ROS2 Communication Security Vulnerability Detection Based on Formal Method

Shuo Yang , Jian Guo * , Xue Rui

Posted Date: 2 April 2024

doi: 10.20944/preprints202404.0143.v1

Keywords: Robotic System; ROS2; Communication mechanisms; Security and Safety Analysis; formal method

Article

ROS2 Communication Security Vulnerability Detection Based on Formal Method

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

[1] MoE Engineering Research Center for Software/Hardware Co-design Technology and Application, ECNU; 51215902140@stu.ecnu.edu.cn.
[2] National Trusted Embedded Software Engineering Technology Research Center, ECNU; jguo@sei.ecnu.edu.cn.
[3] Xinjiang Teacher's College.
* 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 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 paper, we adopt a formal approach to analyze the security of the communication mechanism of ROS2. First, we use 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 introduce the CIA model on the basis of the established vulnerability model and use LTL to define its security properties. Then, we design and implement a vulnerability detection tool for ROS2 applications based on the vulnerability model and security properties. Finally, we experimentally test 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

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. 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 is rapidly increasing. How to test, analyse and enhance the security of robotic systems has become a critical issue as there may be a large amount of confidential or private data in the communication and any failure in its network may also lead to data leakage or system failure [9–11].

In the Robot Operating System (ROS) [12], 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 [13].

Same as ROS1, ROS2 does not have a unified standard for security measurement, so we analyze its security using the CIA model, which is common in information security. In addition, we formally modelled 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 paper is as follows:

- For the different communication mechanisms of ROS2, we have formally modelled and analyzed the potential vulnerabilities in it, and at the same time formally expressed the CIA properties as the security properties;

 doi:10.20944/preprints202404.0143.v1

- 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 reachability analysis and analyzes which properties in the ROS2 CIA are damaged by the detected vulnerabilities.

The rest of the paper is organized as follows. In Section II, We list some of the related work. In Section III, we present the possible communication security vulnerabilities in the different communication mechanisms of ROS2. In Section IV, we formally model and analyze the proposed communication security vulnerability model and the CIA properties that ROS2 needs to satisfy. In Section V, we present the design and implementation of ROS2Tester, an ROS2 vulnerability detection tool. In Section VI, we build the implementation environment for ROS2 security testing and test it using ROS2Tester. In Section VII, the work of this paper is summarized and future research directions are presented.

2. Related Work

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 [13]. 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 system.

At the early stage of ROS development, most of the 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. Rescue robots [14,15], household robots [16,17], telemedicine robots [18], drones [19], etc. Risk analyses also focus on remote communication and control security, such as efficient remote authentication [20], telemedicine protocols [21], etc. These studies reflect the lack of a unified underlying framework for robots, and the research on robot security did not focus on the underlying architecture of ROS/ROS2.

With the gradual maturity of ROS development, more and more researchers began to focus on the impact of 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 emerged. In order to achieve an in-depth analysis of ROS communication security, researchers have explored and summarised it from many different directions. Among them, the researchers test the communication security vulnerabilities of ROS from the perspective of simulated attacks by means of penetration testing [22], of which the typical testing tools are ROSPenTo [23] and ROSploit [24]. ROSPenTo is the first tool to achieve a vulnerability attack on the communication of ROS nodes, parameters, and 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 successfully verified that ROS has security vulnerabilities such as unauthorized publication, unauthorized data access and denial-of-service attacks [25], and proposed a security protection solution in the direction of the application layer for solving such problems. In addition, some researchers have also conducted research, such as using formal methods for verifying the security and reliability of the ROS communication [26] , using runtime verification to enable state monitoring of a running robotic system [27].

Along with the security analysis of ROS, in order to enhance the security of ROS, a number of researchers have proposed a series of schemes, most of which use encryption and authentication mechanisms [28]. However, these solutions cannot fully protect against sophisticated security attacks, so other researchers have explored many other directions for ROS security. ROS_Immunity [29] integrates internal system defences, external system authentication and automated vulnerability detection with Secure–ROS [30] together to provide a set of defences for ROS systems against malicious attackers.

 doi:10.20944/preprints202404.0143.v1

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 ROS, the ROS2 middleware framework is based on the DDS protocol, which provides better performance and security, and therefore the latest robot manufacturers prefer ROS2-based applications. Although ROS2 uses DDS Security and SROS2 [31] To enhance its security and usability, but it is not absolutely secure. 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 [32,33] have analysed 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 papers [32,34] show that SROS2 still has flaws at present by analysing the security analysis of DDS Security or SROS2.

3. ROS2 Communication Security Vulnerability

3.1. Security Vulnerability of Topic

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 get 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 the subscriber after getting the data related to the target topic so as to obtain the data in the topic.

(3) Unauthorized Publication

Same as 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.

 doi:10.20944/preprints202404.0143.v1

3.2. Security Vulnerability of Service

Service communication is a communication mechanism based on 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 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 data of service

In addition to the sensitive data contained within the ROS2 service, the service itself has some data information, such as 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 join 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 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 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

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 data of action

In addition to the sensitive data contained within the ROS2 service, the action itself has some data information, such as 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 action, and 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, where:

 doi:10.20944/preprints202404.0143.v1

- 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) to represent the confidentiality, integrity, and availability of the CIA security properties. LTL formulates over the set AP of atomic proposition 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:

- a: a holds at the current time, and in the trajectory behaves as if it holds at the first position.
- $X\varphi$: φ holds in the next time 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 φ holds sometime in the future.
- $G\varphi$: $G\varphi = \neg\ F \neg\varphi$, which means φ always holds in the future.

