TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Diego del Rio Manzanas 223594IVCM

Securing Remote Connectivity: A Systematic Penetration Testing Analysis of a Telepresence Robot

Master's thesis

Supervisor: Shaymaa Mamdouh Khalil
MSc

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Diego del Rio Manzanas 223594IVCM

Turvalise kaugühenduse loomine: Kaugosalusroboti süstemaatiline läbistustesti analüüs

Magistritöö

Juhendaja: Shaymaa Mamdouh Khalil
MSc

Tallinn 2024

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Diego del Rio Manzanas

10.05.2024

Abstract

Telepresence robots represent a technological advancement with transformative potential across multiple sectors, facilitating remote communication and interaction. However, their integration into various environments introduces inherent security vulnerabilities that must be carefully addressed to mitigate risks and protect sensitive data. This thesis presents a systematic black-box penetration testing methodology for a comprehensive security analysis of the cyber security of a telepresence robot. The proposed methodology is based on PatrIoT for the test framework and uses the STRIDE threat model. The four-phase process gathered information about the telepresence robot and used it to generate a data flow diagram representing the whole system. Potential threats were derived based on STRIDE for each component of the diagram. Then, during the planning phase, these threats were analysed to generate attack tests that attempted to exploit the robot's weaknesses. Based on the success of the attack tests, new ones were generated in several iterations, to ensure that the entire attack surface was covered. Once all the tests had been run, a report was generated with the results of the penetration testing process. Technical details of successful and unsuccessful tests were documented, along with potential fixes that could be applied to strengthen the security of the telepresence robot and its configuration. Specific details are omitted in this thesis to protect the robot's manufacturer's privacy and avoid legal issues. Different vulnerabilities and misconfigurations were found, which allowed to take control of the robot and get information from users and administrators that connect to it. Based on those discovered weaknesses and the proposed improvements, this study contributes to a better understanding of cybersecurity issues affecting telepresence robots. It also provides valuable information that can help manufacturers and owners not only of the robot studied but also of other products with similar characteristics.

This thesis is written in English and is 84 pages long, including 9 chapters, 8 figures and 12 tables.

List of abbreviations and terms

IT	Information Technology
IoT	Internet of Things
ADB	Android Debug Bridge
SSH	Secure Shell
TCP	Transmission Control Protocol
SDK	Software Development Kit
DFD	Data-Flow Diagram
ARP	Address Resolution Protocol

Table of contents

List of figures

List of tables

1 Introduction

In an era of technological advancements and the proliferation of interconnected devices, telepresence robots have emerged as a groundbreaking solution to solve the challenges of physical location barriers. Telepresence robots provide the experience of being present in a particular location, allowing remote interaction to deliver an experience similar to the one the individual controlling the device would have if he were on-site. To accomplish that, the robots are equipped with audio and video capabilities as well as mobility. This hardware allows the user to see, hear and move through the environment [1]. This facilitates real-time presence by providing remote interaction and communication for users in geographically distant locations, blurring the line between virtual and physical spaces.

Telepresence robots have been used in areas like education [2], healthcare [3] or home applications [4]. Despite the advantages of telepresence robots, the combination of their hardware makes them capable of gathering sensitive information about their surroundings and integration in critical environments [5], raising concerns about their security and integrity. These security concerns can be addressed by performing penetration testing. It is defined as a set of activities that aim to identify and exploit security vulnerabilities, showing the effectiveness of the security measures implemented on a system [5]. The discovery of the threats and vulnerabilities affecting the system is the first step towards their mitigation and is key to ensuring awareness and achieving better security.

1.1 Research Problem

Although the usage of telepresence robots is still minimal, and they “have only been studied in controlled or small scale installations” [6], they are increasingly being used in real-world scenarios. Despite the growing popularity and interest, there is a significant lack of comprehensive studies investigating their cyber security.

This thesis aims to address this problem by actively identifying and analysing the security risks and vulnerabilities through penetration testing. For this topic, the following research questions are proposed:

RQ1: What are the potential cyber security threats affecting the telepresence robot?
RQ2: How can systematic penetration testing be adapted to the telepresence robot?
RQ3: What method can be used to choose the tests to be performed?
RQ4: What improvements can be made to enhance the cyber security of the telepresence robot based on the findings from the penetration tests?

The answer to these questions, combined with the systematic methodology of stating and analysing risks, would help improve the security of telepresence robots and, therefore, benefit their manufacturers and users.

1.2 Scope, Goals and Limitations

The primary goal of this thesis is identifying potential cyber security vulnerabilities within the selected telepresence robot. Through structured penetration testing, this research aims to evaluate the effectiveness of the existing cyber security measures implemented in the robot to mitigate cyber security threats. By performing this research, the threats affecting cybersecurity robots are also discovered, even if they are not associated with a vulnerability. This means that some of the potential threats discovered are not exploitable but are useful in shaping the cybersecurity risks of these devices. This contributes to a threat modelling of general telepresence robots with similar characteristics. Furthermore, the study seeks to propose and recommend improvements, based on the outcome of the penetration tests, to enhance the cyber security measures of telepresence robots. In summary, this research seeks to advance the understanding of cyber security issues related to telepresence robotics, provide actionable insights for improving the security of these systems, and offer valuable guidance to manufacturers and users.

This research focuses on conducting a comprehensive penetration testing analysis on a specific brand and model of telepresence robot available in the market. The scope of this study is restricted to the security assessment of the aforementioned robot. This

means it is unique based on its hardware configuration, software ecosystem, user configuration and integration with various environments.

To safeguard the security and privacy of the robot's manufacturer, the telepresence device's brand, model and main characteristics are omitted in this document. In addition, the attacks are limited to only affect the device locally. Even though the device has internet connectivity and relies heavily on the cloud, no attacks target it or any remote asset. This is to avoid legal disputes since the cloud provider hasn't explicitly agreed on testing their endpoint. Contact attempts to ask the manufacturer for collaboration were unsuccessful, as no response was received. Finally, it is important to note that the robot being analysed is currently being used during the development of this thesis. As a result, this research is limited by the need to avoid disrupting its service. Any attacks on the device must be reversible so that the robot can be restored to its default configuration and resume normal operation.

It is essential to note that the findings and outcomes of this research are specific to the selected robot model and characteristics and might differ when applied to different devices. All the aspects narrowing the scope result in the impossibility of generalising this research as it doesn't cover the full spectrum of threats and vulnerabilities affecting this particular robot. Also, replication on other devices must adapt the methodology to fit the robot's hardware, software, and configuration combination.

1.3 Novelty and Contribution

The novelty of this research lies in its focus on penetration testing of telepresence robots, a topic that has not been extensively explored in the cyber security field. There are studies on the cyber security of other IoT devices, including robots [7] and some of them address areas similar to those where telepresence robots are used, like medicine [8] or people care [9]. However, the unique functionalities and other usage scenarios of telepresence robots require a dedicated security investigation, which is the research gap covered by this thesis. This research employs penetration testing techniques tailored to telepresence robots' specific characteristics and hardware configuration, providing new insights into their potential vulnerabilities.

This research makes significant contributions to both academia and industry. For academia, it will enrich the body of knowledge in the cyber security field by shedding light on an unexplored area. The methodology and findings could serve as the foundation for future research in this domain. For the research robot industry, the findings could be instrumental in enhancing the security of their products. Even though it is likely that the vendor already applied a secure by design process and performed threat modelling, it is not public information. The provided threat modelling in this thesis might help telepresence robot vendors ensure the completeness of their threat lists and also guide them in using threat modelling for planning penetration testing in the future. In addition, with the findings of this research, we believe their developing teams can design more secure systems by understanding the potential vulnerabilities that affect their products. Companies would benefit from a decreased probability of cyber-attacks, directly impacting their financial losses and increasing reputation and user trust. Moreover, this research could also help users of telepresence robots by raising awareness about these devices' potential cyber security risks. This could encourage users to adopt safer practices and configurations when using these robots, also contributing to the overall security of these systems.

In conclusion, this research has the potential to significantly advance our understanding of cyber security issues related to telepresence robotics and contribute to the development of more secure systems.

This thesis is divided into nine chapters. This first chapter, Introduction, presents the topic, including the research problem, scope, goals, limitations, novelty and contribution. The second chapter, Background, includes the information needed for a better understanding of this thesis and the related work. The third chapter, Methodology, explains in detail the approach followed for the development of the work and the tools used to achieve it. Its application is shown in subsequent chapters, Reconnaissance, Threat Modelling, Exploitation and Reporting. They include the information gathering, threat elicitation, attack attempts and findings reporting respectively. The eighth chapter, Discussion, shows the interpretation of the results and their implications. Finally, the Summary chapter provides a concise overview of the research.

2 Background

“Robots have become incorporated into daily life over the last half century: what was once only science fiction has now become a reality.” [10]. Today, robots are widespread and help humans in repetitive or demanding tasks. The number of robots operating around the globe is estimated to be 3.5 million [11].

The term telepresence was first used in 1980 by Marvin Minsky [12], describing a human operator who can be physically present at a remote location thousands of miles away. According to his paper, the operator managed to do so through interaction with his actions and sensing feedback provided by teleoperation technology.

Based on that initial idea, telepresence robots are defined as “mobile robot platforms capable of providing two way audio and video communication” [13]. They can also be determined based on their capabilities, “embodied video conferencing on wheels” [14]. These days, people located far away from each other communicate through video chat applications such as Skype, WhatsApp or FaceTime. Telepresence is an attempt to make those online interactions as close as possible to their equivalent if they happened in the physical world.

The first company producing a telepresence robot did so for the health industry. They were also the first to introduce presence robots in hospitals [15]. Since then, several other manufacturers have been developing telepresence robots. Although they still don’t have great popularity, with their market valued at 334 million dollars, the rising demand from the healthcare industry is expected to increase their usage. According to an independent business report, this would raise the market value to 1.6 billion dollars in the next ten years [16].

