International Journal of Computing and Digital Systems
ISSN (2210-142X)
Int. J. Com. Dig. Sys. , No. (Mon-20..)

http://dx.doi.org/10.12785/ijcds/XXXXXX

RoArch: A Novel Approach for Handling Security Incidents in ROS-Based Systems

Yash Patel[1] **and Dr. Parag H. Rughani**[2]

[1]*School of Doctoral Studies & Research, National Forensic Sciences University, Gandhinagar, Gujarat*
[2]*School of Cyber Security & Digital Forensics, National Forensic Sciences University, Gandhinagar, Gujarat*

Received Mon. 20, Revised Mon. 20, Accepted Mon. 20, Published Mon. 20

Abstract: [Abstract] In the face of escalating cybersecurity threats targeting robotic systems, the need for advanced forensic frameworks has become critical. Robotic systems, now crucial in key sectors, lack specialized forensic tools, leading to vulnerabilities in security and operational reliability. This article introduces RoArch, a novel forensic framework tailored for robotic systems, addressing the urgent need for specialized forensic tools in this fast-evolving domain. As robotics become integral to sectors like healthcare, defense, and industrial automation, the demand for tailored forensic methodologies has risen. RoArch features two main components: the RoboShell tool for interactive command execution and anomaly detection, and the RuFo framework for capturing live and volatile data, designed to work with both ROS and ROS2 platforms. This research delves into RoArch's architecture, functionality, and practical effectiveness, with a focus on its versatility in various robotic environments and forensic challenges. Through an in-depth analysis, the article illustrates RoArch's utility in scenarios such as cyber security breach investigations in TurtleBot3 robots, underlining its real-world applicability. RoArch represents a significant leap in robotic forensics, promising to improve the reliability and security of robotic systems in numerous industries. The paper concludes by discussing RoArch's future potential in robotic forensics, advocating for expanded compatibility and the integration of machine learning for enhanced predictive analysis.

Keywords: Robotic Forensics, Robotic Cybersecurity, ROS Cybersecurity, ROS Forensics, Robotic Incident Response, Robotic Incident Management.

1. INTRODUCTION

In recent years, robotics has seen rapid and transformative growth across several sectors, including industry [1], healthcare [2], defense [3], and many more [4][5][6]. Integrating complex robotic systems into critical infrastructures and sensitive areas raises unique cyber security [7][8][9] and forensic challenges [10][11]. Despite technological advancements, there's a notable lack in forensic methodologies and tools for robot systems [12]. This deficiency hinders issue resolution in robotic systems and threatens the security and integrity of operations involving these technologies.

Robotic forensics, though developing, is crucial for overcoming current challenges [13]. Existing forensic models, tailored for standard IT environments, fail to address the complexities and dynamism of robotic systems [14]. This shortfall is evident in cases involving intricate hardware and software interplay, real-time processing, and AI/machine learning integration. Additionally, the changing cyber threat landscape demands a forensic framework that is robust, versatile, and adaptable to the fast evolving field of robotics [15].

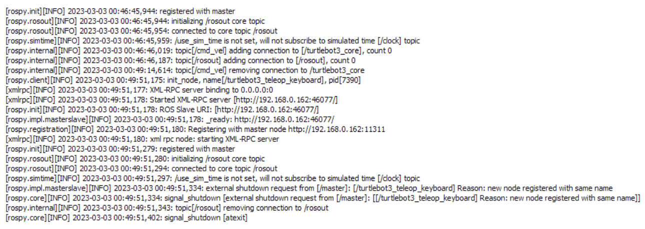

```
[rospy.init][INFO] 2023-03-03 00:46:45,944: registered with master
[rospy.rosout][INFO] 2023-03-03 00:46:45,944: initializing /rosout core topic
[rospy.rosout][INFO] 2023-03-03 00:46:45,954: connected to core topic /rosout
[rospy.simtime][INFO] 2023-03-03 00:46:45,959: /use_sim_time is not set, will not subscribe to simulated time [/clock] topic
[rospy.internal][INFO] 2023-03-03 00:46:46,019: topic[/cmd_vel] adding connection to [/turtlebot3_core], count 0
[rospy.internal][INFO] 2023-03-03 00:46:46,187: topic[/rosout] adding connection to [/rosout], count 0
[rospy.internal][INFO] 2023-03-03 00:49:14,614: topic[/cmd_vel] removing connection to /turtlebot3_core
[rospy.client][INFO] 2023-03-03 00:49:51,175: init_node, name[/turtlebot3_teleop_keyboard], pid[7390]
[xmlrpc][INFO] 2023-03-03 00:49:51,177: XML-RPC server binding to 0.0.0.0:0
[xmlrpc][INFO] 2023-03-03 00:49:51,178: Started XML-RPC server [http://192.168.0.162:46077/]
[rospy.init][INFO] 2023-03-03 00:49:51,178: ROS Slave URI: [http://192.168.0.162:46077/]
[rospy.impl.masterslave][INFO] 2023-03-03 00:49:51,178: _ready: http://192.168.0.162:46077/
[rospy.registration][INFO] 2023-03-03 00:49:51,180: Registering with master node http://192.168.0.162:11311
[xmlrpc][INFO] 2023-03-03 00:49:51,180: xml rpc node: starting XML-RPC server
[rospy.init][INFO] 2023-03-03 00:49:51,279: registered with master
[rospy.rosout][INFO] 2023-03-03 00:49:51,280: initializing /rosout core topic
[rospy.rosout][INFO] 2023-03-03 00:49:51,294: connected to core topic /rosout
[rospy.simtime][INFO] 2023-03-03 00:49:51,297: /use_sim_time is not set, will not subscribe to simulated time [/clock] topic
[rospy.impl.masterslave][INFO] 2023-03-03 00:49:51,334: external shutdown request from [/master]: [/turtlebot3_teleop_keyboard] Reason: new node registered with same name
[rospy.core][INFO] 2023-03-03 00:49:51,334: signal_shutdown [external shutdown request from [/master]: [[/turtlebot3_teleop_keyboard] Reason: new node registered with same name]]
[rospy.internal][INFO] 2023-03-03 00:49:51,343: topic[/rosout] removing connection to /rosout
[rospy.core][INFO] 2023-03-03 00:49:51,402: signal_shutdown [atexit]
```

Figure 1. Illustrates the data logging deficiencies in the robot operating system

The primary issue addressed in this research is that the robot operating system does not store operational logs, such as velocity, movement, and control, as shown in Figure 1. This absence complicates the investigation of robotic systems. The secondary issue is the lack of specific forensic investigation tools or frameworks dedicated to robotic systems, which would otherwise facilitate forensic investigators in resolving crime cases involving robots.

This study presents RoArch, a novel forensic framework designed for robotic systems. Addressing the pressing need for specialized forensic tools, RoArch closes the existing gap with a tailored solution for managing and analyzing robotic systems in their specific operational environment. Built on two key elements, the RoboShell tool and the RuFo framework, it provides a synergistic method to boost the efficiency and efficacy of forensic investigations in robotics.

This research aims to address the key question: How can forensic investigations in robotic systems be optimized for greater efficiency and reliability, given their unique challenges? It focuses on exploring RoArch's design, structure, and operational capabilities, and assesses its effectiveness in practical situations. The study particularly examines RoArch's adaptability to various robotic environments and its capability in managing diverse forensic challenges.

This research has multiple objectives: to conduct an in-depth analysis of the RoArch framework, detailing its components, functionalities, and integration processes; to showcase RoArch's practical applications across several scenarios, emphasizing its versatility and robustness; and to evaluate RoArch's effectiveness in improving forensic investigations in robotics, focusing on efficiency, accuracy, and comprehensiveness.

