Exploring Resilient Operation of Multi-Robot Fleet in Various Attack Scenarios

Athanasios Papanikolaou, Iason Sotiropoulos, Stamatia Rizou, Francisco Fraile, Raquel Julia Ros, Nacim Ramdani

Exploring Resilient Operation of Multi-Robot Fleet in Various Attack Scenarios*

Athanasios Papanikolaou[1], Iason Sotiropoulos[1], Stamatia Rizou[1], Francisco Fraile[2], Raquel Julia[3] and Nacim Ramdani[4]

***Abstract*— In the era of autonomous robotics and the proliferation of robot fleets, the reliability of these systems in the face of potential disruptions becomes a critical consideration. This paper focuses on examining the resilience of robotic fleets, specifically at the communication layer. Through a systematic exploration of various communication-based attack scenarios, the study seeks to unravel vulnerabilities and challenges in maintaining seamless communication within the fleet. The findings aim to contribute insights into fortifying the communication infrastructure of robotic fleets, thereby enhancing their overall resilience in dynamic and potentially adversarial environments. This research aims towards advancing the robustness of autonomous systems and ensuring their dependable operation across diverse applications.**

multi-robot fleet, resilience, MQTT, cybersecurity

I. Introduction

In the era of autonomous robotics, multi-robot fleets have emerged as a major technological advancement, demonstrating their prowess in a multitude of applications ranging from disaster response to environmental monitoring [6], [2]. Robotic fleets composed of diverse agents working collaboratively can potentially revolutionize industries and provide innovative solutions to complex real-world challenges. However, this transformative potential also brings to the fore an array of security concerns [5], as the reliance on interconnected robots introduces vulnerabilities that malicious actors may exploit [16]. In this dynamic landscape, where autonomy and interconnectivity are major factors, resilience becomes paramount. While maintaining service with reduced performance degradation in presence of disturbance, is a robustness feature, the concept of resilience encompasses the ability of a system to adapt and recover swiftly from adversity while maintaining core functionality [10], [14], [13]. In this context, the concept of anti-fragility beyond resilience has also been introduced in the sense that anti-fragile systems may gain some performance when they are exposed to adversity [12], [15].

This study aims to investigate how multi-robot fleets handle different types of attacks. Towards this, challenges, vulnerabilities, and strategies to improve resilience against cyber-threats are explored. More precisely, this paper investigates online task re-allocation and re-scheduling, as means of augmenting a multi-robot fleet management system with resilience mechanisms, and sets out to evaluate the robustness of such a fleet against a range of cyber-attacks. The idea at the core of the designed resilience mechanism is to recompute the multi-robot task allocation and scheduling solution while considering the current state of the fleet, when unexpected events that degrade some performance indicators occur. While online re-allocation strategy has been investigated with human-multi-robot fleet in precision agriculture, our study focuses on the impact of cyber-attacks on the overall system to account for the possibility of changing human parameters over time [9]. Resilience is a topic that is widely investigated in the literature in different contexts. In this paper, resilience is discussed in the context of the communication layer when it comes to vulnerabilities. Our work contributes to the existing literature by focusing on the impact of such vulnerabilities on a robotic fleet and provides simulation results, including mathematical modeling of the associated optimization problem and re-planning in case of perturbation. The experiment conducted in this research represents a scenario of a medical robot fleet operating inside a hospital. Since medical robots undertake tasks of possibly vital importance, ensuring the security of their functions and resilience to disturbances needs to be prioritized.

In the present research, the Fleet Management System (FMS) uses the Message Queuing Telemetry Transport (MQTT) protocol as the system message bus, enabling the exchange of information among robotic agents and the integration of centralized planning and coordination functions in the FMS. While MQTT's lightweight design and low overhead offer significant advantages for IoT applications, it also exposes potential security vulnerabilities, making it an attractive target for cyber attackers [8]. Thus, we identify the use of MQTT as a major vulnerability of our fleet. The cyber-attacks that are used aim at exploiting this vulnerability. We employ various cyber-attacks against our use of the MQTT protocol in order to record which of the explored attack vectors have a larger impact towards such a system. We aim to identify: (1) where is our system more vulnerable, and (2) possible configurations to avoid potential failure.