Since their appearance, different iterations of telepresence robots have evolved with newer hardware features, making them capable of doing more tasks. Their hardware includes the following elements:

- Video capabilities: For capturing input video to visually sense the environment by the operator, most robots rely on cameras. Different models have used simple cameras, high resolution, motorised pan/tilt cameras or 360-degree video equipment for a full view of the surroundings [17]. More advanced telepresence robots combine the images gathered by the cameras with laser scanners for a more precise mapping of the physical space [18]. Most robots also feature a screen so that the people physically interacting with the robot can see the operator or any other image or video that wants to be displayed to the viewer.
- Audio capabilities: To provide two-way audio, telepresence robots feature a combination of microphones with loudspeakers. Microphones record the audio around the robot and deliver it to the operator. Then, the operator can reply through the robot's speakers so he can be heard by the people surrounding the machine. Advanced robots in the market use a microphone array that can localise audio sources, separate sounds, or suppress echo [19].
- Mobility: Almost all telepresence robots feature some mobility mechanism to control and change the physical location of the robot. Most mobility mechanisms include wheels, but a minority of them feature human-like walking via humanoid robots [20]. In combination with the video and surrounding sensing capabilities, most commercial robots feature obstacle detection and avoidance mechanisms [21]. Robots can be guided autonomously, semi-autonomously, or manually by the operator using a phone, a computer via keyboard and mouse, or specific controllers. Although it can change in the future due to high-tech hardware and the generalisation of artificial intelligence usage, most of the current robots are operated manually. Adapted hardware for people with disabilities has also been tested using eye trackers [22] or virtual reality headsets [23]. However, there are exceptions with robots that don't provide mobility mechanisms but instead are static, as shown in previous literature [24].

Telepresence robots are not only restricted to two individuals having a conversation, but they have been used in other more ambitious fields requiring a human presence in remote areas that are not easily accessible, like underwater [25], Antarctica [26] or Mars [27]. However, since those robots have unique, different characteristics from the ones available in the market to fulfil their particular environments, this study will not cover

them. Figure 1 shows examples of telepresence robots for a better visualisation of their characteristics and use cases.

Figure 1. Telepresence Robots [28].

2.1 Related work

Data and information security are two aspects that are highly concerning for companies today [29]. One widely used method of improving security in an organisation is penetration testing. It is described as “a structured process to test an organisation's computing base looking for vulnerabilities like system configuration, software and hardware errors, and its operational process in order to identify the weakness” [29].

This general definition can be applied to all aspects of an organisation. It is possible to perform penetration testing on IT systems but also on some physical assets, such as trying to break into an office using ordinary locksmith methods [30]. However, only IT-related penetration testing has been considered for this research. This includes software and hardware but excludes other methods, such as social engineering attacks.

Penetration testing usually tries to cover all the attack methods that apply to a specific system based on its hardware and software characteristics and its environment. This means that the attacks vary depending on the hardware installed, the wireless connectivity available, the software versions installed and the people or employees with access to the system.

According to a Penetration Testing guide published by Georgia Weidman [31], a typical penetration testing process can have the following phases:

1. Pre engagement: Since penetration testing techniques are intrusive, the attacker must agree with the system's owner on the aspects subject to attack. The scope of the test must be clear, but also the contact information if a critical vulnerability is discovered. Getting a signed authorisation stating that the attacks are allowed and limiting liability is also a good recommendation.
2. Information gathering: Information is critical for the success of penetration testing. This stage aims at collecting as much information about the target as possible. This includes public information, a process known as OSINT (Open Source INTelligence), and other tools to scan networks, such as port scanners.
3. Threat modelling: It is a set of techniques "used to model and analyse technology systems and services to understand better how that system or service might be attacked" [32]. This process leads to a set of threats that can potentially be attacked.
4. Vulnerability Analysis: This stage aims to actively discover vulnerabilities, loopholes that could "be utilized by attackers to launch attacks on technical assets" [33].
5. Exploitation: Based on the discovered vulnerabilities, exploits are run against the target in an attempt to access or modify the system's state. An example exploit is logging in with default passwords and gaining access to the attacked system.
6. Post exploitation: If the exploits were successful, this phase gathers information about the system from inside, trying to propagate along the network or escalate privileges. It is a phase when, once some damage is done, attempts to make things worse.
7. Reporting: This phase involves the documentation of the whole process and shows the findings of it. Any vulnerability and attack method must be clearly stated in this document so that the organisations can remediate the security risks.

Although these phases can be included in penetration testing, they are not present in all methodologies. Effective penetration testing must be structured and standardised, following a well-defined framework or methodology.

Frameworks in the IT industry provide a structured approach and a set of principles, concepts, guidelines and best practices. Ralph E. Johnson [34] described them as "the

skeleton of an application" and "a reusable design of all or part of a system that is represented by a set of abstract classes and the way their instances interact". Avison and Fitzgerald [35] defined a methodology as "a collection of procedures, techniques, tools and documentation aids". Wilhelm mentioned in his book [36] that frameworks focus primarily on processes, activities and tasks in a penetration testing context, whereas a methodology encapsulates them. Both concepts have similarities, but methodologies offer a stricter approach. They are usually more detailed and include step-by-step procedures. They also specify tools and strategies at every stage of the process. In practice, they can both be used indistinctly depending on the use case of the penetration testing process and the system subject to the analysis. Some research texts even use both words to call the same concept within the same text as if they were synonyms, so for this thesis, the choice of their naming is based on the sources used.

Since penetration testing is a procedure that is widely used in the industry, several studies have applied it to a wide range of systems. The primary methodologies and frameworks that can be used for the purpose of penetration testing according to different studies [37], [38] are:

- Open Source Security Testing Methodology Manual (OSSTMM): A peer-reviewed manual providing a scientific methodology for characterising operational security [39]. The guidelines offer verified information so the testers don't need to rely on best practices, but at the same time, that implies an increased complexity and demand for resources [39].
- Payment Card Industry Data Security Standards (PCI-DSS): It is a standard ensuring the security of all operations related to credit card information and payment [40]. This standard is helpful in the context of penetration testing for analysing the security risks of a system that accepts payment operations.
- Open Web Application Security Project (OWASP): This open-source project was designed to combat insecure software. It was initially intended for web security, its main purpose. For this reason, it can be used to test web pages successfully using the Web Security Testing Guide (WSTG) [41]. However, the same project also offers other standards that can be used for application penetration testing, such as The OWASP Mobile Application Security Testing Guide [42].

- NIST SP 800-115: Created by the National Institute of Standards and Technology, this technical guide is intended to assist organisations in performing security tests. Practical recommendations for the design, implementation and maintenance of security tests [43] serve to find vulnerabilities in a system or network.
- Penetration Testing Execution Standard (PTES): This standard sets seven stages to cover the penetration testing process, as explained above by the guide published by Georgia Weidman [31]. This set of guidelines and best practices aims to standardise the penetration testing approach, ensuring consistency throughout the process.

In addition to those alternatives, a new vulnerability research methodology, PatrIoT, specifically crafted for IoT devices emerged in November 2022 [44]. It was released by the Division of Network and Systems Engineering of the KTH Royal Institute of Technology. According to its documentation [44], it provides a structured methodology covering the whole penetration testing process, supplementing it with valuable information like the Compilation of Top 100 Weaknesses, a list of the most common weaknesses in IoT devices. It also includes a custom simplified risk scoring system and different guidelines. This methodology is beneficial for testing IoT products, which often have unique security challenges compared to traditional IT systems. This methodology has been previously used in several studies testing the security of different IoT devices, such as a Smart Garage Door [45], a drone [46] or a vacuum robot [47]. Even though IoT penetration testing has been achieved using general-purpose methodologies in previous literature [48], this research could benefit from an IoT-specific approach.

Apart from those penetration testing methodologies, other threat modelling frameworks can be useful in penetration testing. A threat modelling framework is aimed to identify all possible threats to the system. Mitigation techniques can also be included in the threat models. The threat models can be combined with the methodologies and frameworks described above to represent the security risks. Commonly used methods [49], [50] are:

- STRIDE is the most widely used threat modelling methodology [51]. It covers six different threat categories: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Elevation of Privilege.
- LINDDUN: It is a privacy threat modelling framework providing a systematic approach to privacy threat assessment [52]. The threats affecting the system are assigned into the types that give this framework its name: Linking, Identifying, Non-repudiation, Detecting, Data Disclosure, Unawareness and Non-compliance [53]. Similar to STRIDE modelling, this methodology is also used to identify threats to a system, but it focuses on those affecting privacy.
- PASTA: Also called Process for Attack Simulation and Threat Analysis, it is a risk-centric methodology. It includes seven stages to align business objectives and technical requirements: Defining Objectives, Defining Technical Scope, Decomposing the Application, Analysis Threats, Vulnerability Analysis, Attack Analysis and Risk and Impact Analysis. This approach focuses on business impacts and protecting the most valuable assets [54].
- CORAS: This approach provides a customised threat and risk modelling language. It facilitates the graphical representation of a system but also provides guidelines on how to use the method to carry out practical risk analyses. [55].
- TRIKE: An asset-centric threat modelling approach that accomplishes this by generating threat models reliably and repeatedly. For this approach to be successful, it is key to understand the system completely. Communication with stakeholders is also a big part of this approach [56]. Since Black Box penetration testing is used, this approach is not suitable for this research work.
- Attack Trees: This method provides a methodical way of describing the security of systems based on different types of attacks. The attacks on a system are represented in a tree structure where the goal is the root node, and the ways of achieving the goal are represented as leaf nodes [51]. Attack Trees are sometimes combined with other methodologies, such as STRIDE.

Since telepresence robots have audio and video recording capabilities, they handle sensitive information, so privacy is a big issue to assess. While LINDDUN can assess privacy more effectively, other widely used approaches like STRIDE can also cover non-privacy-related threats, such as Denial of service.

Finally, to give more background to the academic research, it is necessary to establish the type of penetration testing carried out in this work. There are three main types of penetration testing depending on the knowledge of the system to be attacked [57]. In the Black Box Testing type, limited information about the organisation is available to the pen-tester. They use methods such as fingerprinting or footprinting to gather information about the system subject to the attack. This type is more challenging as it requires the tester to perform additional steps to get knowledge that, in the best case, is equivalent to White Box Testing, the second type. In this type, details of the system to be attacked are provided to the pen-tester, including network-topology documents, assets or any other valuable information. This way, testers have full knowledge of the system's internal network. Finally, the Grey Box Testing type is only recognised by some researchers and ignored by other studies [58]. This scenario is in between White and Black Box Testing. Therefore, testers have partial knowledge about the systems to be attacked. Gathering additional information might be required to perform accurate threat modelling. The type used in this research is Black Box Testing. Due to the code not being provided, a combination of LINDDUN and STRIDE for threat modelling was attempted at an early stage of this thesis, but it was unsuccessful. Even though LINDDUN seemed appropriate for this purpose, its application revealed that it requires a deep understanding of the software implementation. There were several sections where it was required to fully understand the handling of every data element and the applicability of law and regulations, which was unfeasible given the lack of access to the code running on the telepresence robot and its cloud provider.