This research is significant as it has the potential to mutate robotic forensics. RoArch, a specialized framework designed for robot systems, fills an essential gap in robot forensics. Its implementation could significantly enhance the reliability and security of robotic operations in several sectors such as defense, healthcare, industrial automation, and many more.

Additionally, this research provides valuable insights that could cover the way for future developments in robotic forensics, aiding in the creation of more advanced tools and methods. Essentially, this study introduces a groundbreaking framework and establishes a new standard in the field of robot forensics. It lays a foundational basis for subsequent research and innovations in managing and forensically examining robotic systems.

This paper delves into advanced forensic tools and methodologies in robotics. Section 2 presents related work in robot forensics. Section 3 introduces the methodology of RoArch, which includes RoboShell and the RuFo Framework, aimed at enhancing forensic analysis in robotic systems. Section 4 discusses the implementation of RoArch, an innovative strategy for addressing cybersecurity in robotics. Section 5 showcases RoArch's application in a cybersecurity breach case involving TurtleBot3. Section 6 discusses key findings, significance, limitations, and scenarios. Finally, Section 7 concludes with insights and recommendations for future research.

2. Related Work

This section delves into the emerging field of forensic investigation in robotic systems, where the literature, rising, reveals critical insights and developments.

The necessity of a structured digital evidence collection framework for robotic systems was emphasized by Giuseppe Vaciago and Francesca Bosco [16]. They proposed establishing best practice guidelines conforming to national or international standards, stressing the importance of protocols for robotic system security.

Victor Mayoral Vilches, et al. [17] developed ros_volatility, a plugin for the Robot Operating System (ROS) that proved effective in detecting unauthorized ROS node registrations in their experimental setup, identifying the intruder in test scenarios.

Mawj Mohammed Basheer and Asaf Varol [18] highlighted the lack of extensive research in robotic forensics through their review of ROS security and digital forensics, calling attention to the need for further exploration in this domain.

Koscher, et al. [19] examined vulnerabilities in automobile systems, drawing parallels to the need for robust security in complex robotic systems. Casey [20], in a similar vein, addressed digital forensic complexity, suggesting methods adaptable for investigating robotic system anomalies and malfunctions.

The enhancement of forensic methodologies in robotics was pursued by Khalastchi and Kalech [21] through a survey on fault detection in multi-robot systems. Similarly, Abeykoon and Feng [22] proposed a strategy for examining ROS vulnerabilities to cyber threats.

Focusing on the TurtleBot3, Amsters and Slaets [23] analyzed its architectural and functional complexities, illuminating the intricacies of its software framework. Hossen, et al. [24] further contributed by exploring the diagnostic aspects of TurtleBot3's potential failure modes and system performance.

In summary, this analysis underscores the significant gaps and areas for growth in robotic system forensics. There is a clear need for live and dead robot forensics, more resources, advanced expertise, refined methodologies, and innovative techniques, along with the development of comprehensive, universally accepted standards for robotic forensic investigations.

3. Methodology

This section of the paper explores the design and structure of RoArch, a powerful forensic framework specifically adapted for robotic systems, as shown in Figure 2. RoArch is designed to clearly improve the efficiency and effectiveness of forensic investigations in the field of robotics. Mainly RoArch has two key components: RoboShell and the RuFo framework. These tools are integrated within RoArch to enable a dependent approach, enhancing the overall functionality and capabilities of the framework in robotic forensics.

RoArch serves as a foundational element in this re-

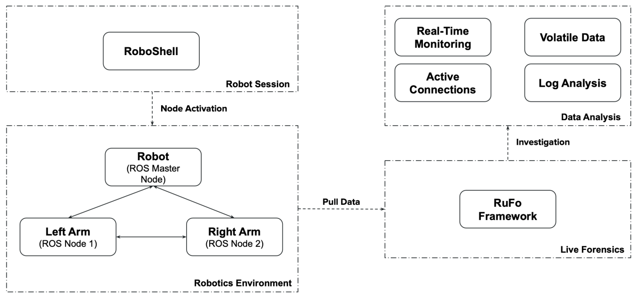


Figure 2. RoArch Overview: Innovative Robot Architecture Design

search, embodying a comprehensive strategy for the management and analysis of robotic systems. The framework is built upon the notion of a Robotics Environment, which constitutes the core of a robot's entire operational range. This environment is dynamic, covering all facets of a robot's functionality.

The RoArch (Robot Architecture) framework enhances the robotic system's operational efficiency and security. It features a Robot Session to describe the active state of robots, and a RoboShell Tool for interactive command execution similar to traditional command-line interfaces. A Node Activation Mechanism manages Robot Operating System (ROS) nodes for specific functionalities, while an advanced Data Analysis and Management sector addresses real-time monitoring and data handling. The architecture includes a Robot's Investigation and Live Forensics component, integrated with the RuFo Framework for real-time forensic analysis, ensuring a comprehensive and secure robotic system architecture.

The design of the architecture highlights the seamless integration of real-time operations with retrospective analysis. It underlines the importance of ongoing system monitoring, data extraction (Pull Data), and the capability to conduct live forensic investigations. This dual emphasis on both real-time and post-operational analysis renders RoArch especially appropriate for high-reliability and mission-critical robotic applications.

Exploring RoArch (Robot Architecture) involves an in-depth look at the core functionalities of its key components: RoboShell and the RuFo Framework. The Table I below provides a comparative overview, highlighting distinctive capabilities such as data analysis, anomaly detection, event handling, compatibility with ROS/ROS2 ,etc.

A. *RoboShell Tool*

RoboShell is developed as an advanced tool tailored to meet the dynamic requirements of robotic system monitoring and analysis. The algorithm of the RoboShell Tool is an automated logging and monitoring script designed for robotic systems, as shown in Figure 3. It initializes a log file, displays a banner, and redirects output to the log with timestamps. The tool extracts and logs the source IP address, retrieves the external IP address, and defaults to 'N/A; on failure. It reads initial and updated CPU stats to calculate and log CPU usage. By capturing and logging all activities in a robot bash session, the tool provides comprehensive terminal logs for analysis.

```
Algorithm 1 RoboShell Tool
Require: Log file location, CPU threshold, Network timeout
Ensure: System and network log with CPU & IP metrics
 1: function GetSourceIP
 2:     for all service in ServiceList do
 3:         if RetrieveIP(service) is successful then
 4:             return IP address
 5:         end if
 6:     end for
 7:     return "N/A"
 8: end function
 9: function MonitorCPU(threshold)
10:     initial ← GetCPUStats()
11:     while True do
12:         Sleep for 1 second
13:         updated ← GetCPUStats()
14:         usage ← CalculateCPU(updated)
15:         if usage > threshold then
16:             return usage
17:         end if
18:     end while
19: end function
20: sourceIP ← GetSourceIP()
21: cpuUsage ← MonitorCPU(threshold)
22: Start new bash session
23: Capture session with script command
24: Exit bash session
25: return Terminal RoboShell log
```

Figure 3. RoboShell Tool - Algorithm

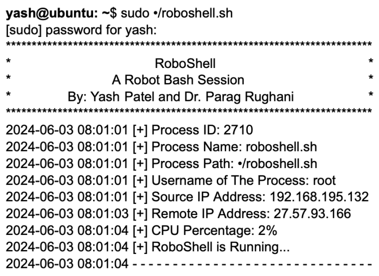

```
yash@ubuntu: ~$ sudo •/roboshell.sh
[sudo] password for yash:
**********************************************************************
*                              RoboShell                             *
*                        A Robot Bash Session                        *
*                 By: Yash Patel and Dr. Parag Rughani               *
**********************************************************************
2024-06-03 08:01:01 [+] Process ID: 2710
2024-06-03 08:01:01 [+] Process Name: roboshell.sh
2024-06-03 08:01:01 [+] Process Path: •/roboshell.sh
2024-06-03 08:01:01 [+] Username of The Process: root
2024-06-03 08:01:01 [+] Source IP Address: 192.168.195.132
2024-06-03 08:01:03 [+] Remote IP Address: 27.57.93.166
2024-06-03 08:01:04 [+] CPU Percentage: 2%
2024-06-03 08:01:04 [+] RoboShell is Running...
2024-06-03 08:01:04 --------------------------------
```

Figure 4. RoboShell Interface: Advanced Tool Integration

It excels in logging robot command interactions during bash sessions, effectively documenting control directives and corresponding robot actions. With its compatibility with both ROS and ROS2 frameworks, RoboShell's adaptability is significantly enhanced, positioning it as an essential instrument in the field of robotic software. Additionally, its intuitive interface is designed to accommodate users of varying expertise, from novices to experienced professionals, as shown in Figure 4.