The paper is organized as follows: First, a literature review on MQTT security aspects is gathered in Sect. II, then the mathematical modeling of multi-robot task allocation and scheduling is presented in Sect. III. Following that, we briefly

*This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101007673

[1]A. Papanikolau, I. Sotiropoulos, and S. Rizou are with SingularLogic, 14564 Nea Kifisia, Greece (email: apapanikolaou@singularlogic.eu)

[2]F. Fraile is with Universitat Politècnica de València, CIGIP, 46022 Valencia, Spain

[3]R. Julia RD is with Robotnik Automation, 46988 Paterna, Spain

[4]N. Ramdani is with Univ. Orléans, INSA CVL, PRISME UR 4229, F45072 Orléans, France

present a list of attack vectors in Sect. IV, and lastly in Sects. V and VI we present the experimentation set-up and then the results of our study as well as details on how the experiments were conducted. Concluding remarks are gathered in Sect. VII.

II. Related Work

The field of MQTT security has received significant attention from researchers and cybersecurity professionals. Numerous studies have explored potential vulnerabilities and assessed the efficacy of various attacks on MQTT-based systems. In [8] an extensive review of the vulnerabilities, attack vectors, and potential solutions associated with the MQTT (Message Queuing Telemetry Transport) protocol in the context of the (IoT) is provided. Similarly, [4] focused on already existing security solutions while providing a novel and deeply technical approach to achieving confidentiality and integrity of messages. Several works have focused on denial of service attacks in MQTT. Specifically, [11] provided an in-depth investigation of two classes of DoS attacks as well as some security considerations towards mitigating those attacks. While demonstrating the exploitation of a particular vulnerability, [17] introduces an innovative attack method to authenticate the feasibility of the proposed exploit. Lastly, [3], [7] dealt with detection of DoS attacks and consequently their prevention. On the other hand, several works focus on Man-in-the-Middle attacks in MQTT, as well as the consequences that successful attacks of this nature may have on IoT networks. These works [18] also utilize various methods for message generation, after the attacker has successfully accessed a network.

III. Multi-robot task allocation and scheduling

In this section we describe the mathematical modeling of the multi-robot task allocation and scheduling problem as well as the resilience mechanism.

A fleet of m autonomous robots defined in $R = \{1,...,m\}$ must accomplish n tasks defined in $T = \{1,...,n\}$. Each robot can physically move between any pair of nodes (i.e. arcs) (i, j) in a weighted complete directed graph $G = (V,A)$ such as $i, j \in A$ and $(i, j) \in V$. Let $C = (c_{ij})$ be a cost (distance) matrix associated with A. Any robot $r \in R$ starts its route from the robot charging node and must return to this node at the end of the route, such as the route does not contain any sub-tour. The path described by each robot is then a circuit.

Such a formulation corresponds to a well-known problem in the literature called the multiple Traveling Salesman Problem [1]. However, our variant must take into account time constraints such as robots must accomplish each task within a specific time window defined beforehand.

Let E and L be respectively sets of earliest and latest dates of execution for each task $t \in T$. Let Ω be the set of all service time durations associated to any given task $t \in T$. Let S be the average speed of any given robot $r \in R$. To fully satisfy a task (i.e. with no delay), a robot must complete the execution of the task within its allocated time window. In the sequel, we propose a Mixed Integer Linear Programming (MILP) formulation of our multi-robot task allocation problem.

a) Notation: Let us define, m the number of robots, n the number of tasks, $R = \{1,...,m\}$ the set of robots, $T = \{m+1,...,m+n\}$ the set of tasks, $A = \{1,...,m+n\}$ the set of all nodes (robot charging nodes and task nodes), V the set of arcs (i, j), $G = (V,A)$ the complete directed graph, $C = \{c_{ij}\}$ the distance cost matrix, E the set of earliest dates for tasks, L the set of latest dates for tasks, $[e_t; l_t]$ the time window for task $t \in T$, Ω set of all service time durations for tasks, $S = \{s_m\}$ the set of robot average velocities, $\overline{m}$ a very large number used to linearize quadratic constraints.