4.1. Modeling of Topic Security Vulnerability

4.1.1. Vulnerability Model

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

The model for stealing basic data of the topic is

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

- $S_1 = \{idle, check1, auth, comm, wait\_t, check2, success, fail\}$.
  There are eight states in S_1. In these states, *idle* denotes the initial state of the vulnerability model; *check*1 denotes the state to check whether 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 whether the intruder can get the base data of topic; *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; *get_topic* denotes to get the information of the target topic. *authorized*, *get_domain*, *get_topic* and *return_topic* are channels which 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.

 doi:10.20944/preprints202404.0143.v1

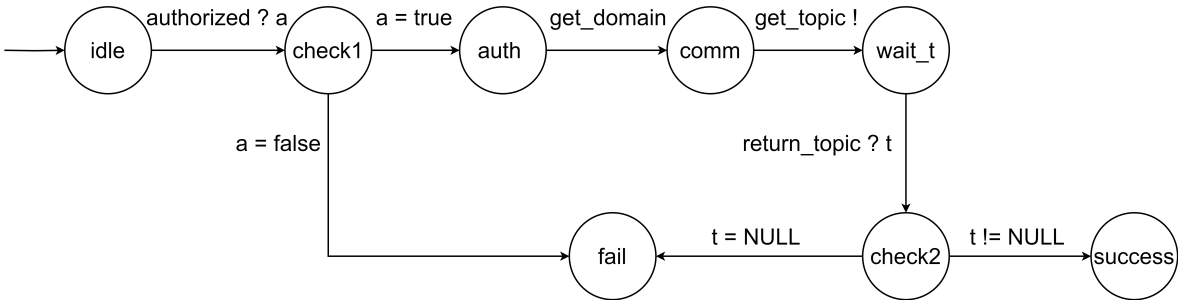

Figure 1. Stealing basic data of the topic.

The model for the unauthorized subscription is

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

- $S_2 = \{idle, check1, auth, comm, wait\_t, check2, sub, wait\_m, success, fail\}$.
  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 wait the topic which has been subscribed; *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 subscribe topic by channel *subscribe* and receives the topic by channel *publish*. If it can not receive message in a long time, it will get a timeout response by 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.

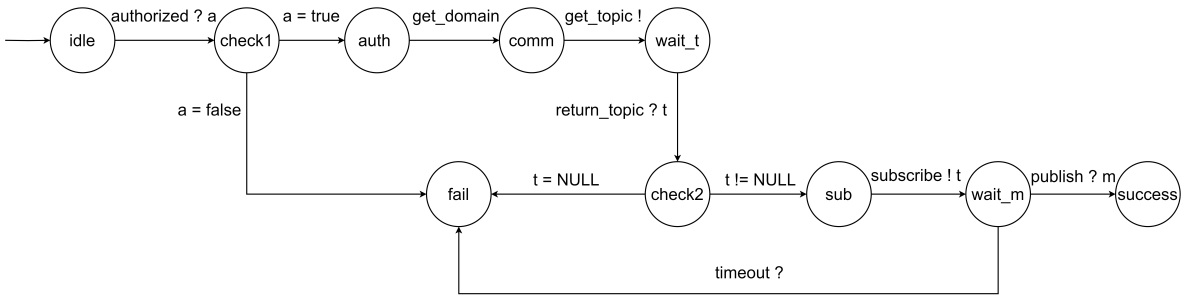

Figure 2. Unauthorized Subscription.

The model for unauthorized publication is

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

- $S_3 = \{idle, check1, auth, comm, wait\_t, check2, pub, wait\_m, success, fail\}$.
  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.

 doi:10.20944/preprints202404.0143.v1

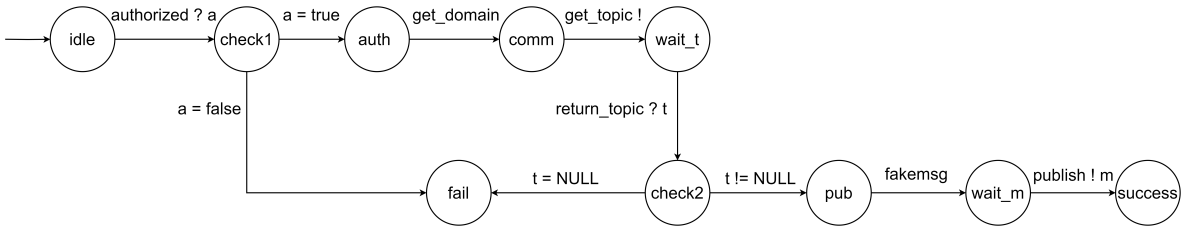

Figure 3. Unauthorized Publication.

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 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 into 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 II, we introduced two vulnerabilities in the ROS2 Service communication mechanism, stealing basic data of service and unauthorized service call. And in the following, we will model M_4, M_5 for each of these two vulnerabilities.

4.2.1. Vulnerability Model

The model for stealing basic data of service is

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

- $S_4 = \{idle, check1, auth, comm, wait\_s, check2, success, fail\}$.
  There are eight states in S_4. In these states, *idle* denotes the initial state of the vulnerability model;

 doi:10.20944/preprints202404.0143.v1

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 get the data of service by channel *return_service*.
- The initial state set $I_4 = \{idle\}$.
- The transition relationship between the states δ is shown in Figure 4.