RoboShell enhances the efficiency of robotic investigations by providing detailed insights into control commands and robot actions. Its capability in detecting anomalies significantly improves the system's reliability and safety. This tool serves as an instrument for analyzing current operations and also acts as a catalyst for future research and training. It enables particular scrutiny of robotic behaviors, facilitating a deeper understanding of their operational dynamics.

TABLE I. Comparative Functions of RoboShell and RuFo Framework

Capability	RoboShell Tool	RuFo Framework
Data Analysis	Command/action history with timestamps.	System, memory, network, logs, process, file analysis.
Anomaly Detection	Identifies and facilitates corrections of operational issues.	Artifact collection, network monitoring, malware detection.
Event Handling	Event recreation for root cause analysis.	-
Compatibility ROS/ROS2	Compatible with ROS/ROS2 middleware platforms.	Compatible with ROS/ROS2 middleware platforms.
Development Support	Real-time directive monitoring and feedback.	-
Security	-	System integrity and security enhancements.
User Accessibility	User-friendly interface for technical/non-technical users.	User-friendly interface for technical/non-technical users.

Within the expansive field of robotics and automation, the collective attributes of RoboShell contribute significantly to enhanced efficiency and reliability. Its capacity to enable process replication is essential for complex investigations. The tool's architecture is forward thinking, as demonstrated by its compatibility with both ROS and ROS2 frameworks. Additionally, its advanced monitoring features are vital in an era where immediate feedback is key to the successful execution of robotic operations.

B. *RuFo Framework*

The Robot Utility Forensics (RuFo) Framework represents an advancement in the field of robotic forensics, responding to the increasing incorporation of robotic systems across diverse industries. Designed to tackle the distinct challenges inherent in robotic systems, RuFo provides extensive tools for the acquisition of live and volatile data. Its compatibility with both the ROS (Robot Operating System) and ROS2 underscores its applicability and flexibility in modern robotic research and practical applications.

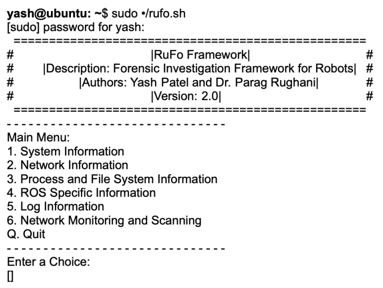
The algorithm of the RuFo Framework is a comprehensive logging and monitoring tool specifically designed for robotic systems, as shown in Figure 5. It begins by defining a log file and redirecting output to it. The framework presents a script header and a main menu, enabling users to select from various options, including system information, network information, process and file system details, ROS-specific data, log information, and network monitoring. Each option leads to a submenu offering specific functions, such as displaying CPU information, scanning for suspicious files, or retrieving ROS topics. The menu repeats until the user chooses to quit, ensuring detailed terminal logs for thorough analysis.

```
Algorithm 2 RuFo Framework
Require: LOG_FILE, Config Parameters
Ensure: Forensic Data
Init logging → LOG_FILE
Load config → defaults
Show metadata
Menu:
  a. System Information
  b. Network Information
  c. Process/File System Information
  d. ROS Information
  e. Logs Information
  f. Network Monitoring/Scanning
while User session do
    Render menu → Capture input
    if User input is "a" then
        Collect metrics → Check thresholds
    else if User input is "b" then
        Assess network status
    else if User input is "c" then
        List processes, Check file system
    else if User input is "d" then
        Query Nodes/Topics
    else if User input is "e" then
        Search → Filter → Export
    else if User input is "f" then
        Scan ports → Analyze traffic
    else if User input is "Quit" then
        Exit loop → Finalize
    else
        Invalid selection → Prompt retry
    end if
end while
Save logs → Close session
```

Figure 5. RuFo Framework - Algorithm

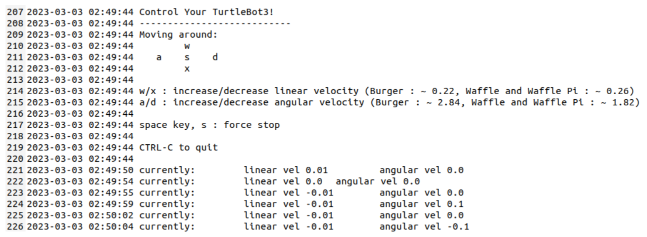
The RuFo framework focuses on strengthening security and forensic effectiveness, incorporating several critical features, as shown in Figure 6. It firstly integrates Proactive Threat Detection, which facilitates the identification and pre-emption of potential security breaches and malware, playing a pivotal role in preserving the integrity and safety of the system. Secondly, it encompasses Detailed Incident Analysis, offering in-depth data essential for comprehensive incident investigations and the formulation of efficient response strategies. This level of detail in data analysis is key to comprehensively understanding and effectively addressing security incidents.

Third, the framework facilitates Continuous Monitoring and Troubleshooting, guaranteeing uninterrupted surveillance and health assessment of robotic systems. This capability allows for the prompt detection and rectification of problems, promoting sustained operational functionality. Furthermore, the framework enhances Operational Efficiency by offering a holistic view of the system. This overarching perspective assists in optimizing the use of resources and managing time effectively, thus boosting the



```
yash@ubuntu: ~$ sudo ./rufo.sh
[sudo] password for yash:
 ===================================================
#                    |RuFo Framework|                    #
#   |Description: Forensic Investigation Framework for Robots|  #
#          |Authors: Yash Patel and Dr. Parag Rughani|        #
#                      |Version: 2.0|                      #
 ===================================================
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Main Menu:
1. System Information
2. Network Information
3. Process and File System Information
4. ROS Specific Information
5. Log Information
6. Network Monitoring and Scanning
Q. Quit
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Enter a Choice:
[]
```

Figure 6. RuFo Framework: Main Menu Interface Snapshot



```
2023-03-03 02:49:44 Control Your TurtleBot3!
2023-03-03 02:49:44 ---------------------------
2023-03-03 02:49:44 Moving around:
2023-03-03 02:49:44         w
2023-03-03 02:49:44    a    s    d
2023-03-03 02:49:44         x
2023-03-03 02:49:44
2023-03-03 02:49:44 w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
2023-03-03 02:49:44 a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)
2023-03-03 02:49:44
2023-03-03 02:49:44 space key, s : force stop
2023-03-03 02:49:44
2023-03-03 02:49:44 CTRL-C to quit
2023-03-03 02:49:44
2023-03-03 02:49:50 currently:        linear vel 0.01         angular vel 0.0
2023-03-03 02:49:54 currently:        linear vel 0.0  angular vel 0.0
2023-03-03 02:49:55 currently:        linear vel -0.01        angular vel 0.0
2023-03-03 02:49:59 currently:        linear vel -0.01        angular vel 0.1
2023-03-03 02:50:02 currently:        linear vel -0.01        angular vel 0.0
2023-03-03 02:50:04 currently:        linear vel -0.01        angular vel -0.1
```

Figure 7. RoboShell Analysis: Detailed ROS Log File Visualization

efficiency of operations. Together, these attributes reinforce the framework's ability to protect against threats and ensure proficient forensic analysis and operational management.