b) Decision variables: Let the decision variables be

$$\forall i, j \in A, x_{ij} = \begin{cases} 1, & \text{if arc } (i, j) \text{ is used by any robot,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

where f_i is the arrival date of robot on node $i \in A$, and d_i the departure date of robot from node i.

c) Objective function: Let the objective function be

$$\min(\sum_{i \in R} f_i) \tag{2}$$

The objective function (2) minimizes the sum of return dates of the robots to their respective charging node.

d) Constraints: Let the constraints be

$$x_{ii} = 0 \qquad \forall i \in A \tag{3}$$

$$\sum_{j \in T} x_{rj} \leq 1 \qquad \forall r \in R \tag{4}$$

$$\sum_{i \in T} x_{ir} \leq 1 \qquad \forall r \in R \tag{5}$$

$$x_{rr+1} = 1 \qquad \forall r \in \{1,...,m-1\} \tag{6}$$

$$\sum_{r' \in R \setminus \{r+1\}} x_{rr'} = 0 \qquad \forall r \in \{1,...,m-1\} \tag{7}$$

Equation (3) forbids robots to use arcs located in matrix diagonal. Equations (4) (resp. (5)) allow for each robot only one departure from (resp. one arrival to) its charging node, ensuring only one circuit per robot is allowed. Equations (6) forces a virtual move from any given robot r to the next robot $r+1$ and (7) forbids the way back (i.e from any robot r to a previous one). These two equations are later used to track route time while keeping a light mathematical formulation with 2-indexes binary decision variable x and single-index continuous decision variables f and d.

$$\sum_{t \in T} x_{r-1t} \geq \sum_{t \in T} x_{rt} \qquad \forall r \in \{2,...,m\} \tag{8}$$

$$\sum_{t \in T} x_{tr-1} \geq \sum_{t \in T} x_{tr} \qquad \forall r \in \{2,...,m\} \tag{9}$$

Constraints (8) (9) ensure that if any given robot r is used to satisfy a task, then robot $r-1$ is also in use, assuming r is not the first robot. Note that equations (6) to (9) only apply

if problem instances contain more than one robot (i.e $m > 1$)

$$\sum_{i \in A} x_{it} = 1 \qquad \forall t \in T \tag{10}$$

$$\sum_{j \in A} x_{tj} = 1 \qquad \forall t \in T \tag{11}$$

$$\sum_{h \in A} x_{ht} = \sum_{j \in A} x_{tj} \qquad \forall t \in T \tag{12}$$

Equation (10) (11) are task satisfaction constraints: each task node must be visited once by robots. Flow conservation is ensured by constraint (12). However, this constraint isn't enough to eliminate sub-tours by itself. This is not an issue since temporal continuity constraints introduced later prevent any sub-tour possibility.

$$d_t \geq f_t + \omega_t \qquad \omega_t \in \Omega, \forall t \in T \tag{13}$$

$$f_j + \overline{m} \cdot (1 - x_{tj}) \geq d_t + \frac{c_{tj}}{s_r} - \overline{m} \cdot (1 - x_{tj}), \quad \forall t \in T, \forall j \in A \tag{14}$$

$$d_r \leq f_t - \frac{c_{rt}}{s_r} \cdot x_{rt} \qquad \forall r \in R, \forall t \in T \tag{15}$$

$$f_t \geq e_t \qquad e_t \in E, \forall t \in T \tag{16}$$

$$d_t \leq l_t \qquad l_t \in L, \forall t \in T \tag{17}$$

Temporal continuity conservation is ensured by (13) and (14) on tasks, while constraint (15) initializes the departure date for all robots. Adding these 3 constraints ensure that all sub-tours are eliminated since temporal decision variables can only increase according to the robot path. A sub-tour would violate these constraints. Finally, constraints (16) and (17) are the time window constraint on tasks. Note that robots are in fact allowed to come earlier than the earliest date for any given task $t \in T$ since the value taken by decision variable f_t actually corresponds to the starting date for the execution of task t.