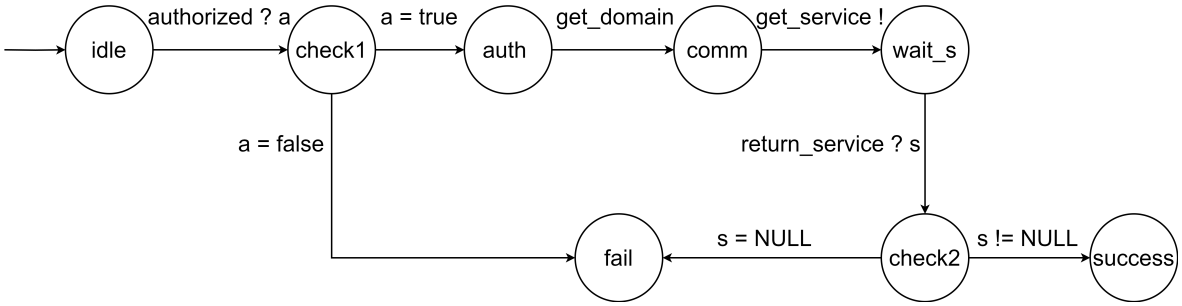

Figure 4. Stealing Basic data of service.

The model for unauthorized service call is

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

- $S_5 = \{idle, check1, auth, comm, wait\_s, check2, service, call, success, fail\}$.
  There are ten states in S_3. In these states, *service* denotes that the intruder has obtained the data of service; *call* denotes that intruder has sent a service request to 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.

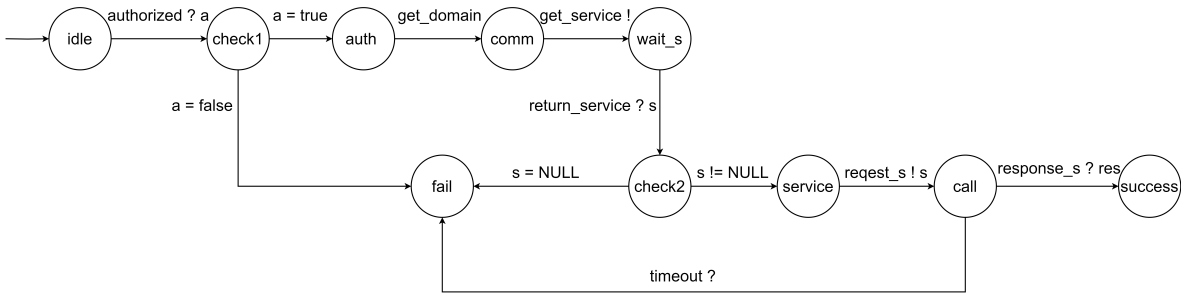

Figure 5. Unauthorized Service.

4.2.2. Security Specification

Service confidentiality in ROS2 requires that only authorized nodes have access to data related to the service.

 doi:10.20944/preprints202404.0143.v1

- 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 II, we introduced two vulnerabilities in the ROS2 Action communication mechanism, stealing basic data of action and unauthorized action call. We will model M_6, 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

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

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

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

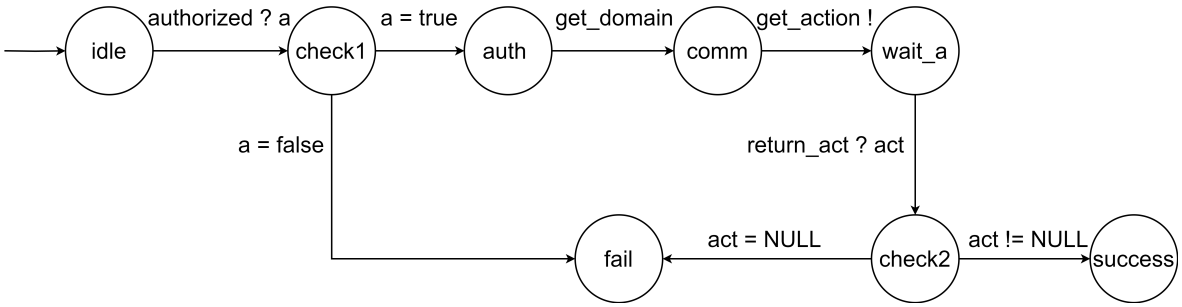

Figure 6. stealing basic data of action.

The model for unauthorized action call is

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

- $S_7 = \{idle, check1, auth, comm, wait\_a, check2, action, call, success, fail\}$.
  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; *success* and $fail$ represent whether or not the response was successfully received, respectively. And the other states is the same as 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.

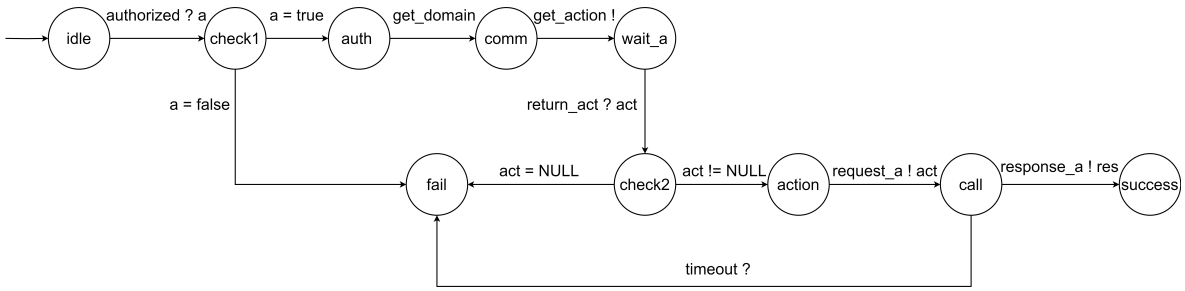

Figure 7. unauthorized action call.

