The Security Analysis of ROS2 Communication

Shuo Yang[1], Hongru Li[2], and Jian Guo[3(✉)]

[1] MoE Engineering Research Center for Software/Hardware Co-design Technology and Application, ECNU, Shanghai, China
51215902140@stu.ecnu.edu.cn
[2] Shanghai Trusted Industry Internet Software Collaborative Innovation Center, ECNU, Shanghai, China
51215902160@stu.ecnu.edu.cn
[3] Xinjiang Teacher's College and National Trusted Embedded Software Engineering Technology Research Center, ECNU, Shanghai, China
jguo@sei.ecnu.edu.cn

Abstract. With the increasing use of robots in various fields, the importance of communication security between robots and their components has become a pressing concern. As the primary development framework for robot applications, ROS2 is replacing ROS1 at a rapid pace, and its security issues have direct implications for the security of robot systems. This paper presents an exploration and study of the communication security issues of ROS2 by combining CIA triad with the ROS2 communication mechanism. We propose the fundamental security requirements of the ROS2 system under different communication mechanisms and provide formal modeling and definition. Moreover, we classify and analyze network attacks at the ROS2 level and implement a tool, ROS2Tester, to conduct modeling the ROS2 formal security modules, penetration testing and evaluating the security of ROS2 systems.

Keywords: ROS2 · communication security · CIA · formal model

1 Introduction

In recent years, the rise of robotics has led to their widespread adoption in various fields [1], including manufacturing, logistics, healthcare, and smart homes. As robots become more intelligent and autonomous, their ability to interact and communicate with the external world has accelerated, making robot security a crucial concern. Any failure within the robotic networks can result in the malfunctioning of the entire robotic system, thereby posing a severe threat to human safety and environmental security [2–4].

This paper is funded by Shanghai Trusted Industry Internet Software Collaborative Innovation Center.

H. Duan et al. (Eds.): SecureComm 2023, LNICST 568, pp. 122–139, 2025.
https://doi.org/10.1007/978-3-031-64954-7_7

In the Robot Operating System (ROS) [5], 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 [6].

Same as ROS1, there is no unified standard for security measurement in ROS2, so this paper uses CIA triad which is common in information security for security analysis and proposes the basic requirements that ROS2 system should meet. In addition, since there is no testing tool similar to ROSPenTo for ROS2, this paper implements potential network attacks in the tool ROS2Tester, and tests and analyzes some ROS2 applications through this tool. The main work of this paper is as follows:

- Based on the CIA triad, the basic security requirements for the ROS2 communication mechanism from the three perspectives of confidentiality, integrity and availability are proposed and formally modeled and defined. For ROS2, topic, service and action must all satisfy these three security goals to ensure the communication security;
- Based on the proposed security requirements, we conducted a security analysis of the communication mechanism in the ROS2 system, and identified potential security threats and vulnerabilities. Furthermore, we developed a vulnerability detection tool, ROS2Tester, using these potential security threats to perform vulnerability analysis and detection on ROS2 applications. Through our experiments, we found that there are still security threats to ROS2 and its components that could compromise its confidentiality, availability and integrity.

The remainder of this paper is structured as follows. Section 2 presents an overview of the basic architecture and communication mechanisms of ROS2. In Sect. 3, we formally define the communication security requirements that ROS2 must satisfy based on the CIA triad. Section 4 provides a comprehensive overview of the current potential security threats and vulnerabilities of ROS2 and categorizes them based on the established security requirements. Section 5 introduces ROS2Tester, the vulnerability detection tool developed in our work, and discusses its design and implementation. In Sect. 6, we set up an implementation environment for ROS2 security testing and evaluate it with ROS2Tester. Finally, in Sect. 7, we summarize our findings and suggest potential avenues for future research.

2 Related Work

In order to better analyze the communication security of ROS/ROS2 for the purpose of improving its security, many researchers have explored this area. A part of researchers focus on the potential vulnerabilities of the system and how

to detect them, while others focus on protection schemes to improve the security of the system. ROSPenTo [7] and ROSploit [8], for example, are classic tools for security detection of ROS-based applications. The researchers conducted penetration testing of ROS-based applications by implementing ROSPenTo using XML-RPC API, showing that ROS has vulnerabilities such as unauthorized publication. At the same time, they analyzed the security of ROS based on this finding and proposed light-Weight precautions to harden ROS at application-level [9]. ROSploit also tested ROS security based on penetration testing principles, and based on the test results, the researchers showed that ROS has security problems, and later proposed ROS_Immunity [10] as a protection mechanism to increase the security of ROS. ROS_Immunity integrates internal system defense, external system authentication, and automated vulnerabilities. System authentication, and automated vulnerability detection with Secure-ROS [11] to provide a set of defenses for ROS systems against malicious attackers. In addition, to enhance the security of ROS, the main schemes currently used are encryption and authentication mechanisms [11–13]. Differential simulation testing [14], formal verification [15,16] and runtime verification [17–19] and other methods are also used to check the reliability and security of ROS.