A. Resilience mechanism

Resilience requires detection of performance degradation, followed by adaptation. Our approach relies on a performance indicator that aggregates two performance indicators: the delay on task completion, computed as the difference between the actual date and the planned date, and the delay on robot motion on the prescribed path.

1) Delay on task completion: This indicator is computed only when the task expected completion date is reached. Let us define T^k_{actual} the actual completion date, T^k_{plan} the planned completion date, and t the current date. The delay of robot m on completing task k is then given by

$$\begin{aligned} &\text{if } \exists T_{actual,k} \quad \text{then} \quad \tau_m = T_{plan,k} - T_{actual,k} \\ &\qquad\qquad\quad \text{else} \quad \tau_m = T_{plan,k} - t \end{aligned} \tag{18}$$

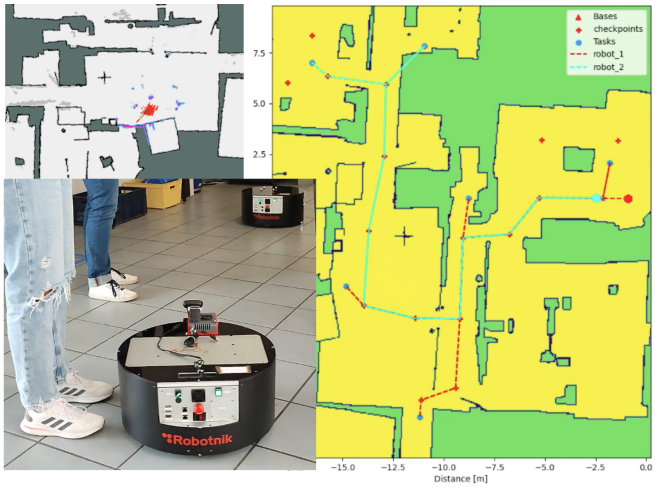

Fig. 1. Implementation example. The service points (blue) and waypoints (red). The optimal plans calculated by the optimal task planner : robot 1 in red, robot 2 in blue. The robot fleet operation in space shared with humans. Sensors visualisation from Robot 2 during the experiment

2) Robot motion delay: This performance indicator aims to detect delays while the robot is moving from one service point to another. It aims to anticipate delays on task completion assuming the robot may not be able to catch up delays accumulated when the robot moves from one service point to another. The robots advertise their location to the fleet management system periodically. Assuming a constant robot velocity, one can then compute, for each location, the planned time of arrival and the actual time of arrival. Hence, a continuous motion delay of robot m is given by

$$\sigma_{m,k} = t - \hat{t}_m \tag{19}$$

where the robot actual motion $\hat{t}_m = T_k + p(k,m)/s_m$ is obtained by projecting the actual robot position onto the planned graph. Let A_k and A_{k+1} be waypoints, the two end-nodes of the segment constituting an arc on the directed graph G, M_m robot actual position, s_m robot velocity, and T_k the time when last node was reached. One has: $p(k,m) = \cos\alpha\,|M_m A_s|$, $\alpha = \widehat{A_{k+1} M_m A_k}$. The actual robot motion does not follow precisely the planned path. We use projection of robot motion on graph arcs to compute motion delay on the graph.

3) Filtered performance criteria: To filter out disturbances of short duration, or deviations that can easily be corrected by robot embedded navigation and motion control algorithms, the two performance indicators are filtered by a first order integrator, tuned by the user. A re-planning at time $t_{k^\star_m}$ is triggered when the following condition is met:

$$k^\star_{m_k} = \min\{k \,|\, \exists m_k, \, ((\tau_{m,k} > \varepsilon_\tau) \vee (\sigma_{m,k} > \varepsilon_\sigma))\} \tag{20}$$

where thresholds ε_τ and ε_σ are tuned by the user. Robot m_{k^*} which solves Eq.(20) is deemed not operational and is removed from the set of robots R to be used for task allocation.