4.3.2. Security Specification

ROS2 action confidentiality requires that only authorized nodes have access to data related to the action.

- 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:

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

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

Thus, the action confidentiality can be expressed as:

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

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 follow:

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

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

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

5. ROS2 Communication Security Vulnerability Detection Method

5.1. Framework of Method

In order to test and analyze the impact of the above attacks on ROS2 applications, we have designed a vulnerability detection method and tool for ROS2 in which these attacks have been implemented and integrated. Our method is designed to detect potential security vulnerabilities in ROS2 systems by conducting penetration testing [35]. This method comprises two modules: the communication domain scanning module and the vulnerability detection module, as shown in Figure 8.

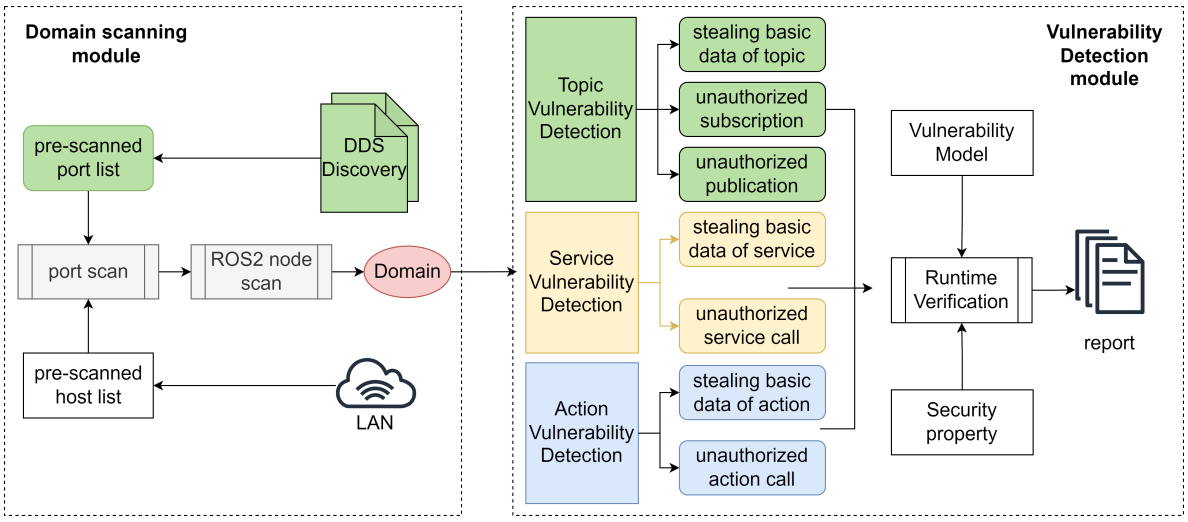

Figure 8. Framework of Method.

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 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.

 doi:10.20944/preprints202404.0143.v1

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 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.

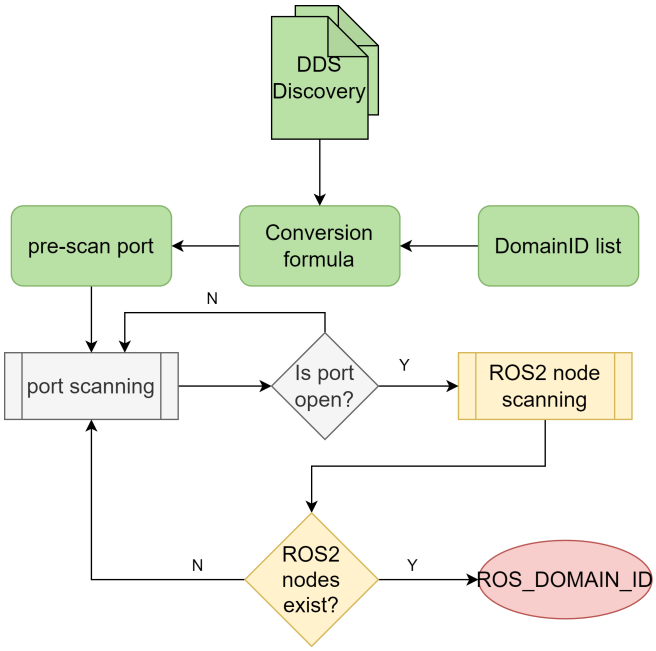

Figure 9. Domain Scanning Module.

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 [36] to scan the ports sequentially in the pre-scanned port list. If the port is active, it conducts a node scan to confirm whether an ROS2 node is running on it. If an 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: topic vulnerability detection module, service vulnerability detection module and action vulnerability detection module.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

5.3.1. Topic Vulnerability Detection Module

In the Topic Vulnerability Detection Module, the tool mainly carries out the three attacks of stealing basic data of the topic, unauthorized subscription and unauthorized publication to detect vulnerabilities on 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.