Compared with ROS1, ROS2, a new generation of robot operating system, has undergone tremendous security enhancements. ROS2 uses DDS Security [20] as a security enhancement module with strong authentication and permission management mechanisms, and uses SROS2 [21] to enhance its usability. However, ROS2 is not absolutely secure, because the security problems existing in ROS1 ROS2 may not be able to solve all, and ROS2 itself may also have new security problems. Some early studies [22,23] have conducted a comprehensive analysis of the security and performance of ROS2 and SROS2, trying to find a balance between the two. Meanwhile, other studies [22,24] show that SROS2 still has flaws at present by analyzing the security analysis of DDS Security or SROS2. The researchers [25] stated the limitations and future work of the ROS2-based application in forensic investigation.

3 ROS2 Communication Security Requirements

3.1 CIA Triad

The CIA triad [26] is a crucial standard for computer system security, widely utilized in designing and assessing information security measures and systems to determine their adherence to fundamental information security requirements [27–29]. Within CIA, confidentiality, integrity, and availability, as three core elements, interact with one another, and the system and its data can only be secured if all three properties are satisfied concurrently.

To precisely describe and apply the CIA triad in information security systems, we provide a formal definition of a basic secure communication system S.

$$S = \{C, M, A\} \tag{1}$$

where,

- C: The set of communication components used to send or receive messages.
- M: The set of messages in communication.
- A: The set of actions, containing the following actions:
  - $SRC \subseteq M \times C$: $\forall(m, s) \in SRC$, the component $s \in C$ is the expected sender of the message $m \in M$. For convenience of presentation, we define $expected\_src(m, s) \equiv (m, s) \in SRC$.
  - $RCV \subseteq M \times C$: $\forall(m, r) \in RCV$, the component $r \in C$ is the expected recipient of the message $m \in M$. For convenience of presentation, we define $expected\_rcv(m, r) \equiv (m, r) \in RCV$.
  - $INJECT \subseteq M \times C$: $\forall(m, c) \in INJECT$, the component $c \in C$ sends the message $m \in M$ into the system. For convenience of presentation, we define $inject(m, c) \equiv (m, c) \in INJECT$.
  - $INTERCEPT \subseteq M \times C$: $\forall(m, c) \in INTERCEPT$, component $c \in C$ receives message $m \in M$ from the system. For convenience of presentation, we define $intercept(m, c) \equiv (m, c) \in INTERCEPT$.
- In addition, the following formal expressions are defined in this paper:
  - $E(a), a \in INJECT \cup INTERCEPT$: the action a has occurred before.
  - $F(a), a \in INJECT \cup INTERCEPT$: the action a will finally occur.

In accordance with the definition provided by the CIA triad, a communication system can be considered secure only if it satisfies the three fundamental attributes of confidentiality, integrity, and availability. The detailed description and formal definition of each attribute are presented as follows:

Confidentiality: Confidentiality refers to the ability to prevent unauthorized individuals or systems from accessing sensitive information, ensuring that only authorized users or systems can access such information. In other words, the message m can only be received and accessed by its intended recipient, which is formally defined as follows:

$$E(intercept(m, c)) \Rightarrow expected\_rcv(m, c) \tag{2}$$

Integrity: Integrity refers to the capacity to prevent unauthorized modifications or corruption of data in systems, thus guaranteeing the preservation of their original state and value.

$$E(inject(m, c)) \Rightarrow expected\_src(m, c) \tag{3}$$

Availability: Availability is the ability to ensure that a system is available when needed and remains in proper operating condition.

$$inject(m, c_1) \wedge expected\_rcv(m, c_2) \Rightarrow F(E(intercept(m, c_2)) \tag{4}$$

3.2 CIA of ROS2

All three communication mechanisms of ROS2 can be utilized by the robotic systems for communication and data transmission, and each of them may become the target of attacker's intrusion if not adequately protected. Therefore, for the entire robotic system based on ROS2 to ensure security, each communication mechanism must satisfy confidentiality, integrity, and availability. Given the differences in the communication components and message types of the three communications, we define each communication model of ROS2 based on the basic communication system model described above, as follows:

$$S_{ROS2} = \{C, M, A\} \tag{5}$$

- $C = pub \cup sub \cup srv\_client \cup srv\_server \cup act\_client \cup act\_server$:
  - $c \in pub$ represent c is a publisher in the topic communication.
  - $c \in sub$ represent c is a subscriber in the topic communication.
  - $c \in srv\_client$ represent c is a client in the service communication.
  - $c \in srv\_server$ represent c is a server in the service communication.
  - $c \in act\_client$ represent c is a client in the action communication.
  - $c \in act\_server$ represent c is a server in the action communication.
- $M = topic \cup req \cup res$:
  - $m \in topic$ represents a topic in the topic communication.
  - $m \in req$ represents a request in the service or action communication.
  - $m \in res$ represents a response in the service or action communication.
- $A = SRC \cup RCV \cup INJECT \cup INTERCEPT$, which has been described clearly in Sect. 3.1.