IV. Attack Vectors

The multi-robot optimal task assignment and scheduling (MRTA) described in Sect. III is implemented as a centralized management system. The result of the MRTA is given as a queue of basic tasks (e.g., go to a specific location) that each robot needs to complete. With the fleet of robots endowed with a centralised management system, for inter-robot communication we use the MQTT protocol: the MQTT broker is used to dispatch the queue of tasks to each robot and receive status updates from the fleet. The autonomous mobile robots are controlled with the open-source Robot Operating System (ROS). A newly developed Command Manager executable allows the user to send on the ROS network, the sequence of complex command translating the queue of optimally scheduled tasks, using simple string-encoded messages and read the feedbacks. ROS topics are used for node communication and for publishing and reading command messages and feedback. A dedicated ROS-MQTT bridge node provides functionality to the bidirectional bridge between the MQTT and ROS messages, hence the MQTT broker and the ROS network. Fig. 1 shows the actual implementation with Robotnik's RB1 mobile robots evolving in an environment shared with humans.

In the sequel, we embark on a comprehensive exploration of the multifaceted attack vectors utilized by cyber adversaries to infiltrate, compromise, and exploit digital systems, more specifically those systems that make use of the MQTT protocol. Drawing on the research and insights from the reviewed literature, we have identified the main attack vectors that threaten our system. It is important to note that given the ever-evolving nature of cyberattacks, it is highly possible that we may witness an increase in the frequency and sophistication of attacks in the future, therefore we must acknowledge that the examined attack vectors are subject to potential updates, considering the dynamic nature of the subject matter. We classify the attack vectors into two overarching categories for better organization and understanding: Denial of Service and Man in the Middle attacks.

A. Denial of Service

A Denial of Service attack (DoS attack) is a cyber-attack in which the perpetrator seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting the services of a host connected to a network. In our particular setting, all DoS attacks are aimed towards the message broker, and by disabling its function one can potentially disable the entire system. There are various DoS attacks in the available literature, here we deal with those that are potentially able to shut down or considerably slow the function of the broker, and for this purpose, we can further classify them into the following categories:

a) Connection flood: Connection flood is a Denial of Service (DoS) attack that targets a system by overwhelming it with a very high number of connection requests. This attack is designed to exhaust the resources of the target system, rendering it incapable of responding to legitimate users' requests. This can result in slow communication or complete disruption of the MQTT network, affecting all connected devices.

b) Memory exhaustion: A memory exhaustion attack is a form of cyber attack that aims to overwhelm a target system's available memory resources, ultimately causing the system to become slow, unresponsive, or even crash. This type of attack leverages vulnerabilities within a system's memory management to consume its memory resources at an unsustainable rate. In the context of MQTT, a memory exhaustion attack involves an attacker sending several messages to the broker or the targeted devices. These messages are often crafted to exploit memory-related vulnerabilities, such as buffer overflows or memory leaks. As the system processes these malicious messages, it allocates memory resources to handle them. However, if the messages are designed to consume more memory than the amount which the system can handle, the attack leads to a rapid depletion of available memory.

c) CPU exhaustion: A CPU exhaustion attack is a type of cyber attack that aims to deplete a system's central processing unit (CPU) resources, similarly to memory exhaustion. Within the context of MQTT protocol, a CPU exhaustion attack involves an attacker exploiting vulnerabilities in the MQTT broker or the connected devices to generate a high volume of computationally intensive operations. These operations can include complex calculations, data processing, or cryptographic operations that demand significant CPU resources.