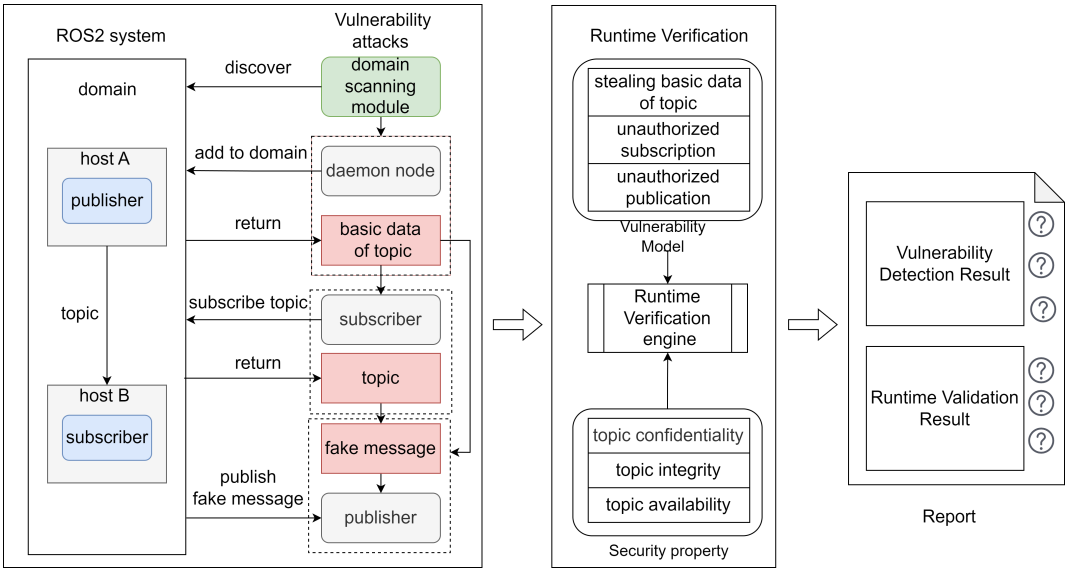

Figure 10. Topic Vulnerability Detection Module.

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 join it into the communication of that system. After successfully joining into the communication, 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 get 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 behaviour and state changes of the intruder and the system at the same time. Then these information will be corresponded to the vulnerability models. Based on this, it 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.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

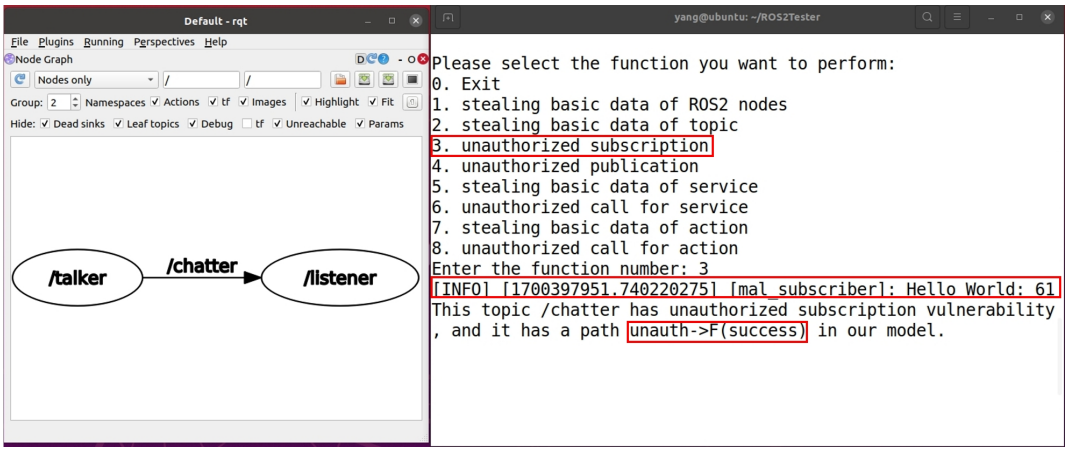

Figure 11. Topic Vulnerability Detection Module.

5.3.2. Service Vulnerability Detection Module

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

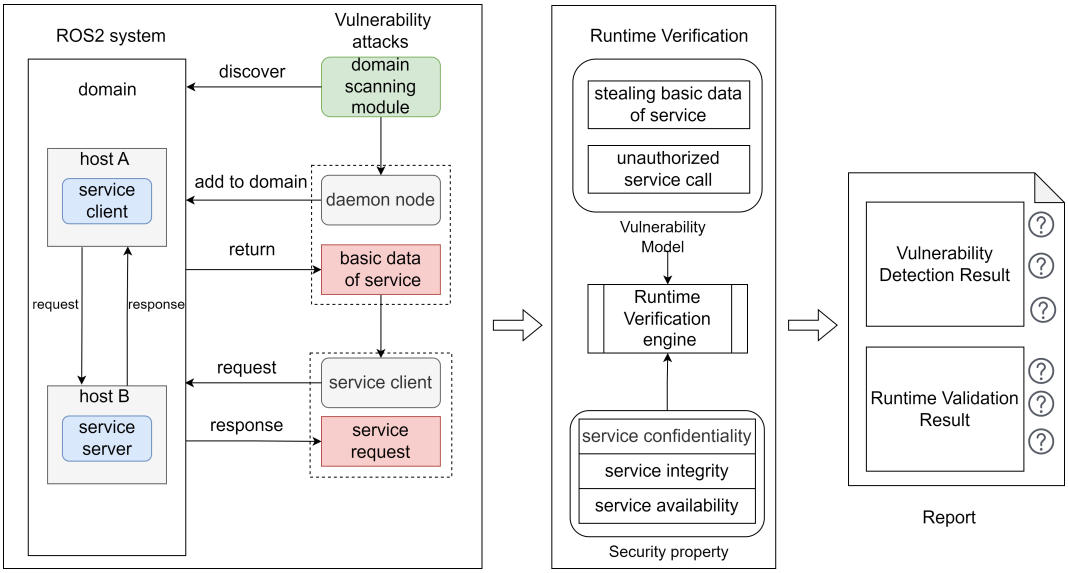

Figure 12. Service Vulnerability Detection Module.

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.