The RuFo framework focuses on strengthening security and forensic effectiveness, incorporating several critical features. It firstly integrates Proactive Threat Detection, which facilitates the identification and pre-emption of potential security breaches and malware, playing a pivotal role in preserving the integrity and safety of the system. Secondly, it encompasses Detailed Incident Analysis, offering in-depth data essential for comprehensive incident investigations and the formulation of efficient response strategies. This level of detail in data analysis is key to comprehensively understanding and effectively addressing security incidents.

Third, the framework facilitates Continuous Monitoring and Troubleshooting, guaranteeing uninterrupted surveillance and health assessment of robotic systems. This capability allows for the prompt detection and rectification of problems, promoting sustained operational functionality. Furthermore, the framework enhances Operational Efficiency by offering a holistic view of the system. This overarching perspective assists in optimizing the use of resources and managing time effectively, thus boosting the efficiency of operations. Together, these attributes reinforce the framework's ability to protect against threats and ensure proficient forensic analysis and operational management.

4. Implementing RoArch: Enhancing Forensic Investigations in Robotics

A novel Robot Architecture (RoArch) has been developed to optimize forensic investigations within robotic systems. This architecture features the RoboShell tool, which efficiently captures and archives log data from the robot operating system. Moreover, the integration of the Robot Utility Forensics (RuFo) framework into this architecture significantly expedites the investigation process.

The combined utilization of RoboShell and RuFo establishes a comprehensive standardization protocol for forensic procedures in robotics, particularly in environments where the robot operating system is utilized. The deployment of RoArch is expected to substantially improve both the efficiency and effectiveness of forensic analyses in the field of robotics.

Table II and Table III describes the rigorous compatibility testing of RoArch across a spectrum of ROS and ROS2 platforms, encompassing installation, operational functionality, interoperability, performance, and security aspects. It affirms RoArch's comprehensive validation, showcasing its adaptability and robustness within diverse robotic environments.

A. *Evaluating RoboShell: A Key Interface for ROS Interaction*

This section presents the results from assessing the RoboShell Tool, a key element for interfacing with robotic systems, especially when working with the TurtleBot3.

After initiating RoboShell, the initial step was to start the ROS core (roscore), a fundamental component for activating the ROS system and facilitating communication through multiple background processes. These processes encompassed the ROS Master, responsible for name registration and lookup services, the Parameter Server, which manages the dynamic storage and retrieval of parameters, and the rosout logging node, essential for gathering debug messages.

In the environment of the ROS framework, a calibration node was utilized, specifically for hardware components like sensors or actuators. An instance of this was the integration of a camera with the TurtleBot3, which necessitated calibration for precise image processing. This process included compensating for lens deformation. The calibration data were preserved either as parameters on the Parameter Server or within configuration files.

Teleoperation (Teleop) was executed through a dedicated node, such as turtlebot3_teleop_keyboard, which enabled remote control of the TurtleBot3. This node functioned by interpreting keyboard keypresses and translating them into ROS messages, which in turn directed the movements of the robot.

The logging proficiency of RoboShell was showcased via its detailed log files, which particularly recorded operational parameters and system statuses of the TurtleBot3, as shown in Figure 7. These logs proved vital for troubleshooting purposes, monitoring robot functionalities, and auditing the commands executed during specific timeframes.

The log files comprehensively captured sessions where

TABLE II. Compatibility Testing of RoArch with ROS Platforms

Test Criteria/Platform	ROS Jade Turtle	ROS Kinetic Kame	ROS Lunar Loggerhead	ROS Melodic Morenia	ROS Noetic Ninjemys
Installation & Configuration	×	✓	×	✓	✓
Operational Functionality	×	✓	×	✓	✓
Interoperability	×	✓	×	✓	✓
Performance & Stability	×	✓	×	✓	✓
Security & Compliance	×	✓	×	✓	✓

TABLE III. Compatibility Testing of RoArch with ROS2 Platforms

Test Criteria/Platform	ROS2 Eloquent Elusor	ROS2 Foxy Fitzroy	ROS2 Galactic Geochelone	ROS2 Humble Hawksbill	ROS2 Iron Irwini
Installation & Configuration	×	✓	×	✓	✓
Operational Functionality	×	✓	×	✓	✓
Interoperability	×	✓	×	✓	✓
Performance & Stability	×	✓	×	✓	✓
Security & Compliance	×	✓	×	✓	✓

the TurtleBot3 was manipulated using a keyboard teleoperation node. These logs explained the initiated processes, configurations of operational parameters, and specific commands for controlling the robot's movement, including the modulation of linear and angular velocities. Additionally, it documented the process ID, process name, username associated with the process, source and destination IP addresses, CPU usage percentage, and a timeline of events.

The detailed entries in the log files served as a significant resource for comprehending the interaction dynamics of the TurtleBot3 during manual navigation. This information not only aided in achieving precise control over the robot but also provided a basis for assessing its responsiveness during interactive tasks.

The assessment of RoboShell highlighted its effectiveness in facilitating interactions with robotic systems, especially in intricate operational environments such as those involving the TurtleBot3. The combination of its intuitive interface and sophisticated logging and debugging capabilities was found to be crucial for continuous real-time monitoring as well as thorough post-event analysis.

B. Understanding the RuFo Framework: Unveiling Forensic Potential in Robotics

The Robot Utility Forensics (RuFo) Framework integrates more than 40 specialized techniques for forensic investigation. This subsection outlines the operational methodologies and functionalities of the RuFo Framework, specifically designed for implementation within a robot operating system. The mathematical equation for the RuFo Framework is depicted in Figure 8. The equation models to initialize log files, redirect outputs, and display headers. This is followed by a main menu loop where user choices activate specific submenus, including system, network, process, ROS, logs, and network monitoring. Each submenu executes designated tasks or returns to the main menu, continuing this process until the user terminates the session.

Notations:

L : Define LOG_FILE
R : Redirect stdout and stderr to LOG_FILE
H : Display script header
M : Main menu function
S_i : sys_info_menu() choice i
N_i : net_info_menu() choice i
P_i : proc_info_menu() choice i
R_{s_i} : ros_info_menu() choice i
L_{g_i} : log_info_menu() choice i
N_{m_i} : net_mon_menu() choice i
Q : Quit program
I : Infinite loop to keep the menu running until user quits
G : Terminal log

Equation:

$$\mathrm{RuFo}(G) = L + R + H + \left(M + \sum_{j=1}^{k} F_m \right) + I$$

where

$$F_m = \begin{cases} \sum_{i=1}^{a} S_i, & \text{if user selects System Information} \\ \sum_{i=1}^{b} N_i, & \text{if user selects Network Information} \\ \sum_{i=1}^{c} P_i, & \text{if user selects Process and File System Information} \\ \sum_{i=1}^{d} R_{s_i}, & \text{if user selects ROS Specific Information} \\ \sum_{i=1}^{e} L_{g_i}, & \text{if user selects Log Information} \\ \sum_{i=1}^{f} N_{m_i}, & \text{if user selects Network Monitoring and Scanning} \\ Q, & \text{if user chooses to quit} \end{cases}$$

I = Infinite loop calling M until Q

Figure 8. RuFo Framework Mathematical Equation

This interface acts as a gateway to a range of forensic investigative features, methodically organized into six principal categories: System Information, Network Information, Process and File System Information, ROS Specific Information, Log Information, and Network Monitoring and Scanning. Choosing any of these options leads to a dedicated submenu, equipped with a variety of investigative tools and functions, as depicted in Figures 9.

