA Workshop on Open Source Software, Kobe, Japan, 12–17 May 2009; Volume 3, p. 5.
14. Değirmenci, E.; Kirca, Y.S.; Yolaçan, E.N.; Yazici, A. An Analysis of DoS Attack on Robot Operating System. *Gazi Univ. J. Sci.* **2023**, *36*, 1050–1069. [CrossRef]
15. Zhai, G.; Zhang, W.; Hu, W.; Ji, Z. Coal mine rescue robots based on binocular vision: A review of the state of the art. *IEEE Access* **2020**, *8*, 130561–130575. [CrossRef]
16. Vuong, T.; Filippoupolitis, A.; Loukas, G.; Gan, D. Physical indicators of cyber attacks against a rescue robot. In Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS), Budapest, Hungary, 24–28 March 2014; pp. 338–343.
17. Khan, A.T.; Li, S.; Cao, X. Human guided cooperative robotic agents in smart home using beetle antennae search. *Sci. China Inf. Sci.* **2022**, *65*, 122204. [CrossRef]
18. Brondi, S.; Pivetti, M.; Di Battista, S.; Sarrica, M. What do we expect from robots? Social representations, attitudes and evaluations of robots in daily life. *Technol. Soc.* **2021**, *66*, 101663. [CrossRef]
19. Coble, K.; Wang, W.; Chu, B.; Li, Z. Secure software attestation for military telesurgical robot systems. In Proceedings of the 2010-Milcom 2010 Military Communications Conference, San Jose, CA, USA, 31 October–3 November 2010; pp. 965–970.
20. Jang, S.M.; Hong, Y.J.; Lee, K.; Kim, S.; Chiến, B.V.; Kim, J. Assessment of user needs for telemedicine robots in a developing nation hospital setting. *Telemed. E-RHealth* **2021**, *27*, 670–678. [CrossRef] [PubMed]
21. Javaid, A.Y.; Sun, W.; Devabhaktuni, V.K.; Alam, M. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. In Proceedings of the 2012 IEEE Conference on Technologies for Homeland Security (HST), Waltham, MA, USA, 13–15 November 2012; pp. 585–590.
22. Groza, B.; Dragomir, T.L. Using a cryptographic authentication protocol for the secure control of a robot over TCP/IP. In Proceedings of the 2008 IEEE International Conference on Automation, Quality and Testing, Robotics, Cluj-Napoca, Romania, 22–25 May 2008; Volume 1, pp. 184–189.
23. Lee, G.S.; Thuraisingham, B. Cyberphysical systems security applied to telesurgical robotics. *Comput. Stand. Interfaces* **2012**, *34*, 225–229. [CrossRef]
24. GvdHoorn. Security about ROS. 2020. Available online: http://wiki.ros.org/Security (accessed on 29 April 2024).
25. Breiling, B.; Dieber, B.; Schartner, P. Secure communication for the robot operating system. In Proceedings of the 2017 Annual IEEE International Systems Conference (SysCon), Montreal, QC, Canada, 24–27 April 2017; pp. 1–6.

26. Arkin, B.; Stender, S.; McGraw, G. Software penetration testing. *IEEE Secur. Priv.* **2005**, *3*, 84–87. [CrossRef]
27. Dieber, B.; White, R.; Taurer, S.; Breiling, B.; Caiazza, G.; Christensen, H.; Cortesi, A. Penetration Testing ROS. In *Robot Operating System (ROS): The Complete Reference*; Koubaa, A., Ed.; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2020; Volume 4, pp. 183–225.
28. 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.
29. Dieber, B.; Breiling, B.; Taurer, S.; Kacianka, S.; Rass, S.; Schartner, P. Security for the robot operating system. *Robot. Auton. Syst.* **2017**, *98*, 192–203. [CrossRef]
30. Halder, R.; Proença, J.; Macedo, N.; Santos, A. Formal Verification of ROS-Based Robotic Applications Using Timed-Automata. In Proceedings of the 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), Buenos Aires, Argentina, 27 May 2017; pp. 44–50.
31. Huang, J.; Erdogan, C.; Zhang, Y.; Moore, B.; Luo, Q.; Sundaresan, A.; Rosu, G. ROSRV: Runtime Verification for Robots. In *Runtime Verification*; Bonakdarpour, B., Smolka, S.A., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2014; pp. 247–254.
32. Rivera, S.; State, R. Securing Robots: An Integrated Approach for Security Challenges and Monitoring for the Robotic Operating System (ROS). In Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), Bordeaux, France, 17–21 May 2021; pp. 754–759.
33. Sundaresan, A.; Gerard, L.; Kim, M. Secure ROS. Available online: http://secure-ros.csl.sri.com/ (accessed on 29 April 2024).
34. Mayoral-Vilches, V.; White, R.; Caiazza, G.; Arguedas, M. Sros2: Usable cyber security tools for ros 2. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 11253–11259.
35. Open Source Robotics Foundation. ROS2 Robotic Systems Threat Model. 2019. Available online: http://design.ros2.org/articles/ros2_threat_model.html (accessed on 29 April 2024).
36. Kim, J.; Smereka, J.M.; Cheung, C.; Nepal, S.; Grobler, M. Security and Performance Considerations in ROS 2: A Balancing Act. *arXiv* **2018**, arXiv:1809.09566.
37. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the performance of ROS2. In Proceedings of the 13th International Conference on Embedded Software, Pittsburgh, PA, USA, 2–7 October 2016; pp. 1–10.
38. Deng, G.; Xu, G.; Zhou, Y.; Zhang, T.; Liu, Y. On the (In) Security of Secure ROS2. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; pp. 739–753.
39. Camacho, A.; Icarte, R.T.; Klassen, T.Q.; Valenzano, R.A.; McIlraith, S.A. LTL and Beyond: Formal Languages for Reward Function Specification in Reinforcement Learning. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), Macao, China, 10–16 August 2019; Volume 19, pp. 6065–6073.
40. Bacudio, A.G.; Yuan, X.; Chu, B.T.B.; Jones, M. An overview of penetration testing. *Int. J. Netw. Secur. Its Appl.* **2011**, *3*, 19. [CrossRef]
41. Orebaugh, A.; Pinkard, B. *Nmap in the Enterprise: Your Guide to Network Scanning*; Elsevier: Amsterdam, The Netherlands, 2011.