The same as the topic vulnerability detection module, in the entire vulnerability detection process, the tool will record the intruder and the system at the time of the behaviour and state changes. And then the stealing basic data of service vulnerability model and unauthorized service call model will be verified. Finally, the results of verification will be written into the detection report.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

5.3.3. Action Vulnerability Detection Module

In the action vulnerability detection module, we mainly detect vulnerabilities in the target system by two kinds of attacks, namely, stealing basic data of action and unauthorized action call, and if the attack is successful, the corresponding vulnerability exists in the target system. The implementation flow of this module is shown in Figure 13.

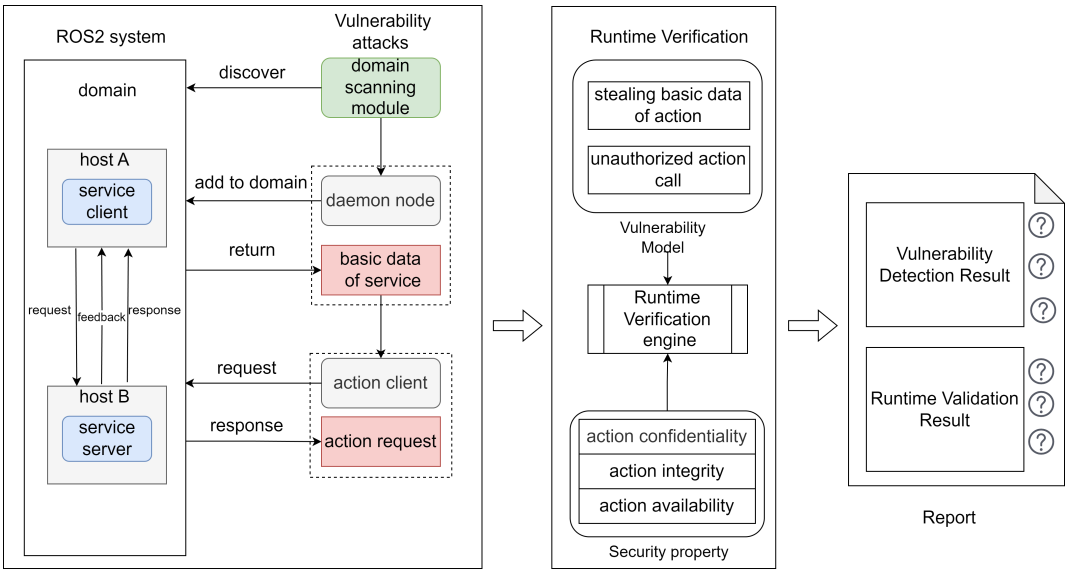

Figure 13. Action Vulnerability Detection Module.

The basic data of an action includes 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 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 time of the behaviour and state changes. And then the stealing basic data of action vulnerability model and unauthorized action call model will be verified. Finally, the results of 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 the entire ROS2 application in order to find the parts of it that could be attacked.

6. Experiment

6.1. Experiment Environment

In order to test the usability of the tool and the communication security of the ROS2 system, we use 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. The 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 the host A and B components.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 April 2024 doi:10.20944/preprints202404.0143.v1

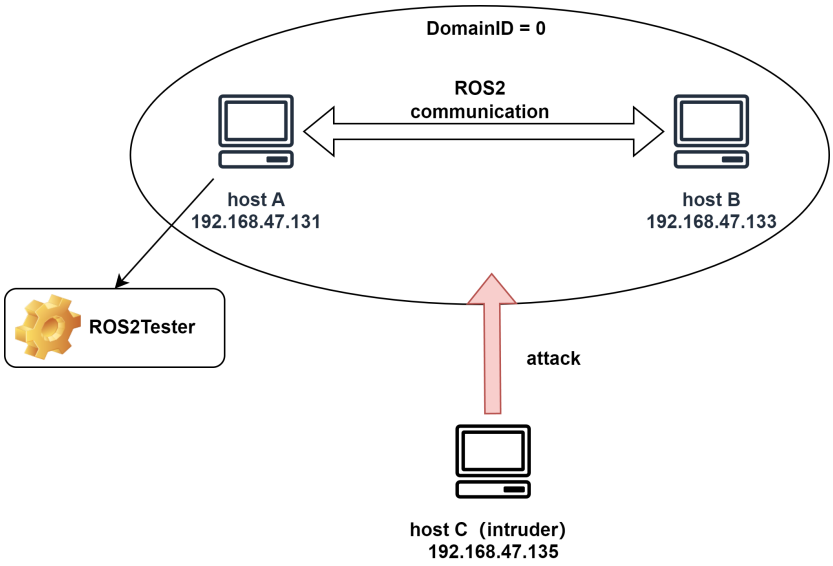

Figure 14. experimental environment

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.