Security Requirements of Topic. The topic mechanism in ROS2 consists of two types of entities: subscriber and publisher. During communication, sensitive information such as the names of publisher and subscriber, topic names and types, and message data must be protected against theft or unauthorized modification.

Publish and subscribe are defined as follows:

- $pub(m, c) \equiv inject(m, c) \wedge c \in pub \wedge m \in topic$
- $sub(m, c) \equiv intercept(m, c) \wedge c \in sub \wedge m \in topic$

Based on the above analysis, the CIA attributes to be satisfied under the ROS2 topic mechanism are defined in this paper as follows:

Confidentiality:

$$E(sub(m, c)) \Rightarrow expected\_rcv(m, c) \tag{6}$$

Integrity:

$$E(pub(m, c)) \Rightarrow expected\_src(m, c) \tag{7}$$

Availability:

$$pub(m, c_1) \wedge expected\_rcv(m, c_2) \Rightarrow F(E(sub(m, c_2))) \tag{8}$$

Security Requirements of Service. Within ROS2's service communication mechanism, there are two distinct communication entities: servers and clients. These entities operate on a request-response model. During service communication, both the request and response messages may contain sensitive data such as service type and parameters or critical calculation results, respectively. To ensure confidentiality in CIA, it is crucial to prevent unauthorized access to this data.

The service request and service response are defined as follows:

- $send\_srv\_req(m, c) \equiv inject(m, c) \wedge c \in srv\_client \wedge m \in req$
- $rcv\_srv\_req(m, c) \equiv intercept(m, c) \wedge c \in srv\_server \wedge m \in req$
- $send\_srv\_res(m, c) \equiv inject(m, c) \wedge c \in srv\_server \wedge m \in res$
- $rcv\_srv\_res(m, c) \equiv intercept(m, c) \wedge c \in srv\_client \wedge m \in res$

Based on the above analysis, the CIA security attributes to be satisfied under the ROS2 service mechanism are defined in this paper as follows:

$Confidentiality$:
$(E(rcv\_srv\_res(m, c_1)) \Rightarrow expected\_rcv(m, c_1)) \wedge$
$(E(rcv\_srv\_req(m, c_2)) \Rightarrow expected\_rcv(m, c_2))$
$Integrity$:
$(E(send\_srv\_res(m, c_1)) \Rightarrow expected\_src(m, c_1)) \wedge$
$(E(send\_srv\_req(m, c_2)) \Rightarrow expected\_src(m, c_2))$
$Availability$:
$(send\_srv\_req(m, c_1) \wedge expected\_rcv(m, c_2) \Rightarrow F(E(rcv\_srv\_req(m, c_2))) \wedge$
$(send\_srv\_res(m, c_2) \wedge expected\_rcv(m, c_1) \Rightarrow F(E(rcv\_srv\_res(m, c_1)))$

Security Requirements of Action. The action mechanism in ROS2 is a hybrid of topic and service. Within the communication framework, both actions and services utilize the request-response model for one-to-many communication. However, unlike services, which receive and process requests and provide responses, action servers also release feedback messages continuously during request processing. Clients can subscribe to these messages to monitor the progress of request processing in real-time. As such, this communication mechanism may include significant data within its feedback, in addition to requests and responses.

The action request and action response are defined as follows:

- $send\_act\_req(m, c) \equiv inject(m, c) \wedge c \in act\_client \wedge m \in req$
- $rcv\_act\_req(m, c) \equiv intercept(m, c) \wedge c \in act\_server \wedge m \in req$
- $send\_act\_res(m, c) \equiv inject(m, c) \wedge c \in act\_server \wedge m \in res$
- $rcv\_act\_res(m, c) \equiv intercept(m, c) \wedge c \in act\_client \wedge m \in res$

Based on the above analysis, the CIA security attributes to be satisfied under the ROS2 action mechanism are defined in this paper as follows:

$Confidentiality$:
$(E(rcv\_act\_res(m, c_1)) \Rightarrow expected\_rcv(m, c_1)) \wedge$
$(E(rcv\_act\_req(m, c_2)) \Rightarrow expected\_rcv(m, c_2))$