B. Man in the Middle

Man-in-the-Middle (MitM) attacks, also known as hijack attack, are a cyber-attack where an attacker intercepts a network connection to alter the data being transferred between the two ends. A MitM attack involves cyber intrusion that exploits vulnerabilities within the communication channels between two parties. The primary intent of such attacks is to intercept, manipulate, or eavesdrop on the exchanged information, while maintaining the facade of seamless communication between the legitimate parties involved. In our scenario, an adversary positions themselves between the clients and the broker, intercepting and possibly altering the MQTT messages exchanged between them. This type of attack aims to breach the confidentiality, integrity, and authenticity of these messages, potentially leading to unauthorized access, data manipulation, or even complete service disruption.

a) Attack Process: The attacker positions themselves as an intermediary between the MQTT clients and the broker. This can be achieved through various means, such as ARP spoofing, DNS spoofing, or by exploiting insecure Wi-Fi connections. As the MQTT messages flow between the clients and the broker, the attacker intercepts the messages and accesses their contnet without the legitimate parties being aware of the intrusion. In some cases, the attacker might choose to alter the intercepted MQTT messages before forwarding them to the intended recipient.The attacker might store the intercepted messages and replay them at a later

time. This can lead to repeated execution of commands, unauthorized data retrieval, or even system disruption.

b) Impact: The impact of a MitM attack on a Mosquitto broker can be severe and wide-ranging: Sensitive data exchanged between clients and the broker can be exposed, leading to privacy violations and unauthorized access to confidential information. By intercepting authentication credentials, the attacker can gain unauthorized access to the MQTT broker, potentially compromising the entire system's security. Altering the contents of MQTT messages can lead to misinformation, unauthorized control of devices, or erroneous system behavior. If the attacker manipulates message flow or deliberately disrupts communication, it can lead to service outages, impacting the functionality of IoT devices relying on the MQTT broker.

V. Experimentation Set-up

For our experimentation we employed a fleet of $m = 4$ autonomous robots that must accomplish $n = 12$ tasks. We modeled the area of our expirementation as a graph G of a total of 36 nodes.

Robotic communications are facilitated using the MQTT protocol. Typically, the MQTT protocol defines two types of network entities: a message broker and several clients. Our set-up is based on this model and our robots come equipped with MQTT clients. Additionally, we consider a central client that is responsible for task allocation (hereby referred to as the fleet management system). Task allocation is based on the mathematical model presented in Sect.III. All these different clients can communicate via a central message broker that transfers all communication messages after correctly formatting them. This broker is the focus point of our attacks.

During our experimentation, we consider several assumptions. Firstly, we assume that the path planning and task allocation model is optimal and in a sense that it always yields accurate results. Furthermore, we assume that any perpetretor has knowledge of our system architecture. In addition, we consider that any such threat has already bypassed most security measures, including possible passwords and encryption. Thus, we are examining how our fleet reacts to an attack that has already passed most of its defenses. The simulation portrays the robots' movement as they engage with and execute each task. To maintain simplicity, we allocate the same time window for every task, and these tasks are distributed across various nodes of the graph. Before initiating the simulation, we assign each task to a specific robot and chart a predetermined path for each robot to follow. These paths have been calculated using the CPLEX optimizer. Throughout the simulation runtime, we introduce multiple attack scenarios, while documenting the count of failed tasks. In the context of building a resilient system, our objective is to minimize the cumulative count of task failures.

VI. Validation

For validation purposes, we conducted a series of simulation-based experiments using attack scenarios representative of potential real-world challenges. The simulation environment was custom-made, created specifically for this study, demonstrating a high degree of relevance and fidelity to real-world conditions, while ensuring a safe and controlled environment. We evaluated the fleet's performance using the mission success rate and task delay time key metrics. These metrics are integral to assessing the overall resilience of the fleet in the face of adversarial challenges. All the above was achieved by deliberately introducing the previously discussed attack scenarios during the simulation runtime. We proceeded to documenting the completion time for each task and the mission success rate (i.e., how many of the missions tasks were complete within the allocated time frame). In order to ensure transparency of the results, we monitored our robots' performance during the simulation runtime without any interference. We noticed that all tasks were completed within the time frame.

The first step towards testing resiliency, was using a slow DoS attack, specifically Slowite [17], on our mosquito broker. We observe that if the simulation has already started (i.e., our robots have already established their connection towards the broker), then the tasks are successfully completed. However, if the attack has launched successfully, our robots are unable to connect to the broker, thus failing at completing their allocated tasks. Another important observation is that should we configure our broker to keep connections alive for a small time (a method to protect against slow DoS attacks), we might cause our robots to disconnect while the attack persist uninterrupted.