This subsection highlights the importance of both the submenus and the main menu, which includes a feature to exit the program. It also provides the functionality to navigate back to the main menu from any submenu. The subsection emphasizes the necessity of safety measures and precautionary protocols, advising users to thoroughly inspect scripts prior to execution, maintain system security through regular updates, and consistently conduct data backups to avert potential data loss.

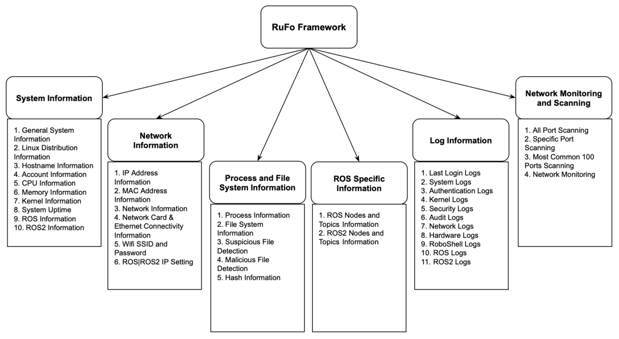


Figure 9. RuFo Framework: Detailed Sub-Menu

1) *System Information of Robot*

The RuFo Framework introduces a comprehensive System Information feature, particularly designed for system administrators and forensic analysts. Accessible via a user-friendly submenu, it delivers extensive insights across multiple dimensions: from General System, Linux Distribution, and Hostname Information to detailed CPU, Memory, and Kernel specifics, alongside security-centric Account Information.

Furthermore, it encompasses ROS and ROS2 configurations, presenting an indispensable tool for diagnosing and managing robotic systems. This feature not only underscores the framework's adaptability in addressing diverse forensic needs but also its alignment with advanced security practices and system diagnostics, enhancing operational transparency and efficiency in robotic ecosystems.

In a critical military robotics deployment, a sophisticated malware attack was detected in a surveillance unit. Forensic investigators used the RuFo Framework's System Information feature to quickly access detailed CPU, Memory, Kernel specifics, and ROS configurations. The in-depth analysis revealed unauthorized access patterns, identifying the malware's origin, and preventing a potential security breach, thereby ensuring mission-critical integrity.

2) *Network Information of Robot*

The RuFo Framework's Network Information functionality emerges as an essential tool for forensic analysis within interconnected robotic ecosystems. Through a detailed submenu, it enables the examination and configuration of network parameters, including IP and MAC addresses, network cards, and wireless settings, tailored for technical professionals. Its command-line interface facilitates precise network diagnostics and adjustments, crucial for network administrators, robotics specialists, and forensic analysts.

This subsection particularly documents the framework's capability to navigate complex network infrastructures, offering indispensable insights for configuring, troubleshooting, and securing network communications in robotic systems, thereby underscoring its pivotal role in advancing forensic investigations in networked environments.

In industrial robotics, a network anomaly disrupted production workflows. Forensic investigators used the RuFo Framework's Network Information functionality to access detailed network parameters, such as IP and MAC addresses, and network card configurations. Utilizing the command line interface, they identified and isolated a compromised network node, restoring secure network communications. This precise troubleshooting ensured uninterrupted robotic operations and protected industrial processes.

3) *Processes and File System Information of Robot*

Within the RuFo Framework, the Processes and File System Information functionality stands out as a pivotal tool for forensic and security analysis in robotic systems. Accessed through a detailed submenu, it offers a deep dive into system processes, file system data, and the detection of suspicious or malicious files. Tailored for system administrators and security professionals, its command-line interface allows for an intricate examination of running processes and disk usage, alongside evaluating file integrity through hash value analysis.

This section underscores the framework's capacity for particular system evaluations, essential in identifying security vulnerabilities and ensuring the integrity of robotic environments, showcasing its indispensable role in forensic investigations.

In a healthcare robotic system, unauthorized access to patient data was detected. Forensic investigators used the RuFo Framework's Processes and File System Information functionality to examine system processes and file integrity through hash value analysis. The command line interface identified a rogue process and altered files, revealing the data breach's entry point. Instantaneous remediation ensured data security and maintained patient confidentiality.

4) *ROS Specific Information of Robot*

The RuFo Framework's ROS Specific Information functionality is a critical asset for professionals managing robotic systems utilizing the Robot Operating System (ROS) and ROS2. Offering deep insights into computational entities (nodes) and communication channels (topics), this feature facilitates the precise monitoring and forensic analysis of robotic communications.

It adeptly handles both ROS and ROS2 architectures, illustrating its versatility in tracing communication networks and understanding dynamic interactions within robots. Through detailed examination of nodes like /turtlebot3_teleop_keyboard and topics such as /cmd_vel, the framework demonstrates its exceptional capability in diagnosing and enhancing the performance of robotic systems, reinforcing its value in robotics research and development.

In a domestic robotics application, unexpected behavioral anomalies in a home assistant robot were reported.

Forensic investigators used the RuFo Framework's ROS Specific Information functionality to examine computational nodes and communication topics. Detailed forensic investigation identified misconfigured communication channels causing uncertain movements. Correcting these configurations restored normal functionality, ensuring reliable and safe robotic operations within the household environment.

5) Logs Information of Robot

The Log Information feature within the RuFo Framework is a cornerstone for forensic analysis, offering exhaustive access to various logs critical for robotic system investigations. This functionality categorizes logs into Last Login, System, Authentication, Kernel, Security, Audit, Network, and specifically, RoboShell, ROS, and ROS2 logs, facilitating a comprehensive review of system activities. These logs play a pivotal role in constructing event timelines, identifying system irregularities, and troubleshooting within ROS environments.

The RuFo Framework's adept integration with ROS and ROS2 underlines its utility in forensic investigations, system diagnostics, and ensuring robotic system integrity, making it an invaluable tool for professionals navigating the complexities of robotic system analysis.

In a military robotic surveillance system, an unexpected shutdown compromised mission critical operations. Forensic investigators used the RuFo Framework's Log Information feature to access Authentication, Kernel, RoboShell, ROS, and ROS2 logs. Detailed forensic investigation revealed unauthorized access attempts and kernel panics correlating with the shutdown. Constructing an event timeline tracker, they identified a cyber attack vector, implemented security patches, and restored system integrity, ensuring operational continuity and mission success.

6) Network Monitoring and Scanning of Robot

The RuFo Framework's Network Monitoring and Scanning feature stands as a testament to its expertise in cybersecurity within robotic systems. Offering a tailored submenu for intricate network analyses, including comprehensive port scanning and real-time traffic monitoring, it addresses various facets of network security.

From unveiling active ports and vulnerabilities to enabling focused examination of network behavior, the framework's capabilities are crucial for cybersecurity professionals. Its detailed command-line interface serves specifically to technical users, facilitating precise network diagnostics and forensic investigations. This versatility underscores the framework's essential role in protecting robot systems against cyber threats, ensuring operational security and integrity.

In a healthcare robotics network, a data breach was suspected, jeopardizing patient confidentiality. Forensic investigators utilized the RuFo Framework's Network Monitoring and Scanning feature for comprehensive port scanning and real-time traffic monitoring. The command line interface identified an open, vulnerable port and unusual network behavior, pinpointing a malware infiltration. Instantaneous isolation and remediation restored network security and protected sensitive patient data.