There exists a lack of studies analysing the cyber security of telepresence robots. From all sources consulted it was only possible to find a limited number of references analysing those devices. In 2017, Rapid7 identified multiple vulnerabilities in the Double Robotics Telepresence Robot [59]. The first vulnerability, R7-2017-01.1, allowed an unauthenticated user to access sensitive device information, including device serial numbers, current and historical driver and robot session information, device installation keys, and GPS coordinates. This could be achieved by manipulating the URL parameters in the API calls. The second vulnerability, R7-2017-01.2, was related to static user session management. The access token, created during account assignment to a robot, was never changed or expired. If this token was compromised, it could be used to take control of a robot without a user account or password. The third

vulnerability, R7-2017-01.3, was associated with weak Bluetooth pairing. The pairing process between the mobile application (iPad) and robot drive unit did not require the user to know the challenge PIN. Once paired with the robot drive unit, a malicious actor could download the Double Robot mobile application from the Internet and use it to take control of the drive unit.

In 2018, Zingbox published multiple vulnerabilities found in the robot Celia manufactured by Vecna [60]. The first vulnerability, CVE-2018-8860, allowed the firmware to be intercepted when the robot downloaded updates as they were transferred using HTTP. This firmware contained hardcoded credentials registered under CVE-2018-8858. The third vulnerability, CVE-2018-8866, allowed remote code execution on most GET parameters due to the lack of proper input validation. The third vulnerability, CVE-2018-17931, also allowed code execution by inserting a USB stick with the code contained in a particular file name executed on boot. The vulnerability CVE-2018-8858 combines different sensitive information disclosed by the robot, including credentials, chat conversations and pictures. Lastly, the vulnerability CVE-2018-17933 allowed the execution of sensitive XMPP commands, resulting in the disclosure of the camera feed.

In 2020, McAfee released the results of a security analysis of the Temi telepresence robot [61]. The first vulnerability, CVE-2020-16170, was due to hardcoded credentials. CVE-2020-16168 revealed an origin validation error, CVE-2020-16167 showed a lack of authentication on a critical function and on CVE-2020-16169 there was an authentication bypass. These three related works on cyber security on telepresence robots show not only potential vulnerabilities that could be found on the device analysed in this thesis but also shows the methodologies used, in all cases requiring first to extract the software and then perform an analysis of the code.

2.2 PatrIoT

PatrIoT is a vulnerability research methodology that evaluates the cybersecurity aspects of different IoT devices. Traditional penetration testing methodologies often prove inadequate in addressing the unique vulnerabilities of IoT ecosystems. Consequently, specialised approaches like the PatrIoT methodology have emerged to provide a practical and agile framework for IoT security assessments. It was revealed in November 2022 by KTH Royal Institute of Technology, and therefore, it’s still not as

widely used as other methodologies because it is still relatively unknown. However, one of its advantages is that it is explicitly designed for IoT devices, making it ideal for this thesis.

PatrIoT offers a four-stage methodology built upon four key elements to provide support during those stages: Logical Attack Surface Decomposition, Compilation of Top Weaknesses, Lightweight Risk Scoring and Step-by-Step Penetration Testing Guidelines [44]. The element Logical Attack Surface Decomposition is used in all phases to provide segmentation of the attack surfaces in an attempt to achieve completeness. The attack surface is logically decomposed based on the technology used, which is divided into seven surfaces: Hardware, Firmware, Radio, Network, Web, Cloud, and Mobile. The Hardware surface includes the system's architecture, as any other computing device, and the device's specific features, such as data sensors, motors or other additional attachments. Firmware combines the operating systems with the utilities installed and the configuration applied to the device. Radio covers the local connectivity, usually wireless connections between the devices or controllers. Network targets the network service and connectivity, usually Wi-Fi, making the device remotely available. The Web covers the web application managing the device. The Cloud surface involves the communication API connecting the device and the server. Finally, Mobile covers the controlling application, usually installed on phones, to allow access and control of the IoT device. With the surface decomposed, it is easier to target each of them individually to assess their specific threats independently.

The first stage is Planning, where it is necessary to define the scope of the tests, gather information about the device to be tested and discover information about it. The scanning can be done actively and passively. Active scanning directly interacts with systems and devices, sending probes and packets to elicit responses, revealing details like open ports, services, and potential vulnerabilities. This provides in-depth information but generates noticeable network traffic. Passive scanning analyses existing network traffic, gleaning information without direct interaction and combining it with information publicly available. For this stage, the first key element is used to ensure the completeness of the planning stage: Logical Attack Surface Decomposition.

In the second stage of the methodology, Threat Modelling is performed. This involves eliciting the threats affecting the system, analysing the potential vulnerabilities that

could lead to the threats and ranking that information using risk scoring to prioritise its analysis. The second key element, Compilation of Top Weaknesses, is used to guide the threat modelling process. This compilation includes a list of 100 security weaknesses that could affect an IoT device. This list is derived from the OWASP IoT Top 10 project [62], which contains the top ten most critical security risks in the IoT space. In addition, this list is enriched with the CWE Mitre's project [63], a list of common weaknesses affecting software and hardware. For firmware vulnerabilities, the OWASP Firmware Security Testing Methodology (FSTM) [64] has also been used as a source for the Threat Traceability Matrix. Finally, for threats affecting the control websites and applications, the OWASP Web Security Testing Guide [65] and OWASP Mobile Security Testing Guide [66], respectively. This list is enriched with clear descriptions, risk impacts, and severity values to assess risk scoring, which is the next element. Once the threats are modelled, the Lightweight Risk Scoring is used to prioritise them. It provides a more straightforward approach to other scoring systems, like the Common Vulnerability Scoring System (CVSS) with eight parameters [67] or OWASP's risk rating approach with 16 parameters [68]. This custom-proposed approach derives from the well-known DREAD method but only considers three parameters on a scale of one to three to output a score for each risk. Those parameters are Impact, Coverage and Simplicity. Default values are provided for Impact, with value 3 for remote code execution, 2 for authentication bypass, weak authentication or authorisation, tampering, and privilege escalation, and 1 for information disclosure and denial of service. Coverage is determined based on the affected users, with a value of three if all users are affected by the threat, two if it only affects a group of users or one otherwise. For the Simplicity parameter, if exploit code is found to be executed without modification, it receives the value three; if the code needs to be modified, it scores two, and if it needs to be developed from scratch, the parameter is set to value one. Once the parameters are set, a formula is used to compute the final risk score:

$$Risk = \frac{impact + coverage + simplicity * 3}{5}$$

The third phase, Exploitation, runs exploits for the vulnerabilities found in the previous stage. Those exploits can be gathered from common vulnerabilities or specially crafted for the device. If the system is successfully hacked, elevation of privileges is attempted to gain full control of the device. The fourth key element, Step-by-Step Penetration

Testing Guidelines, is used to support this phase. During penetration testing, it is usual to find online guidelines for penetration testing of the surfaces Network, Web, Cloud, and Mobile since they are common to most IT scenarios. However, step-by-step guidelines are provided for the surfaces that are IoT-specific, Hardware, Firmware and Radio. This ensures a simplified exploitation stage and consistency across the assessment of different devices, reducing the risk of overlooking crucial steps.

Finally, the last phase is Reporting, which documents the findings of the penetration testing process. It includes details about discovered vulnerabilities, exploited weaknesses, successful privilege escalation attempts and, optionally, recommendations for remediation and mitigation.

PatrIoT methodology covers the complete penetration testing process, guiding and providing helpful information along the different phases. Additionally, it accomplishes its purpose in a flexible way, not limiting the specific methodologies to be used in each phase but only suggesting them. Any language and method can be used for threat modelling, and the same extends to other stages like risk scoring or reporting. In addition to being explicitly designed to address the cybersecurity issues of IoT products, all those features made it the right choice for this research.

2.3 STRIDE

STRIDE is a threat modelling framework developed by Microsoft to help identify and address security vulnerabilities in software products [69]. It stands for:

- Spoofing: Impersonating a legitimate user, system or entity to gain unauthorised access. This could include impersonating a legitimate user or masquerading as a trusted system component.
- Tampering: Modifying data without authorisation. This includes altering information in transit, modifying stored data, or tampering with the integrity of system software or hardware.
- Repudiation: Denying an action performed, making it difficult to trace the perpetrator. This could include users disowning their activities or systems failing to log crucial events.

- Information Disclosure: Exposing confidential information to unauthorised individuals. This includes the unintended release of confidential data such as credentials.
- Denial of Service: Making a system unavailable or unusable. The aim is to disrupt or impair the normal functioning of a system, rendering it temporarily or permanently unavailable to users.
- Elevation of privilege: Gaining unauthorised access to higher levels of permissions within a system. Attackers aim to escalate their permissions to manipulate or control the system beyond their legitimate scope, such as gaining root access.

By systematically analysing these six threat types, the STRIDE model enables us to proactively identify and address potential security risks and discover their associated vulnerabilities. This security threat model was designed to be used during the design phase [69], but in previous literature, it has been used to model threats of final consumer products [70], [71]. STRIDE-per-element analyses threats against individual elements within a system, such as components, modules, and functionalities, while STRIDE-per-interaction analyses threats across interactions between elements, focusing on data flows and communication channels. Previous literature shows that STRIDE-per-interaction is more efficient, and "its protection strategies are normally enough to protect system (as cyber attacks normally involve malicious interactions between system components)" [72]. However, a study comparing both methods revealed a higher level of completeness in the research results using STRIDE-per-element [73], which was the reason for the choice of this method for this thesis.

A data flow diagram representing the whole system is required to perform threat modelling using STRIDE, breaking it down into its logical and structural components. Four data-flow diagram elements are used to represent the system. External entities represent actors interacting with the system but not part of it. Data flows show the paths of data transmitted between the elements. Data stores represent places where the data is held. Processes represent the actions transforming the data. STRIDE threats don't apply equally to all elements; only some are susceptible to each threat, as shown in Table 1.