Continuing our experimentation with denial of service, we attempt to saturate the broker's memory and CPU resources. Towards that end, we implemented two simple attacks that function quite similarly. The first attack aims to overload the broker's memory by sending multiple messages with the maximum payload size. We immediately notice that once we start sending such messages, all the robots freeze, and delays increase significantly. In fact, several replans are launched in vain as the robots do not regain functionality. We can therefore surmise that our system is vulnerable to such an attack. However, we can configure the mosquitto broker to drop connections if a certain payload size is exceeded. We calculate a 16.6% mission success.

The second attack focuses on exploiting the broker's CPU limitations. In order to achieve that, we sent numerous requests to our broker. The messages require computationally expensive operations, and the system becomes severely slower. Visually, we confirm this by noticing a considerable decrease in the speed of the fleet. Additionally, we notice a gradual increase in our measured delays. Due to these slowdowns, we calculated a success rate of 50%. All of DOS based attacks are visually presented in Fig. 2. Continuing our exploration, we turned focus towards our system's response against MitM attacks. In order to begin our test, we used spoofing and packet capturing software (Wireshark and BetterCAP). We were able to capture all messages of our MQTT application and easily gain access to their payload. Thus, monitoring traffic of our robotic fleet is achieved. By this

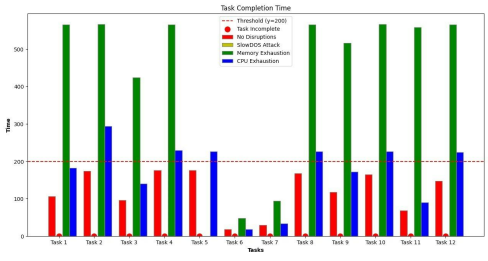

Fig. 2. Task Completion Time in DoS Scenarios

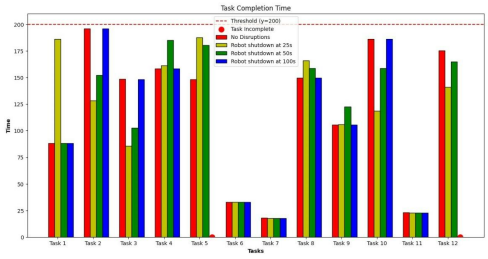

Fig. 3. Task Completion Time in MitM Scenarios

method, a perpetrator can discern a plethora of information such as the location, direction, and movement of the robots, and the time and place of completed tasks.

Additionally, we were able to intercept and modify messeges between the clients and the broker. After this was achieved, acting as the perpetrator, we could publish malicious messages to our broker, causing robotic failure. To delve deeper into the attack scenario, we supposed the perpetrator dispatches a meticulously crafted message capable of rendering a robot inactive. In this context, we scrutinize how our system responds to the deactivation of a particular robot at various points within the simulation.

To begin with, our first malicious message was sent at 25 seconds of the simulation runtime. Our fleet launched a replan and managed to complete all the tasks in time, thus leading to 100% success rate.

The identical situation unfolded when the attack occurred precisely at the 50-second mark during the simulation, mirroring the outcomes and circumstances observed in the previously described scenario. One the other hand, once the robot malfunctioned at 100 seconds, a replan was launched. However, the optimizer could not find a solution for all tasks to be completed on time, and task re-allocation did not occur. As a result, the tasks handled by the dowed robot remained incomplete. Here, we report 83.33% success rate (see Fig. 3).

VII. Concluding Remarks and Future Work

While our study has demonstrated promising results in fortifying multi-robot fleets against cyber-attacks, it is essential to acknowledge the potential vulnerabilities that persist, particularly in centralized systems. Should a centralized attack occur, our findings highlight alarming vulnerabilities that necessitate further investigation and refinement of our resilience mechanisms. Moving forward, addressing these challenges will be crucial in ensuring the robustness and security of multi-robot fleet management systems in real-world applications. Moving forward, our research will extend to testing with real robots to validate the efficacy of our proposed resilience strategies in practical scenarios. Additionally, we plan to explore non-centralized implementations to assess their feasibility and performance in enhancing the system's resilience against cyber-attacks.