Integrity:
$(E(send\_act\_res(m, c_1)) \Rightarrow expected\_src(m, c_1))\wedge$
$(E(send\_act\_req(m, c_2)) \Rightarrow expected\_src(m, c_2))$
Availability:
$(send\_act\_req(m, c_1) \wedge expected\_rcv(m, c_2) \Rightarrow F(E(rcv\_act\_req(m, c_2)))\wedge$
$(send\_act\_res(m, c_2) \wedge expected\_rcv(m, c_1) \Rightarrow F(E(rcv\_act\_res(m, c_1)))$

4 ROS2 Security Attacks

4.1 Classification of Attacks

Previous studies on ROS1 security attacks have shown that attacks were often achieved by exploiting its communication mechanism, resulting in network attacks. For example, the unauthorized publication in ROSPenTo was used to achieve a Dos attack and damage system availability. Although ROS2 has improved its security compared to ROS1, it is not fully protected against all the attacks that existed in ROS1. Therefore, this paper discusses some of the attacks facing ROS1 while analyzing potential attacks on ROS2.

We collect and classify network attacks on the basis of the ROS2 communication security requirements presented in Sect. 3. Each attack's impact on system security is analyzed, and the attacks are categorized into three types: confidentiality attacks, integrity attacks, and availability attacks. Figure 1 shows the classification of attacks.

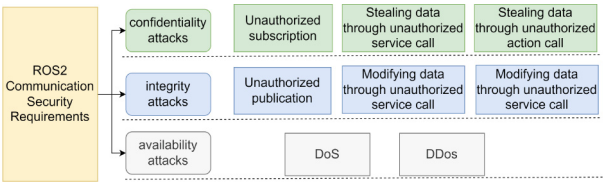

Fig. 1. classification of attacks

4.2 Confidentiality Attack

Confidentiality attacks in ROS2 focus on stealing sensitive data from the system, and we focus on unauthorized subscription, stealing data through unauthorized service call and stealing data through unauthorized action call.

- **Unauthorized subscription**. In the ROS2 topic communication mechanism, unauthorized access to message data is possible, as any node in ROS2 can subscribe to any topic without authorization. This vulnerability can be exploited by intruders to steal messages, potentially exposing sensitive system or user data. By creating a malicious node and impersonating a subscriber using obtained data related to the target topic, intruders can gain unauthorized access to topic data.

- **Stealing data through unauthorized service call**. In the ROS2 service communication mechanism, any ROS2 node can invoke any service through a request to access, modify data or invoke specific commands. This vulnerability can be exploited by an intruder to achieve data theft through invoking the relevant service. After obtaining the parameter information of the targeted service, the intruder can create a malicious node and establish a service client on it. The intruder can then invoke the service by sending a request through the created service client, based on the acquired parameter information.
- **Stealing data through unauthorized action call**. In the ROS2 action mechanism, any node in ROS2 can send a request to invoke an action and access its associated data. An intruder can exploit this mechanism to steal data by invoking the relevant action. After obtaining the parameter information for the target action, the intruder can create a malicious node and establish an action client on that node. Then, using the obtained parameter information, the action client sends a request to the server to invoke the action and steal the data returned by the action server.

4.3 Integrity Attacks

Attacks that can compromise the system's integrity are categorized as integrity attacks. Based on the communication mechanism used, such attacks are further classified as unauthorized publication, modifying data by unauthorized service call, and modifying data by unauthorized action call.

- **Unauthorized publication**. Unauthorized publication attacks refer to the malicious or false information dissemination by an attacker who impersonates a legitimate publisher node in a communication network. These attacks can cause erroneous data or commands to be injected into the network, resulting in disruptions to normal operations and potentially severe consequences.
- **Modifying data through unauthorized service call**. The service mechanism in ROS2 has three main functions: data acquisition, data modification, and command publication. Attackers can exploit these functions by making unauthorized service calls, which can result in the theft of data, compromise of ROS2's confidentiality, and injection of false commands or modified data, compromising the system's integrity. However, the impact of this attack may vary depending on the functionality provided by the called service.
- **Modifying data through unauthorized action call**. Similarly, the ROS2 action server also provides data acquisition, data modification, and command invocation functions, with the additional feature of providing feedback on the progress of executing the target request. Attackers can compromise the integrity of the system by injecting false commands or modifying original data in the network through unauthorized action invocation.

4.4 Availability Attack

To ensure the normal functioning of ROS2, it is essential to ensure the availability of data. Availability attacks, typically referred to as Denial-of-Service (DoS)

attacks in network security, can be easily achieved in ROS2 by publishing large amounts of false data. ROS2 nodes can set the publishing frequency of messages arbitrarily, making it vulnerable to unauthorized topic publishing attacks. An intruder can configure the publishing frequency to send a large amount of false or useless data to all nodes subscribed to the topic, resulting in a DoS attack.