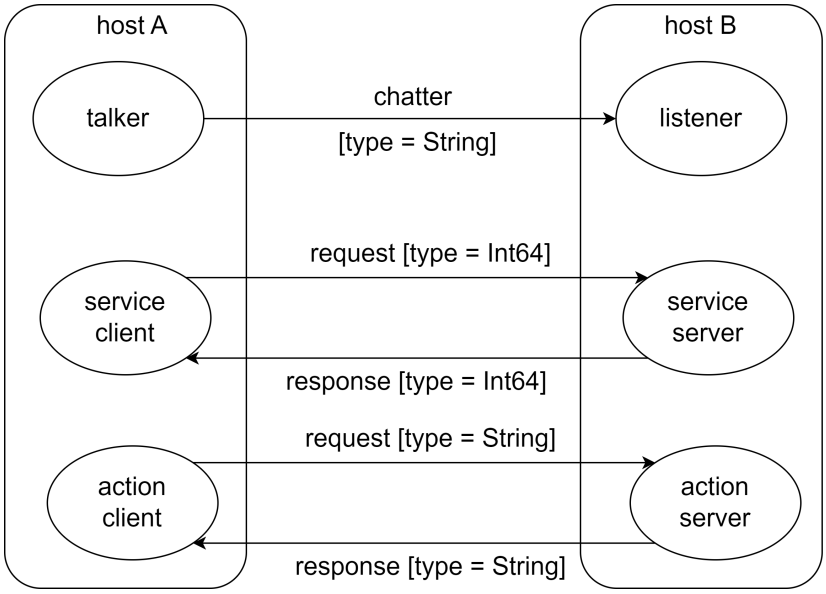

Figure 15. ROS2 system communication structure

In this system, topic communication is carried out between 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 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 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 analyse the security of SROS2, we can

 doi:10.20944/preprints202404.0143.v1

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 Result

In this experiment, we performed 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 Result

security property / type of vulnerability	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 stealing 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 basic data of the service and unauthorized service call 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 basic data of 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. Analysed in principle, the attacker's theft of some basic information may only lead to the risk of data leakage, and is unable to inject malicious data into the system or affect the normal communication between the system nodes. Therefore, these vulnerabilities can't damage the integrity and availability of the system.

Similarly, for service communication and action communication, the theft of their basic information cannot serve the purpose of data injection and disruption of system communication. and therefore cannot compromise the integrity and availability of the system. For unauthorized service call and unauthorized action call, 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 capable of compromising all of the three security properties.

6.2.2. Vulnerability Detection Result

In this experiment, we performed vulnerability testing of 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 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.

 doi:10.20944/preprints202404.0143.v1

Table 2. Detection Result

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 authorised 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 unauthorised nodes, and cannot withstand vulnerability attacks from authorised 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

To ensure that the vulnerability detection time for ROS2-based applications with a large number of nodes is acceptable, we tested ROS 2-based applications with a large number of nodes. As shown in Table 3, 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 minutes. 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.

Table 3. Tool Performance

time(s) number of the nodes / type of attack	100	500	1000
stealing basic data of the topic	0.0046	0.0128	0.0228
unauthorized subscription	0.0517	0.2547	0.5095
unauthorized publication	0.0524	0.2631	0.5344
stealing basic data of service	0.0269	0.0409	0.0759
unauthorized service call	66.9664	320.4411	750.5637
stealing basic data of action	0.5106	2.2425	5.8425
unauthorized action call	72.1933	369.9608	854.5155

 doi:10.20944/preprints202404.0143.v1

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 compare ROS2Tester with ROSPenTo [23], and ROSploit [24].

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 4 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 4. comparison of tool detection vulnerability coverage

vulnerability type \ tool name	**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 4, 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 centralised 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 decentralised. Therefore it is also difficult for intruders to achieve the modification of node parameters.

7. Conclusion

In this paper, we formally modelled common communication security vulnerabilities in ROS2 applications and use 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 the reachability analysis, by 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.

 doi:10.20944/preprints202404.0143.v1

ROS2 is still in rapid development, but there is a lack of tools for security test. The work in this paper has explored some of these aspects, which are very favorable to enhancing the security of ROS2. In the future, 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.

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.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

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

References

1. Jones, J.L. Robots at the tipping point: the road to iRobot Roomba. *IEEE Robotics & Automation Magazine* **2006**, *13*, 76–78.
2. Wang, Z.; Tian, G.; Shao, X. Home service robot task planning using semantic knowledge and probabilistic inference. *Knowledge-Based Systems* **2020**, *204*, 106174.
3. Ji, Z.; Qiu, R.; Noyvirt, A.; Soroka, A.; Packianather, M.; Setchi, R.; Li, D.; Xu, S. Towards automated task planning for service robots using semantic knowledge representation. IEEE 10th International Conference on Industrial Informatics. IEEE, 2012, pp. 1194–1201.
4. Freschi, C.; Ferrari, V.; Melfi, F.; Ferrari, M.; Mosca, F.; Cuschieri, A. Technical review of the da Vinci surgical telemanipulator. *The International Journal of Medical Robotics and Computer Assisted Surgery* **2013**, *9*, 396–406.
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. 2014 IEEE international conference on robotics and automation (ICRA). IEEE, 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. *Journal of Intelligent & Robotic Systems* **2015**, *80*, 15–31.
7. Luo, R.C.; Hsu, T.Y.; Lin, T.Y.; Su, K. The development of intelligent home security robot. IEEE International Conference on Mechatronics, 2005. ICM'05. IEEE, 2005, pp. 422–427.
8. Luo, R.C.; Chou, Y.T.; Liao, C.T.; Lai, C.C.; Tsai, A.C. NCCU security warrior: An intelligent security robot system. IECON 2007-33rd Annual Conference of the IEEE Industrial Electronics Society. IEEE, 2007, pp. 2960–2965.
9. Plósz, S.; Schmittner, C.; Varga, P. Combining safety and security analysis for industrial collaborative automation systems. Computer Safety, Reliability, and Security: SAFECOMP 2017 Workshops, ASSURE, DECSoS, SASSUR, TELERISE, and TIPS, Trento, Italy, September 12, 2017, Proceedings 36. Springer, 2017, pp. 187–198.
10. Kirschgens, L.A.; Ugarte, I.Z.; Uriarte, E.G.; Rosas, A.M.; Vilches, V.M. Robot hazards: from safety to security. *arXiv preprint arXiv:1806.06681* **2018**.
11. Lacava, G.; Marotta, A.; Martinelli, F.; Saracino, A.; La Marra, A.; Gil-Uriarte, E.; Vilches, V.M. Cybsersecurity Issues in Robotics. *J. Wirel. Mob. Networks Ubiquitous Comput. Dependable Appl.* **2021**, *12*, 1–28.
12. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y.; others. ROS: an open-source Robot Operating System. ICRA workshop on open source software. Kobe, Japan, 2009, Vol. 3, p. 5.
13. DEĞİRMENCİ, E.; KIRCA, Y.S.; YOLAÇAN, E.N.; YAZİCİ, A. An Analysis of DoS Attack on Robot Operating System. *Gazi University Journal of Science* **2023**, pp. 1–1.
14. Bonaci, T.; Chizeck, H.J. On potential security threats against rescue robotic systems. 2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2012, pp. 1–2.
15. Vuong, T.; Filippoupolitis, A.; Loukas, G.; Gan, D. Physical indicators of cyber attacks against a rescue robot. 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS). IEEE, 2014, pp. 338–343.