References

[1] T. Bektas. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34(3):209–219, june 2006.

[2] M. Carpentiero, L. Gugliermetti, M. Sabatini, and G. B. Palmerini. A swarm of wheeled and aerial robots for environmental monitoring. In *2017 IEEE 14th ICNSC*, pages 90–95, 2017.

[3] D. Dikii, S. Arustamov, and A. Grishentsev. Dos attacks detection in mqtt networks. *Indonesian Journal of Electrical Engineering and Computer Science*, 21:601, 01 2021.

[4] D. Dinculeană and X. Cheng. Vulnerabilities and limitations of mqtt protocol used between iot devices. *Applied Sciences*, 9:848, 02 2019.

[5] V. Dutta and T. Zielińska. Cybersecurity of robotic systems: Leading challenges and robotic system design methodology. *Electronics*, 10(22), 2021.

[6] M. A. Gutiérrez, S. Nair, R. E. Banchs, L. F. D'Haro Enriquez, A. I. Niculescu, and A. Vijayalingam. Multi-robot collaborative platforms for humanitarian relief actions. In *2015 IEEE R10-HTC*, pages 1–6, 2015.

[7] A. P. Haripriya and K. Kulothungan. Secure-mqtt: an efficient fuzzy logic-based approach to detect dos attack in mqtt protocol for internet of things. *EURASIP Journal on Wireless Communications and Networking*, 2019, 04 2019.

[8] A. J. Hintaw, S. Manickam, M. F. Aboalmaaly, and S. Karuppayah. MQTT vulnerabilities, attack vectors and solutions in the internet of things (IoT). *IETE Journal of Research*, 69(6):3368–3397, May 2021.

[9] M. Lippi, J. Gallou, A. Gasparri, and A. Marino. An optimal allocation and scheduling method in human-multi-robot precision agriculture settings. In *2023 31st Mediterranean Conference on Control and Automation (MED)*, pages 541–546, 2023.

[10] S. Mayya, D. S. D'antonio, D. Saldana, and V. Kumar. Resilient task allocation in heterogeneous multi-robot systems. *IEEE Robotics and Automation Letters*, 6(2):1327–1334, Apr. 2021.

[11] U. Morelli, I. Vaccari, S. Ranise, and E. Cambiaso. Dos attacks in available mqtt implementations: Investigating the impact on brokers and devices, and supported anti-dos protections. pages 1–9, 08 2021.

[12] A. Munoz, J. Billsberry, and V. Ambrosini. Resilience, robustness, and antifragility: Towards an appreciation of distinct organizational responses to adversity. *International Journal of Management Reviews*, 24(2):181–187, Jan. 2022.

[13] G. Neville, S. Chernova, and H. Ravichandar. D-ITAGS: A dynamic interleaved approach to resilient task allocation, scheduling, and motion planning. *IEEE Robotics and Automation Letters*, 8(2):1037–1044, Feb. 2023.

[14] G. Notomista, S. Mayya, Y. Emam, C. Kroninger, A. Bohannon, S. Hutchinson, and M. Egerstedt. A resilient and energy-aware task allocation framework for heterogeneous multirobot systems. *IEEE Transactions on Robotics*, 38(1):159–179, Feb. 2022.

[15] N. N. Taleb and R. Douady. Mathematical definition, mapping, and detection of (anti)fragility. *Quantitative Finance*, 13(11):1677–1689, Nov. 2013.

[16] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat. A critical cybersecurity analysis and future research directions for the internet of things: A comprehensive review. *Sensors*, 23(8), 2023.

[17] I. Vaccari, M. Aiello, and E. Cambiaso. Slowite, a novel denial of service attack affecting mqtt. *Sensors*, 20:2932, 05 2020.

[18] H. Wong and T. Luo. Man-in-the-middle attacks on mqtt-based iot using bert based adversarial message generation. 08 2020.