In contrast, the service and action communication mechanisms of ROS2 utilize a request-response model where the client node can only send the next request after receiving a response from the server-side. This mechanism limits the frequency of sending requests to the speed of processing requests on the server-side, making it difficult to perform DoS attacks on the ROS2 system.

5 ROS2 Vulnerability Detection Tool

5.1 Framework of ROS2Tester

In order to test and analyze the impact of the above attacks on ROS2 applications, we specifically have developed a vulnerability detection tool for ROS2-based application in which these attacks have been implemented and integrated. ROS2Tester is designed to detect potential security vulnerabilities in ROS2 systems by conducting penetration testing [30]. The tool comprises two modules: communication domain scanning module and vulnerability detection module, as shown in Fig. 2.

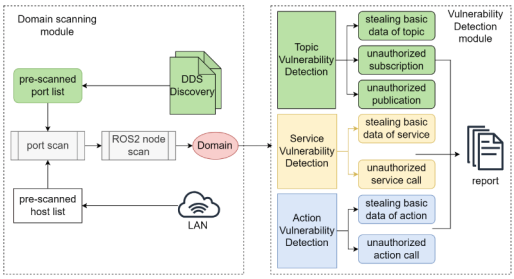

Fig. 2. ROS2Tester framework

The communication domain scanning module uses network port scanning to locate ROS2 systems running in the LAN. The vulnerability detection module scans for vulnerabilities according to different communication mechanisms in the system 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.

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:

$$DiscoveryMulticastPort = PB + DG * DomainID \tag{9}$$

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.

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 [31] 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 for the purpose of testing 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.

Topic Vulnerability Detection Module. This section describes the topic vulnerability detection module, which detects vulnerabilities in the target system using three attacks: stealing basic data of topic, unauthorized subscription, and unauthorized publication. The success of any of these attacks indicates the presence of vulnerabilities in the target system. The implementation flow of this module is illustrated in Fig. 3.

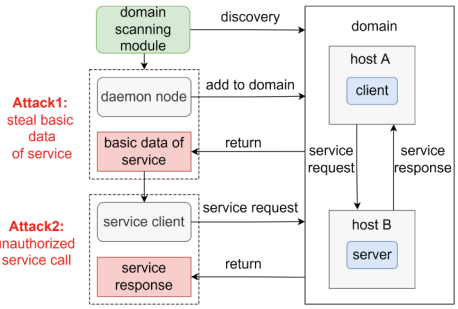

Fig. 3. Topic Vulnerability Detection

Basic data about topic includes topic name, topic type, and message types. The module accomplishes this by adding an intruder host to the communication domain of the attack target and then creating a daemon node to continuously obtain information about other nodes.

To implement unauthorized subscription attack, we first obtain a list of all node information through the daemon node, and then for each topic, we obtain its message type and create a subscriber node in turn. After running this node, the success of the attack is judged according to whether the returned message data is empty or not, as shown in Algorithm 1. Additionally, the module can also send fake data to the target topic by creating a fake publisher in a similarly way.

Algorithm 1: unauthorized subscription

Input: domainId
Output: success, msg
dNode ← create_daemon_node(domainId);
tList ← get_topics_list(dNode);
while *t is not the last topic in tList* **do**
    mType ← get_message_type(t.name,t.type);
    subNode ← create_subscriber(t,mType);
    msg ← spin(subNode);
    if *msg == NULL* **then**
        success ← false;
    else
        success ← true;
    end
    process_next_topic;
end
return msg, success;

Service Vulnerability Detection Module. In this module, we use two types of attacks to detect vulnerabilities in the target system: 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 Fig. 4.

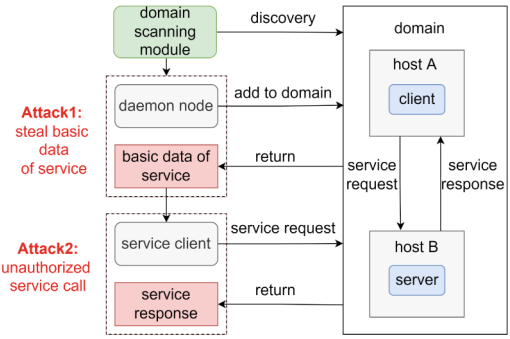

Fig. 4. Service Vulnerability Detection

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.

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 Fig. 5.

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.