Table 1. Susceptibility of DFD elements to STRIDE threats [74].

DFD element	S	T	R	I	D	E
Entity	✓		✓			
Data Flow		✓		✓	✓	
Data Store		✓	✓	✓	✓	
Process	✓	✓	✓	✓	✓	✓

STRIDE has been chosen for this research because of its wide usage and acceptance. It is a recognised framework with a large amount of literature available on its successful use in the IT and, more specifically, penetration testing industry, with proven results of its suitability for this type of research. In addition, it follows a structured approach while being easy to implement, so it perfectly fits this research.

Since the previous literature review has shown that there is a research gap in analysing the security of telepresence robots, the related work showed different techniques to accomplish the same goal on general IT and IoT devices. The related work researched for this thesis, which is explained in this section, has provided guidance on the different approaches for threat modelling and penetration testing. This knowledge enabled us to adapt the existing methodologies to fit the specific conditions of a telepresence robot, as shown in the next section of this document.

3 Methodology

This section describes the structured methodology used to conduct comprehensive penetration testing on the telepresence robot subject to study. This research uses a qualitative research method in the form of a case study. The methodology combines STRIDE and PatrIoT to provide an understanding of the telepresence robot security landscape. As shown in Figure 2, the proposed methodology follows the PatrIoT structure, with the same phases: Reconnaissance, Threat Modelling, Exploitation and Reporting. The methodology stages and their key elements are summarised in Figure 2, showing the phases of the proposed method and the tools and sources of information used in each of them.

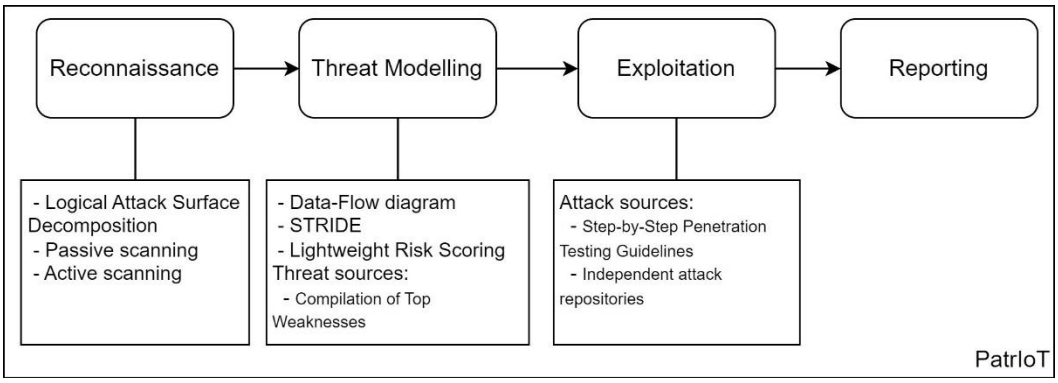

Figure 2. Methodology used, based on PatrIoT.

Elements of PatrIoT are used along the whole process, like Logical Attack Surface Decomposition, for a better understanding of the system and the threats that could affect it. This element is heavily used in the first phase, Reconnaissance, as it is critical to ensure a complete understanding of the telepresence robot and its environment. This phase aims to gather as much valuable information about the system as possible. Once a clear understanding is achieved, more information is gathered using passive and active scanning.

The next phase is Threat Modelling, where, using the information gathered from the previous section, a data-flow diagram is created showing a detailed view of the robot's system. This diagram shows all components that are part of the system and the data

exchange among them. With this representation, threat modelling using STRIDE is performed, using the Compilation of Top Weaknesses as a source of possible threats to the system. Finally, those threats discovered are ranked according to the risk they pose to the system, based on the Lightweight Risk Scoring.

The next phase, Exploitation, shows the planning of the tests that attempt to exploit the system. The Step-by-Step Penetration Guidelines, in combination with other public sources of attacks, are used to craft those tests. This phase shows both successful and failed attempts, as both help give an overview of the robot's security.

Finally, the Reporting phase documents the results of the tests so they can be shared with the parties involved. This report also includes suggested improvements that, once applied, could enrich the system's cyber security.

3.1 Tools

This section presents the list of hardware and software used in this thesis. The devices used to achieve the telepresence robot penetration testing are an Android phone, a Windows laptop to simulate interactions with the robot, and a computer capable of booting Windows and Kali Linux to execute the attacks.

The Reconnaissance phase involved the use of active network scanning software. Angry IP Scanner [75] provided the robot's IP address, and Nmap [76] was used to discover open ports and services on the robot. The tools Fing for Android [77] and MACVendors [78] provided detailed information about network devices based on their MAC addresses.

The Exploitation phase included several utilities with different purposes. HTTP Toolkit [79] and PuTTYgen [80] were used to generate keys and certificates. Android SDK Platform Tools (ADB) [81] was used to establish shell access to the robot and execute various commands. Firmwalker [82] was used to automatically analyse the robot's file system, and Apktool [83] was used to decompile and analyse the robot's control application. Android Studio [84] was used to create a virtual device for the dynamic analysis of an Android application. Metasploit [85] was used to execute attacks on the robot, along with MsfVenom [86] to create a malware application. TeamViewer [87] was employed to remotely control devices. Several network utilities were used, with

Linux Wifi Hotspot [88] used to create a compromised access point, along with Wireshark [89] to analyse its traffic. Bettercap [90] was used to perform ARP spoofing and deny service to the robot, and Aircrack-ng [91] for other denial-of-service attacks. Network traffic modification involved the use of Ettercap [92], Burp Suite [93], Burp-Non-HTTP-Extension [94], Python [95] and its utilities Scapy [96] and NetfilterQueue [97].

4 Reconnaissance

For a successful attack on a system, it is critical to understand what the system is and what specific characteristics apply to it. This section collects information about the hardware components of the system, the operating system and software running on it, and network and communication protocols. Information gathering has been accomplished using passive and active methods. Passive methods used in this thesis include gathering information from the device's user, developer manuals and physical evaluation. Whereas passive methods provide detailed information about the model's characteristics, active methods have provided real-world updated data fitting the model's real configuration. In this section, the type of information gathered is presented, but the content is omitted to preserve the privacy of the brand and model studied. Only basic information about the system is shown, so it can't be mapped to a single device model.

4.1 Passive scanning

The manufacturer offers user and developer manuals of the device. Those manuals include very detailed information about the hardware configuration, the connection among those components, the operating system, additional services running on it, and the filesystem. In addition, they provide helpful information about how to access the shell, install third-party custom programs, and interact with its API. This information, combined with the physical inspection of the robot, provides valuable insights about the device.

The complete robot's hardware architecture was discovered. The device is running on an Intel 64-bit processor. For connectivity, the processing board is connected to the screen via HDMI. USB ports are offered but only internally in an expansion hub that requires disassembly to be accessed. By default, the USB expansion hub has a Wi-Fi dongle featuring the standard 802.11ac.

The robot's Physical evaluation revealed that to access the motherboard and the expansion ports, it is necessary to unscrew the lid from the main body to reveal access to the internals. In the entire system, no connectivity port is easily accessible to the users; they are all hidden under covers, which require tools to be opened. There is one power button accessible to the user. The last noticeable element is a speaker providing buttons for audio-related activities such as controlling the volume, microphone and call initialisation and termination.

From a software side, the documentation reveals that the operating system is a custom hybrid between Android and Ubuntu. This provides services from both ecosystems, such as remote connectivity using Android Debug Bridge (ADB) or SSH on an OpenSSH server. In addition, the manufacturer offers an SDK, allowing integrations to control the robot's different features. It can be used to take full control of the screen and audio system or physical interfaces, including lights or motors. Apart from the SDK, which is publicly available, the software and services installed in the robot aren't accessible, and therefore, they can't be statically analysed.

In order to connect to the robot, the user needs to access the connection portal's website with a web browser. For this, a user account with access to the robot is required, or a temporary link needs to be created granting access to an unregistered user for a specific duration. Based on the robot's network conditions, there are two modes of operation for video calls. Once the user requests a connection to start a call, the request is processed, and the server and robot evaluate if the robot is locally reachable by the client's computer. This would be the case if the client is connected to the same Wi-Fi as the robot. This is preferred as it provides the most stable signal and minimum latency. The connection will be established through the server if the robot is unreachable locally. In this mode, the cloud acts as a relay, directing the information from one endpoint to another. The details of the behaviour of the Cloud API are unknown, and it hasn't been discovered using active scanning.

Wi-Fi. Most IoT devices include additional features such as Bluetooth or ZigBee, which are not included in this model. However, Wi-Fi connectivity is analysed from two perspectives: the Network attack surface, including its services, and the Radio attack surface regarding wireless signals. The Mobile attack surface does not apply to this

device. A mobile application does not exist to connect to the robot as in other IoT devices, so the static analysis of that feature is not included.

Finally, the documentation also has a separate section for security and privacy describing how data is protected. Even though the protocols are not described in detail in this thesis to avoid mapping the robot to a specific brand and model, it is stated that call-related data is end-to-end encrypted. The data is not even available to the data provider's server as it is only decrypted in the robot and client's browser. The rest of the data (status and commands) is transmitted to the cloud using HTTPS.

4.2 Active scanning

For active scanning of the robot, the network was analysed to find out the robot's IP address. With the configuration applied to the robot, it was connected to an open Wi-Fi (with no password required) available in the building where the robot is used. Once connected to the Wi-Fi hotspot, the tool Angry IP scanner was used to provide a list of hosts available in the network along with their hostname. On an Android device, the application Fing was used for the same purpose but included detailed information based on the MAC address. This additional information filtered out devices that were not the telepresence robot based on the device's manufacturer. Then, the Nmap tool was used to discover more details about the system, in this case, the services that are active and exposed. The results showed that the ports with open and running services were 22, used for SSH, and 5555, used for ADB. In addition, it was discovered that when a user is connected to a call, an additional port is open to enable connectivity between the user and the device. All those ports use the TCP protocol. Nmap was unable to recognise the operating system and suggested several candidates, including different versions of Linux and Android devices. This was the expected outcome due to the hybrid OS. In another scenario, it would be helpful to scan and analyse the website controlling the functionalities of the telepresence robot. However, since it is out of the scope of this research, it was omitted.

A summary of the data collected is displayed in Table 2 for a better understanding of the data collected and the methods used for it.

Table 2. Summary of information gathered during Reconnaissance phase.