5. RoArch in Action: Investigating TurtleBot3's Cyber Security Breach

This section evaluates the effectiveness of the RoArch Framework through a real world experiment involving the TurtleBot3, subjected to a cyber attack named pico-ducky. This attack, crafted via a Raspberry Pi Pico in a USB Rubber Ducky configuration, led to various malfunctions in the TurtleBot3. The experiment provided a practical condition to assess RoArch's ability to identify and analyze robotic cybercrimes. The attack's impact on the TurtleBot3 exemplified potential vulnerabilities in robotic systems. Using the RoboShell tool and the RuFo Framework, the experiment demonstrated the efficacy of RoArch in a real world scenario. Key architectural elements and their roles were as follows:

- RoboShell: Facilitated direct interaction with TurtleBot3, allowing control and communication, which were crucial for understanding the attack's mechanics.
- Raspberry Pi Pico: Served as the attack vector, demonstrating how external devices could manipulate robotic operations.
- TurtleBot3: Highlighted as both an operational unit and a vulnerability point, susceptible to external manipulations.
- Command and Control (C2) Station: Illustrated the remote orchestration of the TurtleBot3 post-attack, a typical cyber attack model.
- RuFo Framework: Played a pivotal role in real-time monitoring and analysis. Its capabilities in volatile data monitoring, active connections oversight, log, and data analysis were instrumental in flagging and investigating the cyber incident.

The architectural analysis underscored the comprehensive nature of RoArch, focusing on robotic cyber vulnerabilities and forensic solutions. The experiment, illustrated in Figure 10, utilized both RoboShell and the RuFo Framework effectively, showcasing their synergy in detecting and analyzing security breaches within the robotics environment. This integrated approach affirmed RoArch's robustness in addressing the complexities of robotic cyber security and forensic investigation.

A. Exploitation of TurtleBot3 using Raspberry Pi Pico

The section discusses an innovative exploitation algorithm using the Raspberry Pi Pico for automating control of a TurtleBot3 robot. This algorithm leverages the Pico microcontroller board's ability to act as a Human Interface

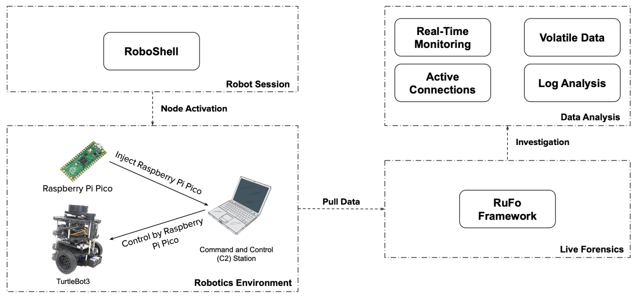
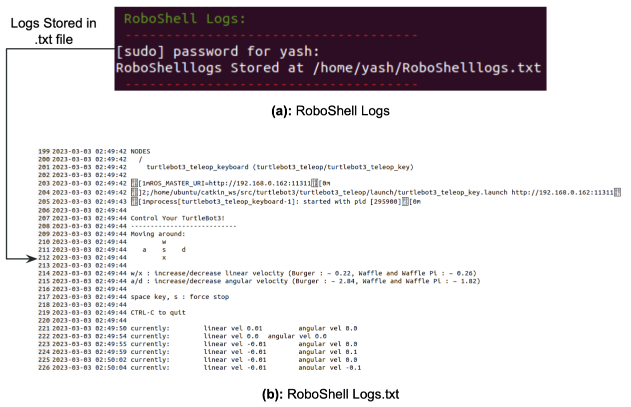


Figure 10. Analyzing the Pico-Ducky Cyber Incident: An Architectural Perspective



```
RoboShell Logs:
---------------------------------
[sudo] password for yash:
RoboShelllogs Stored at /home/yash/RoboShelllogs.txt
---------------------------------
```

(a): RoboShell Logs

```
2023-03-03 02:49:42 NODES
2023-03-03 02:49:42   /
2023-03-03 02:49:42     turtlebot3_teleop_keyboard (turtlebot3_teleop/turtlebot3_teleop_key)
2023-03-03 02:49:42
2023-03-03 02:49:42 ␛[1mROS_MASTER_URI=http://192.168.0.162:11311␛[0m
2023-03-03 02:49:42 ␛]2;/home/ubuntu/catkin_ws/src/turtlebot3/turtlebot3_teleop/launch/turtlebot3_teleop_key.launch http://192.168.0.162:11311␛
2023-03-03 02:49:43 ␛[1mprocess[turtlebot3_teleop_keyboard-1]: started with pid [295900]␛[0m
2023-03-03 02:49:44
2023-03-03 02:49:44 Control Your TurtleBot3!
2023-03-03 02:49:44 ---------------------------
2023-03-03 02:49:44 Moving around:
2023-03-03 02:49:44         w
2023-03-03 02:49:44    a    s    d
2023-03-03 02:49:44         x
2023-03-03 02:49:44
2023-03-03 02:49:44 w/x : increase/decrease linear velocity (Burger : ~ 0.22, Waffle and Waffle Pi : ~ 0.26)
2023-03-03 02:49:44 a/d : increase/decrease angular velocity (Burger : ~ 2.84, Waffle and Waffle Pi : ~ 1.82)
2023-03-03 02:49:44
2023-03-03 02:49:44 space key, s : force stop
2023-03-03 02:49:44
2023-03-03 02:49:44 CTRL-C to quit
2023-03-03 02:49:44
2023-03-03 02:49:50 currently:        linear vel 0.01         angular vel 0.0
2023-03-03 02:49:54 currently:        linear vel 0.0  angular vel 0.0
2023-03-03 02:49:55 currently:        linear vel -0.01        angular vel 0.0
2023-03-03 02:49:59 currently:        linear vel -0.01        angular vel 0.1
2023-03-03 02:50:02 currently:        linear vel -0.01        angular vel 0.0
2023-03-03 02:50:04 currently:        linear vel -0.01        angular vel -0.1
```

(b): RoboShell Logs.txt

Figure 11. RoboShell Logs Acquiring from RuFo Framework

Device (HID), specifically emulating keyboard functions. It's important to note that the ethical deployment of this script is permissible exclusively within authorized robotic systems, as its unauthorized use could be construed as malicious.

The algorithm begins with an Initialization and Keyboard Setup phase. Here, the Raspberry Pi Pico is configured to mimic a keyboard by importing necessary modules like time, board, usb_hid, and various adafruit_hid components. This setup is crucial for the Pico to interact with the host device as a virtual keyboard.

In the next phase, Login Attempts, the script utilizes the keyboard object to input a series of common administrative usernames and passwords. This involves methodical entry with pauses between attempts, mimicking human typing rhythms and avoiding rapid login attempt detection that could trigger security protocols.

The third phase, Interrupt the Teleop_Keyword Node, involves more complex operations. The Pico simulates pressing CTRL+ALT+T to open a terminal window on the host Linux system. After opening the terminal, the script executes commands to establish an SSH connection to a specified IP address and launches a teleoperation node for the TurtleBot3, setting up the environment variable TURTLEBOT3_MODEL and initiating the roslaunch turtlebot3_teleop turtlebot3_teleop_key.launch command.

In the final phase, Control the Teleop_Keyword Node, the script opens a new terminal and repeats the SSH connection and teleoperation node launching process. This redundancy ensures continued control of the TurtleBot3 in case of any disruptions. The script inputs single-letter keystrokes corresponding to different movement directions for the robot, such as forward, backward, stop, right, and left. Each command is spaced with a 7-second pause for the robot to respond appropriately.

After executing the movement commands, the script pauses for 60 seconds. This extended pause allows time to observe the robot's actions, confirming the execution of the final command or serving as a buffer before the script concludes.