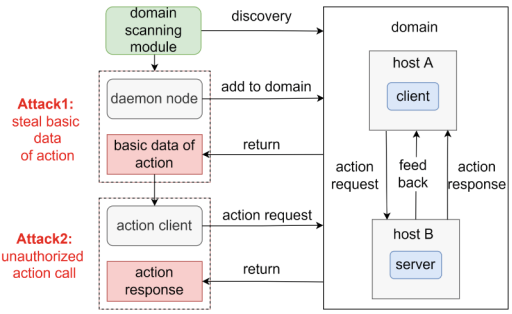

Fig. 5. Action Vulnerability Detection

6 Experiment

6.1 Experiment Environment

To conduct a communication security analysis of ROS2 and its security component SROS2, we set up an experimental environment consisting of three hosts in the same LAN. As shown in Fig. 6, experiments are conducted to detect vulnerabilities in three different communication mechanisms of ROS2, requiring the construction of three different experimental environments based on the communication mechanism used.

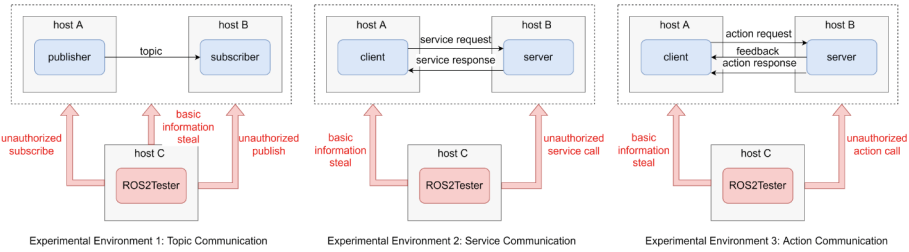

Fig. 6. Experimental Environment

For each experimental environment, we build simple and realistic ROS2 systems. For Environment 1, we set up a publisher node in Host A to publish integer data to the topic and a subscriber node in Host B to be responsible for subscribing to the data. Similarly, we built the service/action client and server in two hosts to build the corresponding ROS2 systems for environments 2 and 3, respectively.

Take experimental environment 1 as an example, to detect vulnerabilities and perform security analysis on the ROS2 topic mechanism, we create the publisher node *talker* on host A to publish data to *chatter* and the subscriber node

listener on host B to subscribe the data from *chatter*. Then run ROS2Tester on host C to conduct three basic attacks on the ROS2 system: base data stealing, unauthorized subscription, and unauthorized publication. The detailed information is shown in Fig. 7, where the top-left corner displays rqt running on Host A, which is used to visualize the running ROS2 system.

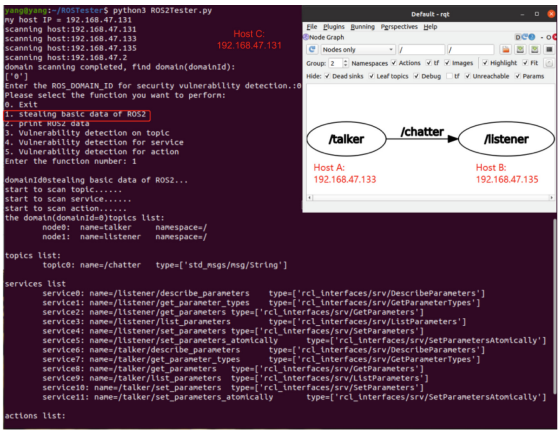

Fig. 7. The execution and result of ROS2Tester

Through the above three experimental environments, 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 use ROS2Tester to perform the same vulnerability detection and compare the security changes before and after using SROS2.

6.2 Results and Analysis

In our experiments, we conduct vulnerability detection for each of the three communication mechanisms in ROS2: topic, service, and action, and compare the security changes in ROS2 with SROS2 and ROS2 without SROS2.

By analyzing the generated vulnerability detection reports, we found that:

- Without SROS2, network attacks in the experiments successfully achieved theft and injection of communication data, as well as interference and disruption of the communication process for all three communication mechanisms.
- With SROS2, ROS2 can prevent the experimental host C from participating in ROS2 communication, thus preventing any attack on SROS2.

Our experiment result finds that SROS2 is generally reliable for securing ROS2, but we also discovered some security problems with SROS2 when

attempting to attack a ROS2 system protected by SROS2. Specifically, the key transmission mechanism and user authentication mechanism of SROS2 are imperfect, making it possible for attackers to intercept security files such as keys. If an attacker intercepts the key and other security files, they can use them to directly attack a protected node in the ROS2 system from an intruder node.

To comprehensively analyze the security of ROS2, we enhanced the attacks by exploiting the flaws of SROS2, allowing the attacker to use intercepted security files for authentication and authorization.