Method	Category	Details
Passive	Architecture	Hardware (processor, memory, disks).
Passive	Connectivity	Physical ports, screen, buttons, speaker and microphones.
Passive	Software	Operating System and services enabled (SSH and ADB).
Passive	SDK	Usage of the SDK for its control.
Passive	Network topology	Connection modes (Local and Relay), Connection methods (via Cloud) and Data encryption (HTTPS and TLS)
Active	Network details	IP and MAC addresses and open ports (ADB, SSH and call service).

5 Threat Modelling

This chapter will show the comprehensive threat modelling approach followed in this research using STRIDE as a threat modelling methodology, while PatrIoT guides the threats specific to IoT devices. Through in-depth data flow analysis, attack vectors, and potential impact, this combined approach aims to identify security risks across all facets of the control system, safeguarding both the robot and its users. Both models require a data-flow diagram, represented in Figure 4.

5.1 Data-Flow Diagram

A data-flow diagram is used to better understand how a telepresence robot works and which components are involved in its typical use. This shows a high-level overview of the system, including the processes it contains, the external entities it interacts with, the data stores and the flow of information among all those elements [98]. This enhances the device's knowledge and helps understand potential vulnerabilities and security considerations associated with transmitting and processing information.

Based on the two modes of communication between the client and the robot while a call is established, there are two options for the data-flow diagram, as seen in Figure 3. While in local mode, the video feed is transmitted peer-to-peer; it is routed through the cloud in relay mode.

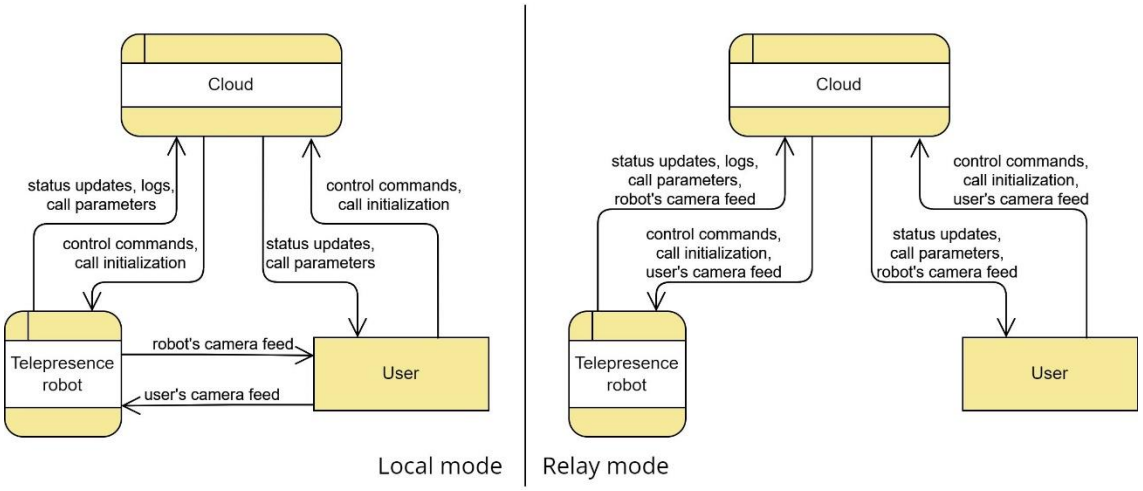

Figure 3. Modes of operation as a data-flow diagram.

The final diagram can then be composed using both modes, even though they will not run simultaneously. The final data-flow diagram is displayed below in Figure 4.

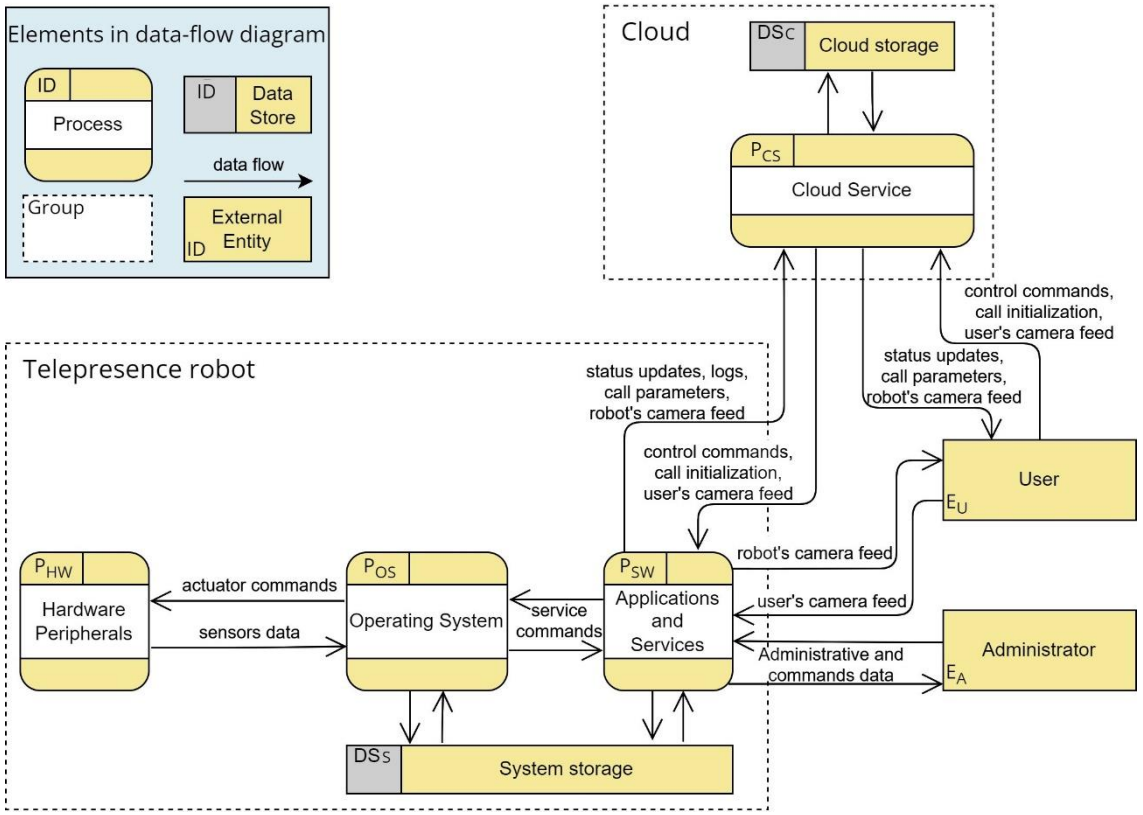

Figure 4. Telepresence robot as a data-flow diagram.

For the sake of simplicity of the document, the different entities and flows of the diagram have been assigned identifiers, as shown in Table 3. In that table, the identifiers have been generated to include the element type, with External Entities being Ei, Pi for the Processes, DSi for the Data Stores, and DF_i for the data flows, where '*i*' stands for

the element number. These same IDs will be used later in the thesis to reference the elements in the data-flow diagram.

Table 3. Data-flow diagram identifiers.

External Entities

ID	Name
E_U	User
E_A	Administrator

Processes

ID	Name
P_{HW}	Hardware Peripherals
P_{OS}	Operating System
P_{SW}	Applications and Services
P_{CS}	Cloud Service

Data Stores

ID	Name
DS_S	System Storage
DS_C	Cloud Storage

Data Flows

ID	Name
DF_1	Actuator commands
DF_2	Sensors' data
DF_3	Service commands
DF_4	Status updates
DF_5	Logs
DF_6	Call parameters
DF_7	Robot's camera feed
DF_8	Control commands
DF_9	Call initialization
DF_{10}	User's camera feed
DF_{11}	Administrative and Commands data
DF_{12}	Database accesses

Three processes and a data storage represent the telepresence robot. The hardware is the first of them (P_{HW}), containing all the peripherals and electronics specific to telepresence robots. These peripherals sense and actuate on the environment, capturing data about the robot's surroundings and giving feedback to it. For this, it includes sensors like cameras, actuators such as wheels, and electronics combining features like the touchscreen or USB and other serial ports. The robot has a bidirectional flow of information (DF_1, DF_2) between the hardware peripherals processes (P_{HW}) and the Operating System process (P_{OS}).

To perform communication and control, the Application and Services process (P_{SW}) is in charge of connecting all the information and commands of the robot with the exterior, communicating with the Operating System via bidirectional service commands (DF_3). Even if a call is inactive, it communicates with the cloud provider, sending the data collected by the sensors (DF_4) and system logs (DF_5). This process also receives control commands on operating the actuators and the information to be displayed on the screen (DF_8). As an intermediary, the cloud process (P_{CS}) communicates with the client (E_U), sending status updates about the robot (DF_4) and receiving control commands (DF_8).

When a call is to be established, it is necessary to determine the connection mode. The clients can send call initialisation commands (DF_9) to the cloud, and the cloud then communicates to the robot to request the call (DF_9). Then, based on the network conditions, the connection mode is determined, and the connection is established accordingly. If it is local, it is necessary to create a direct connection between the robot and the user requesting it. The robot returns the call parameters (DF_6) to the cloud, which are then forwarded to the client. Once this process is done, it is then possible for the client to connect directly to the telepresence robot, sending the webcam feed to it (DF_{10}) and receiving the data collected from the robot's cameras (DF_7). On the other hand, if the connection is of type relay, both robot and client send their video feeds to the cloud and receive the stream of the other endpoint, downloading it from the same web service.

Finally, there is one external entity interacting with the robot. In the diagram, it is named Administrator (E_A), but it can be any person connecting to the robot locally, usually for administrative or developing purposes. The robot can open local connections like the SSH or ADB shells for local configuration, establishing a bidirectional data flow (DF_{11}). Finally, two data stores are used, one managed by the robot (DS_S) and the other by the cloud provider (DS_C). Both entities have bidirectional data flow to their corresponding databases (DF_{12}).

5.2 STRIDE

To ensure a systematic STRIDE threat modelling, each threat type was analysed for every entity in the diagram presented in Figure 4. Threats were analysed for processes, external entities, data stores and data flows. For the data flows, threats focused on every combination of source, data flow and destination in the diagram, where sources and destinations are processes, external entities and databases. This ensures that every interaction and entity in the diagram is analysed for potential threats. With the Compilation of the top weaknesses matrix provided by PatrIoT, it is possible to list the threats that can potentially affect the system and link them to the STRIDE model. In addition, other well-known vulnerabilities and weaknesses affecting IT systems have been added to the threats. All the threats present in this section are hypothetical, it is a list of threats that could potentially affect the system but need later analysis and

exploitation to confirm or deny their exitance. The threats found are listed in each STRIDE threat type section.