 doi:10.20944/preprints202404.0143.v1

16. Denning, T.; Matuszek, C.; Koscher, K.; Smith, J.R.; Kohno, T. A spotlight on security and privacy risks with future household robots: attacks and lessons. Proceedings of the 11th international conference on Ubiquitous computing, 2009, pp. 105–114.
17. Adam, N. Workshop on future directions in cyber-physical systems security. Report on workshop organized by Department of Homeland Security (DHS), 2010.
18. Coble, K.; Wang, W.; Chu, B.; Li, Z. Secure software attestation for military telesurgical robot systems. 2010-MILCOM 2010 MILITARY COMMUNICATIONS CONFERENCE. IEEE, 2010, pp. 965–970.
19. Javaid, A.Y.; Sun, W.; Devabhaktuni, V.K.; Alam, M. Cyber security threat analysis and modeling of an unmanned aerial vehicle system. 2012 ieee conference on technologies for homeland security (hst). IEEE, 2012, pp. 585–590.
20. Groza, B.; Dragomir, T.L. Using a cryptographic authentication protocol for the secure control of a robot over TCP/IP. 2008 IEEE International Conference on Automation, Quality and Testing, Robotics. IEEE, 2008, Vol. 1, pp. 184–189.
21. Lee, G.S.; Thuraisingham, B. Cyberphysical systems security applied to telesurgical robotics. *Computer Standards & Interfaces* **2012**, *34*, 225–229.
22. Arkin, B.; Stender, S.; McGraw, G. Software penetration testing. *IEEE Security & Privacy* **2005**, *3*, 84–87.
23. 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 (Volume 4)*; Koubaa, A., Ed.; Studies in Computational Intelligence, Springer International Publishing: Cham, 2020; pp. 183–225.
24. Rivera, S.; Lagraa, S.; State, R. ROSploit: Cybersecurity Tool for ROS. 2019 Third IEEE International Conference on Robotic Computing (IRC), 2019, pp. 415–416.
25. Dieber, B.; Breiling, B.; Taurer, S.; Kacianka, S.; Rass, S.; Schartner, P. Security for the robot operating system. *Robotics and Autonomous Systems* **2017**, *98*, 192–203.
26. Halder, R.; Proença, J.; Macedo, N.; Santos, A. Formal Verification of ROS-Based Robotic Applications Using Timed-Automata. 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), 2017, pp. 44–50.
27. Huang, J.; Erdogan, C.; Zhang, Y.; Moore, B.; Luo, Q.; Sundaresan, A.; Rosu, G. ROSRV: Runtime Verification for Robots. Runtime Verification; Bonakdarpour, B.; Smolka, S.A., Eds.; Springer International Publishing: Cham, 2014; Lecture Notes in Computer Science, pp. 247–254.
28. Breiling, B.; Dieber, B.; Schartner, P. Secure communication for the robot operating system. 2017 Annual IEEE International Systems Conference (SysCon), 2017, pp. 1–6. ISSN: 2472-9647.
29. Rivera, S.; State, R. Securing Robots: An Integrated Approach for Security Challenges and Monitoring for the Robotic Operating System (ROS). 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), 2021, pp. 754–759. ISSN: 1573-0077.
30. Sundaresan, A.; Gerard, L.; Kim, M. Secure ROS.
31. Mayoral-Vilches, V.; White, R.; Caiazza, G.; Arguedas, M. Sros2: Usable cyber security tools for ros 2. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2022, pp. 11253–11259.
32. Kim, J.; Smereka, J.M.; Cheung, C.; Nepal, S.; Grobler, M. Security and Performance Considerations in ROS 2: A Balancing Act **2018**. arXiv:1809.09566 [cs].
33. Maruyama, Y.; Kato, S.; Azumi, T. Exploring the performance of ROS2. Proceedings of the 13th International Conference on Embedded Software, 2016, pp. 1–10.
34. Deng, G.; Xu, G.; Zhou, Y.; Zhang, T.; Liu, Y. On the (In) Security of Secure ROS2. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, 2022, pp. 739–753.
35. Bacudio, A.G.; Yuan, X.; Chu, B.T.B.; Jones, M. An overview of penetration testing. *International Journal of Network Security & Its Applications* **2011**, *3*, 19.
36. Orebaugh, A.; Pinkard, B. *Nmap in the enterprise: your guide to network scanning*; Elsevier, 2011.