With the above additions to the experiments, the complete experimental results are presented in Table 1. A "✓" indicates that the attack was successfully executed and the tool detected the presence of the corresponding vulnerability in the system, while an "✕" means that the tool believes that the corresponding vulnerability does not exist in the system.

Table 1. result of experiment

attacker node	type of attack	ROS2	SROS2
unauthorized node	stealing basic data of node	✓	✕
	stealing basic data of topic	✓	✕
	unauthorized subscription	✓	✕
	unauthorized publication	✓	✕
	stealing basic data of service	✓	✕
	unauthorized call for service	✓	✕
	stealing basic data of action	✓	✕
	unauthorized call for action	✓	✕
authorized node	steal basic data of node	✓	✓
	stealing basic data of topic	✓	✓
	unauthorized subscription	✓	✓
	unauthorized publication	✓	✓
	stealing basic data of service	✓	✓
	unauthorized call for service	✓	✓
	stealing basic data of action	✓	✓
	unauthorized call for action	✓	✓

After conducting the above analysis, it is evident that there are still several cyber-attack threats to ROS2, making it challenging to meet the security requirements outlined in this paper based on the CIA triad. Taking topic communication as an example, if an unauthorized subscription attack occurs on the topic m in the ROS2 system, unauthorized subscribers s can subscribe to m, i.e., $E(sub(m, s)) = true$, and $expected\_rcv(m, s) = false$, without fulfilling the condition $E(sub(m, s)) \Rightarrow expected\_rcv(m, s)$. Hence, the system fails to satisfy the confidentiality requirement.

Table 2. Result of tool performance testing

type of attack	time(s)		
	100	500	1000
stealing basic data of node	0.0035	0.0057	0.0085
stealing basic data of 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 call for service	66.9664	320.4411	750.5637
stealing basic data of action	0.5106	2.2425	5.8425
unauthorized call for action	72.1933	369.9608	854.5155

To ensure that the vulnerability detection time for ROS 2-based applications with a large number of nodes is acceptable, we tested ROS 2-based applications with a large number of nodes. In traditional robotic systems, the number of nodes typically ranges from tens to hundreds. As shown in Table 2, 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. For example, the tool can quickly complete the task of "stealing basic data of node" for 1000 nodes in about 0.0085 s. However, the duration of the vulnerability scan for "unauthorized call for action" is affected by the processing time of the action communication server. In our tests, vulnerability scans for 1,000 actions that manage common tasks (e.g., keyboard input and movement direction control) 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.

7 Conclusion

In this paper, we propose basic security requirements for ROS2 communication based on the CIA triads, and analyze the corresponding cybersecurity attacks. We implement these attacks into the ROS2Tester to analyze the security issues in ROS2 applications under different communication mechanisms. Our experiments show that the communication network in ROS2 and SROS2 are vulnerable to cyber attacks such as data theft and data injection.

As ROS2 continues to evolve, there is a need for tools that can enhance the security testing of ROS2. This paper contributes to this effort by exploring some aspects of security testing in ROS2. However, our work still has some limitations. ROS2Tester is not applicable to ROS1-based applications and the

analysis of vulnerability results is not logical enough. In our future work, we will use formal verification techniques to model the CIA properties and ROS2 communication mechanisms to achieve a better analysis of the vulnerabilities.

References

1. Siriweera, A., Naruse, K.: Survey on cloud robotics architecture and model-driven reference architecture for decentralized multicloud heterogeneous-robotics platform. IEEE Access **9**, 40521–40539 (2021)
2. Plósz, S., Schmittner, C., Varga, P.: Combining safety and security analysis for industrial collaborative automation systems. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds.) SAFECOMP 2017. LNCS, vol. 10489, pp. 187–198. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66284-8_16
3. 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)
4. Lacava, G., et al.: Cybsersecurity issues in robotics. J. Wirel. Mob. Netw. Ubiquitous Comput. Dependable Appl. **12**(3), 1–28 (2021)
5. Quigley, M., et al.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software, Kobe, Japan, vol. 3, p. 5 (2009)
6. Degirmenci, E., Kirca, Y.S., Yolacan, E. N., Yazici, A.: An analysis of dos attack on robot operating system. Gazi Univ. J. Sci. 1 (2023)
7. Dieber, B., et al.: Penetration testing ROS. In: Koubaa, A. (ed.) Robot Operating System (ROS). SCI, vol. 831, pp. 183–225. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-20190-6_8
8. Rivera, S., Lagraa, S., State, R.: ROSploit: cybersecurity tool for ROS. In: 2019 Third IEEE International Conference on Robotic Computing (IRC), pp. 415–416 (2019). https://doi.org/10.1109/IRC.2019.00077
9. Dieber, B., Breiling, B., Taurer, S., Kacianka, S., Rass, S., Schartner, P.: Security for the robot operating system. Robot. Auton. Syst. **98**, 192–203 (2017)
10. Rivera, S., State, R.: Securing robots: an integrated approach for security challenges and monitoring for the robotic operating system (ROS). In: 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 754–759 (2021). ISSN 1573-0077
11. Sundaresan, A., Gerard, L., Kim, M.: Secure ROS (2017)
12. Breiling, B., Dieber, B., Schartner, P.: Secure communication for the robot operating system. In: 2017 Annual IEEE International Systems Conference (SysCon), pp. 1–6 (2017). https://doi.org/10.1109/SYSCON.2017.7934755. ISSN 2472-9647
13. White, R., Caiazza, G., Christensen, H., Cortesi, A.: SROS1: using and developing secure ROS1 systems. In: Koubaa, A. (ed.) Robot Operating System (ROS). SCI, vol. 778, pp. 373–405. Springer, Cham (2019). https://doi.org/10.1007/978-3-319-91590-6_11
14. Wang, Y., Wang, B., Guan, Y., Li, X., Wang, R.: Differential fuzz testing of robot operating system. Ruan Jian Xue Bao/J. Softw. **32**(6), 1867–1881 (2021). http://www.jos.org.cn/1000-9825/6254.htm
15. Halder, R., Proença, J., Macedo, N., Santos, A.: Formal verification of ROS-based robotic applications using timed-automata. In: 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE), pp. 44–50 (2017). https://doi.org/10.1109/FormaliSE.2017.9