5.2.1 Spoofing

For the Spoofing threat type, several threats could affect the system. The most common spoofing threats to the system are man-in-the-middle related. In this use case, it would be possible for the attacker to spoof the identity of another entity. This could be used to assume the identity of a legitimate user and connect to the robot unauthorisedly. Taking the identity of the cloud is also hypothetically possible by using fake route advertising by DNS or ARP. Since the video is end-to-end encrypted, it would only be possible to intercept status updates, logs or call initialization parameters. The robot's identity could also be spoofed, and the video feed or other information about the user legitimately connecting to the robot could be obtained.

Finally, an attacker could also impersonate the administrator using weak credentials to connect to the robot locally using ADB or SSH connectivity. This means that potential spoofing threats could be used to spoof the external entities User (E_U) and Administrator (E_A), as well as the services Cloud Service (P_{CS}) and Applications and Services (P_{SW}). If this were the case, it would affect the data flows: Status updates (DF4), Logs (DF5), Call parameters (DF6), Control commands (DF8), Call initialization (DF9) and Administrative and Commands data (DF11).

The weaknesses appearing in the IoT Threat Traceability Matrix that could result in a successful spoofing are Authentication - Weak credentials, Authentication - Weak password recovery, Vendor APIs - Inherent trust of cloud or mobile application, Authentication bypass - Device to cloud or Insecure SSL/TLS issues. Table 4 shows the Spoofing threats along with the potential vulnerabilities or weaknesses that could cause them, grouped in the “Potential vulnerabilities” field.

Table 4. Spoofing threats.

ID	TS_1	**Threat type**	Spoofing
Elements affected		User (E_U)	
Description		An attacker spoofs the identity of a legitimate user to connect to the robot in an unauthorised way.	
Potential vulnerabilities		Weak credentials on the legitimate user's account. Weak password recovery on the legitimate user's account. Authentication bypass guessing the one-time connection link.	
Impact		Authentication bypass.	

ID	TS_2	**Threat type**	Spoofing
Elements affected		Cloud (P_{CS})	
Description		An attacker spoofs the identity of the cloud to act as a man-in-the-middle between the robot and a user.	
Potential vulnerabilities		Insecure SSL/TLS issues. Insecure connections to compromised Wi-Fi networks. Lack of domain spoofing measures. Implicitly trusted cloud.	
Impact		Sensitive data disclosure.	

ID	TS_3	**Threat type**	Spoofing
Elements affected		Administrator (E_A)	
Description		An attacker spoofs the administrator's identity to connect to the robot locally through SSH or ADB.	
Potential vulnerabilities		Weak credentials on the connection accounts. Authentication bypass.	
Impact		Authentication bypass, Code execution.	

ID	TS_4	**Threat type**	Spoofing
Elements affected		Applications and Services (P_{SW})	
Description		An attacker spoofs the identity of the robot.	
Potential vulnerabilities		Implicitly trusted device.	
Impact		Authentication bypass.	

5.2.2 Tampering

Table 1 shows tampering threats that can affect data flows, data stores and processes. Starting by processes, the four of them could theoretically be subject to tampering. However, some of the threats are unfeasible since all hardware is enclosed in the robot's shell, and this research is limited by not disassembling the device. This is the case for the Hardware Peripherals process (P_{HW}), which could be accomplished by unplugging devices like the camera and replacing it by a modified one in a way a potential attacker

could modify the images captured. The same applies to plugging in USB devices, where a plugged-in “Rubber Ducky” device could emulate a legitimate keyboard and access the terminal of the robot [99].

Since all available ports are physically protected and unavailable on the outside, tampering threats don’t affect most of the hardware connectivity. The only hardware accessible to the user is the screen of the robot. This screen lacks any protection mechanism and can be used to modify the device's configuration. For example, some allowed actions are altering Wi-Fi connectivity, which could be used to connect the device to a compromised hotspot, or enabling the ADB service for remote connections.

For the Operating System process (P_{OS}), a method to tamper with the operating system's data wasn’t found. Some other IoT devices have a physical interface to install firmware updates, and a modified version can be installed on them. This doesn’t apply to this specific robot.

The Cloud Service (P_{CS}) also resides out of the scope of this thesis. Since the manufacturer has not granted permission to attack that surface of the system, all cloud elements except for the legitimate flows interacting with it must remain untouched. This also extends to the Cloud Storage (DS_C). However, the Applications and Services process (P_{SW}) is subject to tampering attacks. The system provides a management tool to install Android applications on the device or via ADB commands, which means that tampered versions of applications are subject to be installed.

The System Storage data store (DS_S) is also subject to tampering attacks. Files stored in the system could be modified, although the information gathered doesn’t reveal which data is locally stored in the device, so the impact of a successful data store tampering is unclear. It is assumed that the device has credentials and configuration files stored on it for different purposes as any other Android or Ubuntu device, but at the time of threat modelling, it isn’t confirmed.

For the tampering of data flows, every combination of source, flow and destination was analysed to determine if they are subject to tempering attacks. The actuator commands (DF_1) and sensors data flow (DF_2) can’t be tampered with as it would require disassembly to physically access the connection between the robot and the peripherals,

although it is technically possible. If accomplished, a device could be attached to the motor's wires to send signals taking control of the robot's movement.

Both data flows involving service commands (DF_3) occur locally and are not exposed on any service. They could only be modified if an attacker had connected to the operating system via a shell. The data flows connecting the processes Applications and Services to Cloud Service and Cloud Service to User are treated differently. Status updates (DF_4), logs (DF_5), call parameters (DF_6), control commands (DF_8) and call initialization (DF_9) follow HTTPS protocol in combination with TLS 1.3 standard. If tampered with, an attacker acting as a man-in-the-middle could modify the data and send incorrect status updates or call parameters.

For transmitting the robot's and user's camera feeds (DF_7, DF_{10}), the protocol used is different and encrypted using AES-256. Even though an attacker could technically modify the video frames, the new altered frames wouldn't be coherent and would result in the camera's denial of service instead of the display of a crafted video. Administrative and commands data (DF_{11}) are subject to being tampered with on their wireless connection between the Applications and Services process and Administrator. ADB connection, by default, doesn't provide any security against tampering, so intercepted messages can be modified in both directions of the data flow.

Finally, the connection happens locally for the flows connecting the robot's database to the Operating System and the Applications and Services processes, requiring an active shell to the operating system to modify the information. Table 5 summarises the Tampering threats, weaknesses, and vulnerabilities affecting them.

Table 5. Tampering threats.

ID	TT_1	**Threat type**	Tampering
Elements affected	Hardware Peripherals (P_{HW})		
Description	An attacker can interact with the robot's screen to alter the device's configuration.		
Potential vulnerabilities	Lack of authentication mechanisms.		
Impact	Unauthorized configuration changes, Unexpected system behaviour.		

ID	TT_2	**Threat type**	Tampering
Elements affected		Applications and Services (P_{SW})	
Description		An attacker installs a modified version of an application with malicious code, enabling it to gain control of the robot, manipulate its behaviour or steal sensitive data.	
Potential vulnerabilities		Insecure customisation of OS platforms. Lack of signature on update file. Backdoor firmware.	
Impact		Control Takeover, Code execution, Sensitive data disclosure.	

ID	TT_3	**Threat type**	Tampering
Elements affected		System Storage (DS_S)	
Description		An attacker tampers files installed in the system to alter configuration and credentials.	
Potential vulnerabilities		Hardcoded credentials. Insecure filesystem permissions.	
Impact		Authentication bypass, DoS.	

ID	TT_4	**Threat type**	Tampering
Elements affected		Status updates (DF_4), Logs (DF_5), Control commands (DF_8)	
Description		An attacker alters the status updates or logs sent to the cloud to modify the perceived state of the robot.	
Potential vulnerabilities		Insecure SSL/TLS issues.	
Impact		DoS.	

ID	TT_5	**Threat type**	Tampering
Elements affected		Call parameters (DF_6), Call initialization (DF_9)	
Description		An attacker modifies the call parameters or initialization commands, resulting in an incorrect call establishment.	
Potential vulnerabilities		Insecure SSL/TLS issues.	
Impact		DoS.	

ID	TT_6	**Threat type**	Tampering
Elements affected		Administrative and Commands data (DF_{11})	
Description		An attacker captures and modifies shell access to execute arbitrary commands on the robot.	
Potential vulnerabilities		Lack of transport encryption. Authentication bypass.	
Impact		Code execution.	

5.2.3 Repudiation

The conditions of this research make modelling repudiation threats a challenge. Since the manufacturer didn't agree to share the logs, and the type of penetration testing is black-box, it is impossible to determine what actions are being logged and, therefore, which are subject to repudiation. However, the threats described in this section are analysed from a theoretical perspective, giving an overview of the threats that could deny the involvement of an attacker or user in some action.

The only threat found is the denial of the robot's use. The cloud server logs connections established with the robot and displays them in the management console. However, the robot doesn't seem to send the status of local administrative connections to the cloud server. This means that connections established by ADB and SSH seem free of logging and, therefore, subject to repudiation. An illegitimate user could connect to the robot via ADB or SSH to avoid being detected by the cloud service and control the robot without establishing a call. This could be used, for example, to access the camera without being noticed and without the option to inspect which individual was connecting to it, as no user account is needed. Table 6 shows the only Repudiation threat found.

Table 6. Repudiation threats.

ID	TR_1	**Threat type**	Repudiation
Elements affected	User (E_U)		
Description	A user connects via ADB or SSH to take control of the robot without getting logged or traced.		
Potential vulnerabilities	Lack of complete logging. Authentication bypass.		
Impact	Untraceable unauthorized access.		

5.2.4 Information Disclosure

In order to model the threats of information disclosure type, it is not trivial to determine which information is confidential and which data doesn't have any risks involved when published. It is clear that all user-related information should be kept secret, but other information, such as the versions of software installed or network addresses, is arguable.