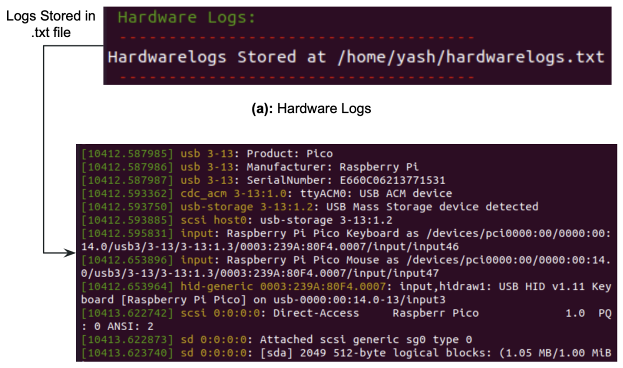
The algorithm demonstrates a sophisticated interplay between the Raspberry Pi Pico and the TurtleBot3, enabling tasks like testing, remote operation, and educational demonstrations. The final outcome is the automated control of the robot's movements, highlighting the script's effectiveness in executing complex login and command sequences through the Raspberry Pi Pico, which adeptly simulates human keyboard interactions.

B. *Investigating Incident using RuFo Framework*

This section presents a detailed forensic analysis of an incident involving a TurtleBot3 robot, conducted using the RuFo Framework. The primary objective was to ascertain whether the issues with the TurtleBot3 were due to technical malfunctions or a result of a cyber attack.

Utilizing the Log Information menu of the RuFo framework, the investigation began with the analysis of RoboShell Logs, option 9 in the main menu. Access to these logs required administrative privileges, as indicated by the terminal's request for a sudo password, as shown in Figure 11. The RoboShellLogs.txt file, located at /home/yash/RoboShellLogs.txt, contained a comprehensive record of logged activities, including command line operations and the initiation of processes identifiable by Process IDs. The commands logged, such as 'w', 'a', 's', 'd', and 'x', corresponded to the robot's movements. The logs also detailed adjustments to the robot's linear and angular velocity and provided status updates on these velocities at specific times. This analysis revealed remote operation of the TurtleBot3 but did not confirm a connection from a specific IP address or machine.

The next step was examining Hardware Logs, selected as option 8 from the Log Information menu, as shown in Figure 12. The hardware logs, stored at a specified file path, offered timestamped records of hardware-related events and states. These logs included entries for network packets blocked by a firewall, connections of USB devices, and hard disk activities. Notably, the logs identified the connection of a Raspberry Pi Pico, detailing its classification as a



```
Hardware Logs:
------------------------------------
Hardwarelogs Stored at /home/yash/hardwarelogs.txt
------------------------------------
```

(a): Hardware Logs

```
[10412.587985] usb 3-13: Product: Pico
[10412.587986] usb 3-13: Manufacturer: Raspberry Pi
[10412.587987] usb 3-13: SerialNumber: E660C06213771531
[10412.593362] cdc_acm 3-13:1.0: ttyACM0: USB ACM device
[10412.593750] usb-storage 3-13:1.2: USB Mass Storage device detected
[10412.593885] scsi host0: usb-storage 3-13:1.2
[10412.595831] input: Raspberry Pi Pico Keyboard as /devices/pci0000:00/0000:00:
14.0/usb3/3-13/3-13:1.3/0003:239A:80F4.0007/input/input46
[10412.653896] input: Raspberry Pi Pico Mouse as /devices/pci0000:00/0000:00:14.
0/usb3/3-13/3-13:1.3/0003:239A:80F4.0007/input/input47
[10412.653964] hid-generic 0003:239A:80F4.0007: input,hidraw1: USB HID v1.11 Key
board [Raspberry Pi Pico] on usb-0000:00:14.0-13/input3
[10413.622742] scsi 0:0:0:0: Direct-Access     Raspberr Pico             1.0  PQ
: 0 ANSI: 2
[10413.622873] sd 0:0:0:0: Attached scsi generic sg0 type 0
[10413.623740] sd 0:0:0:0: [sda] 2049 512-byte logical blocks: (1.05 MB/1.00 MiB
```

(b): Hardware Logs.txt

Figure 12. Hardware Logs Acquiring from RuFo Framework

USB ACM and Mass Storage device and its detection as a keyboard and mouse input device.

Furthermore, the logs registered SCSI commands related to storage device communications, indicating successful interaction between the system and attached hardware components. The investigation revealed the connection of a Raspberry Pi Pico to the TurtleBot3 robot at the incident's time, functioning as a keyboard and mouse peripheral. The matching timestamps in the hardware and RoboShell logs provided conclusive evidence that the Raspberry Pi Pico was involved in controlling the TurtleBot3.

This forensic investigation showcased the capability of the RuFo Framework in providing detailed and organized logs, crucial for diagnosing technical issues, conducting forensic examinations, auditing device connections, and verifying peripheral device's operations. The precise timestamps and device identifiers in the logs facilitated efficient analysis, crucial for understanding the intricacies of the incident involving the TurtleBot3 robot.

6. Discussion

The exploration of RoArch, a novel forensic framework for robotic systems, reveals a significant advancement in the field of robotics forensic. This comprehensive research delineates the intricacies of RoArch, highlighting its two primary components: RoboShell and the RuFo framework. Integrated within RoArch, these tools offer a synergistic approach to enhancing the overall functionality and capabilities of robotic forensics, addressing the urgent need for efficient and effective solutions in this rapidly evolving field. Both tools protect original computer programming tools from unauthorized uses through the Indian Copyright (Amendment) Act of 2012 [25][26].

RoArch serves as a foundation in this research, embodying a strategy that encompasses the management and analysis of robotic systems. The architecture's foundation on the Robotics Environment concept, covering the entire operational spectrum of a robot, marks a significant leap in understanding and managing these complex systems. This environment's dynamic nature emphasizes the multifaceted functionalities inherent in robotics, necessitating a framework like RoArch for comprehensive oversight.

Delving into the specifics, RoboShell tool emerges as a standout component. Its design as an interactive command interface aligns with conventional systems, yet it exhibits advanced capabilities in logging robot command interactions and identifying operational anomalies. This adaptability to both ROS and ROS2 frameworks enhances its utility, making it a valuable tool for a wide range of users, from beginners to experts in the domain. Its proficiency in recording operations and diagnostics, coupled with its anomaly detection and event recreation capabilities, underlines its critical role in advancing robotic investigations and ensuring system reliability.

The RuFo Framework, as another integral component of RoArch, brings to the cutting edge its specialized tools for real-time robotic forensics. With functionalities that range from data analysis, security, and a user-friendly interface, the RuFo Framework stands out in its ability to conduct thorough investigations within robot's systems. It addresses the unique challenges in the robotic domain, providing a holistic perspective of robotic operations that is vital for forensic investigation and security enhancement.

The practical applications of RoArch extend to several critical sectors, demonstrating its versatility and importance. In healthcare robotics, it ensures the integrity and security of sensitive data, a necessity for patient safety. In industrial automation, RoArch plays a pivotal role in maintaining the reliability and safety of automated systems. Furthermore, in defense and security, its application in incident investigations involving robotic systems is invaluable, ensuring the security and effectiveness of these crucial technologies.

A practical case scenario could involve deploying RoArch in an automotive manufacturing environment. Here, it could monitor the operational integrity of assembly line robots, detecting anomalies and conducting forensic analyses in case of malfunctions or security breaches. This application not only ensures the safety of the manufacturing process but also guarantees the quality of the vehicles produced, showcasing RoArch's potential in enhancing operational efficiency and product quality.

RoArch marks a notable progression in robotic forensics, combining real-time and retrospective analysis while being adaptable to diverse robotic systems. This positions it as a valuable asset in future robotic system management and security. Future research and development should aim at broadening its compatibility and improving the user interface, thereby making it accessible to a broader spectrum of users and applicable in various scenarios.

7. Conclusion

This research has methodically examined the architecture and operational aspects of RoArch, an innovative forensic framework designed specifically for robotic systems. This framework responds to the urgent requirement for streamlined and impactful forensic investigations in the field of robotics. The central aim was to present and evaluate the functionalities of RoArch, concentrating on its two key constituents: the RoboShell tool and the RuFo framework. The integration of these components creates a comprehensive methodology, significantly improving the efficiency and efficacy of forensic examinations in robotics.