16. Jian, L., Xiaojuan, L., Yong, G., Rui, W., Zhiping, S.: Modeling and analysis of ROS2 data distribution service for data flow. J. Softw. **32**(6), 1818–1829 (2021)
17. Huang, J., et al.: ROSRV: runtime verification for robots. In: Bonakdarpour, B., Smolka, S.A. (eds.) RV 2014. LNCS, vol. 8734, pp. 247–254. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11164-3_20
18. Ferrando, A., Cardoso, R.C., Fisher, M., Ancona, D., Franceschini, L., Mascardi, V.: ROSMonitoring: a runtime verification framework for ROS. In: Mohammad, A., Dong, X., Russo, M. (eds.) TAROS 2020. LNCS (LNAI), vol. 12228, pp. 387–399. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63486-5_40
19. Kirca, Y.S., et al.: Runtime verification for anomaly detection of robotic systems security. Machines **11**(2), 166 (2023). https://www.mdpi.com/2075-1702/11/2/166
20. (OMG) O.M.G.: OMG data distribution service (DDS), version 1.4. [Online] (2015). https://www.omg.org/spec/DDS/
21. Vilches, V.M., White, R., Caiazza, G., Arguedas, M.: SROS2: Usable Cyber Security Tools for ROS 2 (2022). http://arxiv.org/abs/2208.02615, [cs]
22. Kim, J., Smereka, J.M., Cheung, C., Nepal, S., Grobler, M.: Security and Performance Considerations in ROS 2: A Balancing Act (2018). http://arxiv.org/abs/1809.09566, [cs]
23. Maruyama, Y., Kato, S., Azumi, T.: Exploring the performance of ROS2. In: Proceedings of the 13th International Conference on Embedded Software, pp. 1–10 (2016)
24. 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, pp. 739–753 (2022)
25. Patel, Y., Rughani, P.H., Desai, D.: Analyzing security vulnerability and forensic investigation of ROS2: a case study. In: Proceedings of the 8th International Conference on Robotics and Artificial Intelligence, pp. 6–12. ACM (2021). https://doi.org/10.1145/3573910.3573912
26. Samonas, S., Coss, D.: The CIA strikes back: redefining confidentiality, integrity and availability in security. J. Inf. Syst. Secur. **10**(3) (2014)
27. Nasiri, S., Sadoughi, F., Tadayon, M.H., Dehnad, A.: Security requirements of Internet of Things-based healthcare system: a survey study. Acta Informatica Medica **27**(4), 253 (2019)
28. Gunduz, M.Z., Das, R.: Cyber-security on smart grid: threats and potential solutions. Comput. Netw. **169**, 107094 (2020). https://linkinghub.elsevier.com/retrieve/pii/S1389128619311235
29. Reshan, A., Saleh, M.: IoT-based application of information security triad. Int. J. Interact. Mob. Technol. **15**(24) (2021)
30. Bacudio, A.G., Yuan, X., Chu, B.T.B., Jones, M.: An overview of penetration testing. Int. J. Netw. Secur. Appl. **3**(6), 19 (2011)
31. Orebaugh, A., Pinkard, B.: Nmap in the Enterprise: Your Guide to Network Scanning. Elsevier (2011)