On the Hardware Peripherals process (P_{HW}), the robot's screen reveals valuable information about the system. All network addresses can be listed along with the saved Wi-Fi hotspots, logged-in accounts and installed applications. By default, the

Applications and Services process doesn't reveal any valuable information, but a TCP port is opened when a user is connected. This poses a security risk as scanning the open ports of the device makes it possible to determine if the robot is being used. Depending on the use of the robot, this information could be interesting for an attacker to know when to launch a Denial-of-Service attack to disrupt a call. Also, if the robot is being used for surveillance purposes, an attacker could realise a timeframe when it isn't recording video to physically perform a crime without being registered.

The Operating System process (P_{OS}) can disclose confidential information if an attacker finds a way into the system via a terminal connection or installed malware. The Cloud Service process (P_{CS}) could be subject to data leakage if not implemented correctly. Both data stores are not publicly accessible, but their connection is for local access. Since the cloud is out of the scope of this research, it will be listed as not having any risks associated with it. The System Storage (DS_S) can disclose confidential information also when access to the system is granted. All data flows with the cloud as source or destination are end-to-end encrypted, so their content doesn't reveal any data if intercepted. The actuator commands (DF1), sensors data (DF2) and service commands (DF3) data flows are not accessible by any user and therefore don't disclose any confidential information. Administrative and Commands data (DF_{11}) could have potential information disclosure threats if the packages involved in the shell interaction weren't encrypted. This would allow an attacker to capture the commands executed and see confidential information such as credentials or configuration details. Table 7 shows the Information Disclosure threats.

Table 7. Information Disclosure threats.

ID	TI_1	**Threat type**	Information Disclosure
Elements affected		Hardware Peripherals (P_{HW})	
Description		An attacker can interact with the screen to reveal sensitive information about the robot.	
Potential vulnerabilities		Lack of authentication mechanisms. Device information leakage.	
Impact		Sensitive data disclosure, Identification of vulnerable services, Facilitation of further attacks.	

ID	TI_2	**Threat type**	Information Disclosure
Elements affected		Applications and Services (P_{SW})	
Description		An attacker can scan the open ports of the robot to gather information about its usage.	
Potential vulnerabilities		Device information leakage.	
Impact		Sensitive data disclosure, Bypass of security measures.	

ID	TI_3	**Threat type**	Information Disclosure
Elements affected		Operating System (P_{OS})	
Description		An attacker connects to the operating system, having access to critical information.	
Potential vulnerabilities		Device information leakage. User data disclosure. Insecure filesystem permissions	
Impact		Sensitive data disclosure, Identification of vulnerable services, Facilitation of further attacks.	

ID	TI_4	**Threat type**	Information Disclosure
Elements affected		Cloud Service (P_{CS})	
Description		The cloud service discloses confidential information.	
Potential vulnerabilities		Username enumeration. User data disclosure. Device information leakage.	
Impact		Sensitive data disclosure, Facilitation of further attacks.	

ID	TI_5	**Threat type**	Information Disclosure
Elements affected		System Storage (DS_S)	
Description		An attacker gains access to the robot's filesystem, having access to files, including drivers and configuration files.	
Potential vulnerabilities		Sensitive data exposure - Hardcoded credentials. Sensitive data exposure - Encryption keys and algorithms. Configuration - Insecure filesystem permissions. Insecure data storage.	
Impact		Sensitive data disclosure, Facilitation of further attacks.	

ID	TI_6	**Threat type**	Information Disclosure
Elements affected		Administrative and Commands data (DF_{11})	
Description		An attacker intercepts shell commands disclosing confidential information such as credentials or configuration steps.	
Potential vulnerabilities		Lack of transport encryption. Authentication bypass. User data disclosure.	
Impact		Sensitive data disclosure, Exposure of configuration details, Facilitation of further attacks.	

5.2.5 Denial of Service

Denial of service threats can affect Processes, Data Stores and Data Flows, as shown in Table 1. The Hardware Peripherals (PHW) could be physically damaged to prevent them from functioning, but this research will be limited to an IT perspective, avoiding doing physical damage to any asset. Two methods can also deny the correct working of the Operating System (POS). By physically pressing the power button in the device, which is not a real threat, or by remotely rendering it unavailable. This second method to manually take it down would involve connecting to the shell and running a command to turn off the system. Also, a more advanced technique would be to install a script that executes every time the robot turns on and runs a specific command to turn it off.

The Applications and Services process (PSW) could also deny access to the robot. If an application is overloaded or maliciously forced to freeze, it would disrupt the correct functioning of the robot. To finish with the process threats, the Cloud Service (PCS) could be subject to Distributed Denial of Service (DDoS) attacks if not protected effectively. This threat lands outside this research's scope since the attacks on the cloud provider are not contemplated. The same applies to Cloud Storage (DSC) since the cloud provider should be attacked to perform a denial of service.

The System Storage (DS_S) data store is not subject to denial of service either because it is logically stored in the same storage unit where the operating system is running. So, making the data store unavailable would make the whole Operating System process (P_{OS}) inaccessible.

Regarding denial of service of data flows, the same characteristics apply similarly to the other threat types. The flows that remain inside the telepresence robot can't be denied, which is the case for Actuator commands (DF_1), Sensors' data (DF_2) and Service commands (DF_3). All other data flows are subject to denial-of-service attacks if they are not sufficiently protected against them, so they are subject to this type of threat. However, based on functionality, grouping those data-flow threats based on their paths makes more sense. For this, there will be three distinct groups of flows. The first one groups flows involved in connections established on local mode. This means its denial consists in taking down the direct connections between the User (E_U) and Applications and Services (P_{SW}). The second one tries to interrupt connections established on relay mode, those connecting the same entities but doing so through the Cloud Service

process (P_{CS}). The last group involves the Administrative and Commands data (DF_{11}), whose impact is disrupting local shell connections. Table 8 provides details about the discovered Denial of Service threats.

Table 8. Denial of Service threats.

ID	TD_1	**Threat type**	Denial of Service
Elements affected		Operating System (P_{OS})	
Description		An attacker runs scripts on the system, causing it to shut down or reboot.	
Potential vulnerabilities		Authentication bypass. Backdoor firmware.	
Impact		DoS.	

ID	TD_2	**Threat type**	Denial of Service
Elements affected		Applications and Services (P_{SW})	
Description		An attacker exploits a vulnerability, crashing a running service or application.	
Potential vulnerabilities		Business and logic flaws. Lack of Security Updates and Patches	
Impact		DoS.	

ID	TD_3	**Threat type**	Denial of Service
Elements affected		Status updates (DF_4), Logs (DF_5), Call parameters (DF_6), Robot's camera feed (DF_7), Control commands (DF_8), Call initialization (DF_9), User's camera feed (DF_{10})	
Description		An attacker disrupts the service of a relay call connection.	
Potential vulnerabilities		Malformed input. Connection endpoint disruption.	
Impact		DoS.	

ID	TD_4	**Threat type**	Denial of Service
Elements affected		Status updates (DF_4), Call parameters (DF_6), Robot's camera feed (DF_7), Control commands (DF_8), Call initialization (DF_9), User's camera feed (DF_{10})	
Description		An attacker disrupts the service of a local call connection.	
Potential vulnerabilities		Malformed input. Connection endpoint disruption.	
Impact		DoS.	

ID	TD_5	**Threat type**	Denial of Service
Elements affected		Administrative and Commands data (DF_{11})	
Description		An attacker disrupts the service of an administrative shell.	
Potential vulnerabilities		Malformed input. Connection endpoint disruption.	
Impact		DoS.	

5.2.6 Elevation of Privilege

The last STRIDE threat type, Elevation of Privilege, only affects the data-flow diagram's elements of the type Process. The installed Hardware Peripherals (P_{HW}) do not have a concept such as privileged access. The Operating System (P_{OS}) is subject to privilege escalation on both parts of the hybrid architecture, the Android and Ubuntu consoles. Applications and Services (P_{SW}) are also subject to privilege escalation by running any service or application as the root user to accomplish elevated rights and exploit other system parts. In the Cloud Service (P_{CS}), it is theoretically possible to escalate privileges from a standard user account to an administration account. However, to do so, it is necessary to either attack the cloud provider, staying out of the scope, or impersonate the account with privileged rights, being considered under spoofing threats. Elevation of Privilege threats are detailed in Table 9.

Table 9. Elevation of Privilege threats.

ID	TE_1	**Threat type**	Elevation of Privilege
Elements affected	Operating System (P_{OS})		
Description	An attacker escalates privileges to run a terminal or command with root privileges.		
Potential vulnerabilities	Business and logic flaws. Insecure authorisation. Authentication bypass.		
Impact	Privilege escalation, Complete system compromise.		

ID	TE_2	**Threat type**	Elevation of Privilege
Elements affected	Applications and Services (P_{SW})		
Description	An attacker escalates privileges to run a service or application with increased rights.		
Potential vulnerabilities	Business and logic flaws. Insecure authorisation.		
Impact	Privilege escalation, Complete system compromise.		

Finally, to give a clear picture of how the STRIDE threats affect the telepresence robot's system, Table 10 shows an overview of the elements of the data-flow diagram and the threat types affecting them.

Table 10. STRIDE model of the data-flow diagram.

External Entities

ID	Name	S	T	R	I	D	E
E_U	User	✓		✓			
E_A	Administrator	✓					

Processes

ID	Name	S	T	R	I	D	E
P_{HW}	Hardware Peripherals		✓		✓		
P_{OS}	Operating System				✓	✓	✓
P_{SW}	Applications and Services	✓	✓		✓	✓	✓
P_{CS}	Cloud Service	✓			✓		

Data Stores

ID	Name	S	T	R	I	D	E
DS_S	System Storage		✓		✓		
DS_C	Cloud Storage						

Data Flows

ID	Name	S	T	R	I	D	E
DF_1	Actuator commands						
DF_2	Sensors' data						
DF_3	Service commands						
DF_4	Status updates		✓			✓	
DF_5	Logs		✓			✓	
DF_6	Call parameters		✓			✓	
DF_7	Robot's camera feed					✓	
DF_8	Control commands		✓			✓	
DF_9	Call initialization		✓			✓	
DF_{10}	User's camera feed					✓	
DF_{11}	Administrative and Commands data		✓		✓	✓	
DF_{12}	Database accesses						

5.3 Risk Scoring

This section presents the risks associated with the threats identified during the threat modelling phase. The risk score is calculated for each threat based on three independent variables with values ranging from one to three. The first variable, Impact, rates the potential impact of a successful attack on the system. Coverage measures which users would be affected by the threat, whether it is an individual, a group, or all users in the system. Most threats affect a group of users since the compromise of the robot would affect a group of users rather than all users in the cloud provider. Only in the case a

single user account is compromised it would have the value one. The third variable, Simplicity, ranks the ease of the attack. All risk scores are computed following the same procedure. For a better understanding, the threat TT_2 is used as an example to show how the computations are performed. A value of three was assigned to the Impact field as it enables remote code execution. Coverage was assigned the value of two since it affects the group of users that have access to the robot. Simplicity received the value one since this threat requires modifying the code of the application to be tampered with malware. The risk score is computed according to its formula, Risk = (impact + coverage + simplicity * 3)/5 = (3 + 2 + 1*3)/5 = 1.6. The risk scoring methodology is explained in detail in the section 2.2. Table 11 displays the results of risk scoring on all the identified threats.