This study's principal outcomes include the effective deployment of RoArch across diverse scenarios, underscoring its versatility and dependability. The RoboShell tool exhibited its adaptability, excelling in recording robot command interactions, identifying operational anomalies, and aiding in the reconstruction of events. Conversely, the RuFo framework demonstrated its effectiveness in real-time robotic forensic analysis, equipped with an extensive array of data analysis tools and advanced security features. Collectively, these elements constitute a formidable system, adept at navigating the intricate challenges of robotic forensic investigations.

Despite its successes, the study acknowledges certain limitations. The application of RoArch in real-world scenarios highlighted challenges in user interface design and system compatibility, suggesting areas for future improvement. Additionally, while effective in the scenarios tested, further research is needed to evaluate RoArch's performance across a broader range of robotic systems and environments.

The practical involvement of RoArch is substantial, especially in areas where robotics play a crucial role, such as defense/military, healthcare, industrial automation, and many more. Its ability to ensure the robot's system reliability and security is important in these critical applications. For instance, in automotive manufacturing, RoArch could significantly enhance the safety and quality of production processes by monitoring and analyzing the operations of assembly line robots.

Future research directions involve broadening the framework's compatibility to encompass a more diverse array of robotic systems and refining the user interface for greater accessibility to individuals with different levels of technical proficiency. Additionally, incorporating advanced machine learning algorithms into the framework could enhance its predictive capabilities, thereby facilitating more proactive approaches to forensic investigations.

Overall, RoArch marks a notable progression in robotic forensics. The development and effective implementation of this framework highlight the critical role of specialized tools and approaches in handling the intricacies and safeguarding the security of modern robotic systems. RoArch establishes a new benchmark in the domain and lays the groundwork for future advancements in the management and forensic examination of robotic systems.

References

[1] M. Hägele, K. Nilsson, J. N. Pires, and R. Bischoff, "Industrial robotics," *Springer handbook of robotics*, pp. 1385–1422, 2016.

[2] M. Kyrarini, F. Lygerakis, A. Rajavenkatanarayanan, C. Sevastopoulos, H. R. Nambiappan, K. K. Chaitanya, A. R. Babu, J. Mathew, and F. Makedon, "A survey of robots in healthcare," *Technologies*, vol. 9, no. 1, p. 8, 2021.

[3] G. E. Marchant, B. Allenby, R. Arkin, E. T. Barrett, J. Borenstein, L. M. Gaudet, O. Kittrie, P. Lin, G. R. Lucas, R. O'Meara *et al.*, "International governance of autonomous military robots," *Colum. Sci. & Tech. L. Rev.*, vol. 12, p. 272, 2011.

[4] C. Breazeal, K. Dautenhahn, and T. Kanda, "Social robotics," *Springer handbook of robotics*, pp. 1935–1972, 2016.

[5] J. T. Licardo, M. Domjan, and T. Orehovački, "Intelligent robotics—a systematic review of emerging technologies and trends," *Electronics*, vol. 13, no. 3, p. 542, 2024.

[6] G. C. Maffettone, L. Liguori, E. Palermo, M. di Bernardo, and M. Porfiri, "Mixed reality environment and high-dimensional continuification control for swarm robotics," *IEEE Transactions on Control Systems Technology*, 2024.

[7] Y. Patel and P. H. Rughani, "Unmasking the vulnerabilities: A deep dive into the security threat landscape of humanoid robots," in *The International Conference on Recent Innovations in Computing*. Springer, 2023, pp. 273–289.

[8] Y. Patel, P. H Rughani, and T. Kumar Maiti, "An examination of the security architecture and vulnerability exploitation of the turtlebot3 robotic system," *International Journal of Computing and Digital Systems*, vol. 16, no. 1, pp. 1593–1602, 2024.

[9] Y. Patel and P. H. Rughani, "qcros2: An architecture to secure network communication of ros2-based environment," in *2023 8th International Conference on Robotics and Automation Engineering (ICRAE)*. IEEE, 2023, pp. 253–258.

[10] I. Abeykoon and X. Feng, "Challenges in ros forensics," in *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2019, pp. 1677–1682.

[11] Y. Patel, P. H. Rughani, and D. Desai, "Network forensic investigation of collaborative robots: A case study," in *2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR)*. IEEE, 2022, pp. 51–54.

[12] Y. Patel and P. H. Rughani, "Are we ready to investigate robots? issues and challenges involved in robotic forensics," in *The International Conference on Recent Innovations in Computing*. Springer, 2023, pp. 259–271.

[13] V. Fernando, "Cyber forensics tools: A review on mechanism and emerging challenges," in *2021 11th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. IEEE, 2021, pp. 1–7.

[14] A. Asquith and G. Horsman, “Let the robots do it!–taking a look at robotic process automation and its potential application in digital forensics,” *Forensic Science International: Reports*, vol. 1, p. 100007, 2019.

[15] N. Nelufule, T. Singano, K. Masemola, D. Shadung, B. Nkwe, and J. Mokoena, “An adaptive digital forensic framework for the evolving digital landscape in industry 4.0 and 5.0,” in *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*. IEEE, 2024, pp. 1686–1693.

[16] G. Vaciago and F. Bosco, “New challenges in robotics. cyber security and digital forensics,” *Informatica e diritto*, vol. 23, no. 2, pp. 9–20, 2014.

[17] V. M. Vilches, L. A. Kirschgens, E. Gil-Uriarte, A. Hernández, and B. Dieber, “Volatile memory forensics for the robot operating system,” *arXiv preprint arXiv:1812.09492*, 2018.

[18] M. M. Basheer and A. Varol, “An overview of robot operating system forensics,” in *2019 1st International Informatics and Software Engineering Conference (UBMYK)*. IEEE, 2019, pp. 1–4.

[19] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, “Experimental security analysis of a modern automobile,” in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 447–462.

[20] E. Casey, *Digital evidence and computer crime: Forensic science, computers, and the internet*. Academic press, 2011.

[21] E. Khalastchi and M. Kalech, “Fault detection and diagnosis in multi-robot systems: A survey,” *Sensors*, vol. 19, no. 18, p. 4019, 2019.

[22] I. Abeykoon and X. Feng, “A forensic investigation of the robot operating system,” in *2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2017, pp. 851–857.

[23] R. Amsters and P. Slaets, “Turtlebot 3 as a robotics education platform,” in *Robotics in Education: Current Research and Innovations 10*. Springer, 2020, pp. 170–181.

[24] M. A. Hossen, S. Kharade, B. Schmerl, J. Cámara, J. M. O’Kane, E. C. Czaplinski, K. A. Dzurilla, D. Garlan, and P. Jamshidi, “Care: Finding root causes of configuration issues in highly-configurable robots,” *IEEE Robotics and Automation Letters*, 2023.

[25] Y. Patel and P. Rughani, “Roboshell tool: A robot bash session,” *The Copyright Office Journal*, no. 27, 16368/2023-CO/SW, 2023.

[26] ——, “Rufo (robot utility forensics) framework,” *The Copyright Office Journal*, no. 27, 16374/2023-CO/SW, 2023.

Yash Patel is pursuing a Ph.D. in Computer Science and Technology at the National Forensic Sciences University, Gandhinagar, Gujarat, India. His research thrust areas include cybersecurity, digital forensics, robotics security, and robotics forensic investigation.

Parag Rughani, Ph.D. is currently working as an professor in digital forensics at the National Forensic Sciences University, India. He is IEEE Senior Member and has published more than 25 articles in reputed journals and conferences. His research thrust areas include machine learning, computer forensics, memory forensics and malware analysis.