Table 11. Risk Scores.

ID	Impact	Coverage	Simplicity	Risk	Description
TI_1	2	2	3	2,6	An attacker can interact with the screen to reveal sensitive information about the robot.
TT_1	2	2	3	2,6	An attacker can interact with the robot’s screen to alter the device's configuration.
TI_2	1	2	3	2,4	An attacker can scan the open ports of the robot to gather information about its usage.
TI_3	3	2	2	2,2	An attacker connects to the operating system, having access to critical information.
TI_5	3	2	2	2,2	An attacker gains access to the robot’s filesystem, having access to files, including drivers and configuration files.
TS_3	3	2	2	2,2	An attacker spoofs the administrator's identity to connect to the robot locally through SSH or ADB.
TI_6	2	2	2	2	An attacker intercepts shell commands disclosing confidential information such as credentials or configuration steps.
TT_3	2	2	2	2	An attacker tampers files installed in the system to alter configuration and credentials.
TD_1	1	2	2	1,8	An attacker runs scripts on the system, causing it to shut down or reboot.
TD_3	1	2	2	1,8	An attacker disrupts the service of a relay call connection.
TD_4	1	2	2	1,8	An attacker disrupts the service of a local call connection.
TD_5	1	2	2	1,8	An attacker disrupts the service of an administrative shell.
TI_4	1	2	2	1,8	The cloud service discloses confidential information.
TR_1	1	2	2	1,8	A user connects via ADB or SSH to take control of the robot without getting logged or traced.
TT_2	3	2	1	1,6	An attacker installs a modified version of an application with malicious code, enabling it to gain control of the robot, manipulate its behaviour or steal sensitive data.
TT_6	3	2	1	1,6	An attacker captures and modifies shell access to execute arbitrary commands on the robot.
TE_1	2	2	1	1,4	An attacker escalates privileges to run a terminal or command with root privileges.
TE_2	2	2	1	1,4	An attacker escalates privileges to run a service or application with increased rights.
TS_2	2	2	1	1,4	An attacker spoofs the identity of the cloud to act as a man-in-the-middle between the robot and a user.
TS_4	2	2	1	1,4	An attacker spoofs the identity of the robot.
TT_4	2	2	1	1,4	An attacker alters the status updates or logs sent to the cloud to modify the perceived state of the robot.
TT_5	2	2	1	1,4	An attacker modifies the call parameters or initialization commands, resulting in an incorrect call establishment.
TD_2	1	2	1	1,2	An attacker exploits a vulnerability, crashing a running service or application.
TS_1	2	1	1	1,2	An attacker spoofs the identity of a legitimate user to connect to the robot in an unauthorised way.

6 Exploitation

This section explains the Exploitation phase, which transitions from theoretical threat modelling to the practical assessment of identified vulnerabilities within the telepresence robotics system. The aim is to attempt to exploit the vulnerabilities found for the threats discovered. The Exploitation phase is characterised not only by the execution of targeted attacks but also by meticulous planning. The planning involves preparing the attacks and establishing a methodology for the threats that are tested along with the methods and attack vectors used to attempt the exploitation.

6.1 Planning

The threats were attempted to be exploited in order of their risk score. This ensures that the threats that pose the greatest danger to the system are analysed first, based on their impact, the number of affected users, and the ease of the attack. However, if a test is successful and results in new attack vectors for another risk, new tests are considered. This means that if a test discovers new vulnerabilities in the system, they are also taken into consideration. This can result in a chain of tests aimed at taking total control of the telepresence robot.

The analysed threats are diverse in nature. Some tests involve simple actions or checks, while others require highly advanced methods. When a vulnerability needs to be exploited, we use the Exploit-DB database[1] by OffSec and the Metasploit database[2] from Rapid7, which provide valuable up-to-date knowledge of vulnerability exploits. They include valuable up-to-date knowledge of vulnerability exploits. In addition, the testing guides provided by NSE Lab, the developing team of PatrIoT, have been used. The guides provide steps and methods to attack the different attack surfaces outlined in the methodology.

[1] URL: https://www.exploit-db.com/

[2] URL: https://www.rapid7.com/db/

The initial tests developed were those that assessed the risk of information disclosure (TI_1) and tampering (TT_1) of the robot's screen, given a risk value of 2.6. The tests aimed to obtain as much information as possible from the robot's screen and to modify settings that could compromise its cyber security. Subsequently, the tests focused on the information disclosure caused by an open port in the robot when a call is established (TI_2).

With a risk value of 2.2, the next planned tests attempted to spoof the robot's identity in order to gain shell access (TS_3) and gather information from both the operating system (TI_3) and the filesystem (TI_5). If successful, gaining shell access to the robot greatly simplifies the process of gathering information from the operating system and filesystem. In the designed testing, the tests related to the threats TI_3 and TI_5 are dependent on the success of the tests based on the spoofing threat (TS_3), which would provide shell access. This same precondition behaviour applies to other tests.

The tests with a risk value of 2 were designed with different preconditions. The trials involving the tampering of files in the system (TT_3) also require shell access. However, tests involving the disclosure of shell commands (TI_6) are exempt from this condition and can be executed independently. Different tests were created with a risk value of 1.8 to attempt denial of service on the robot (TD_1), relay call (TD_3), local call (TD_4), or administrative shell (TD_5). The denial-of-service attempts have been designed with the precondition of having shell access, but they do not require it as they can be accomplished from a network perspective.

Even though the cloud is not directly targeted, a test for user enumeration was also considered under the threat TI_4. Tests were also planned for log repudiation threats (TR_1) with the same risk value. Additionally, tests with a risk value of 1.6 were conducted for the installation of malicious applications (TT_2) and the capture and modification of shell commands (TT_6).

The tests are designed to target the sixteen threats with a higher risk value, ranging from 2.6 to 1.6. Those threats with a lower risk, with values ranging from 1.2 to 1.4, are not considered to represent a major risk for the robot since none of them have a high impact, and they are highly complex to execute. If the planned tests are unsuccessful

and no real threat to the robot's system is discovered, new tests will be considered, even targeting those threats with a lower risk.

Figure 5 displays a testing tree that illustrates the results of the planning stage. The square boxes represent the set of tests that have been designed, and the arrows connecting them show the dependencies. For instance, the success of the tests under the group “Robot drivers” depends on the success of the “Extract firmware” tests. Additionally, each set of tests has an identifier of the targeted threat assigned to the top-left corner of its square. For the sake of avoiding repetition, a more detailed explanation of each set of tests can be found in the Execution section.

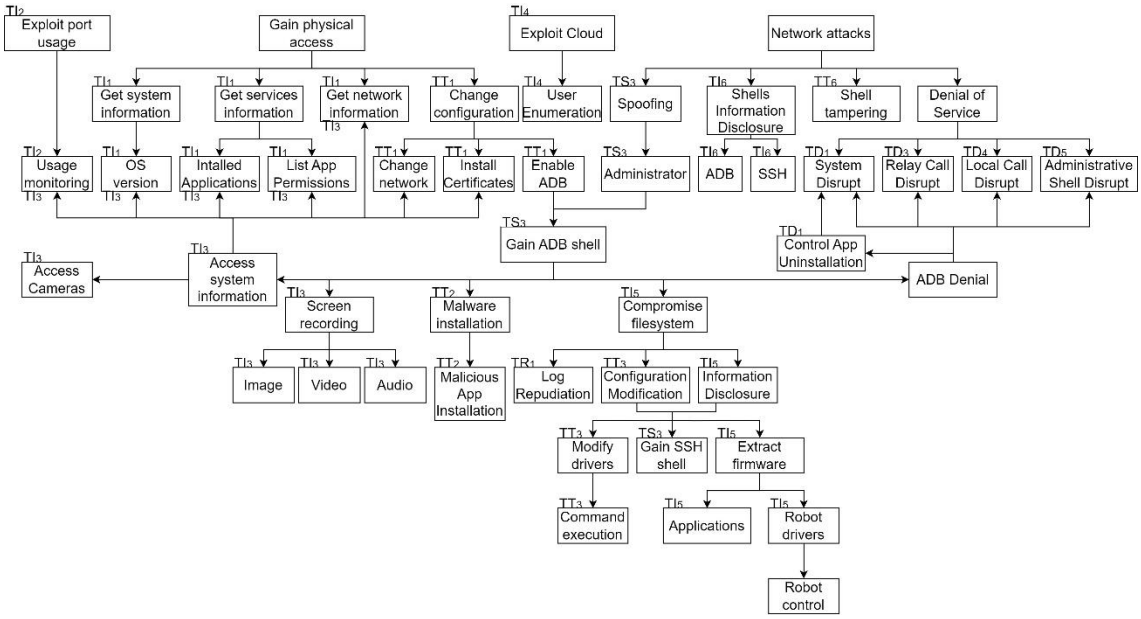

Figure 5. Test tree planning.

6.2 Execution

This section provides information on the tests conducted to exploit the robot's system. As in the rest of the thesis, it is important to note that the level of detail has been adjusted to protect the privacy of the device and its manufacturer.

The initial tests involved obtaining information about the robot by interacting with its screen, which required physical access to the device. Since the screen is being operated by the Android operating system, the status bar can be opened by swiping down from the top of the screen, just like any other smartphone. In that menu, clicking on the gear icon opens the Android settings application, where information about the system is found. The Android version and its security patch level were successfully retrieved, and

a list of installed applications, along with their access permissions, was compiled. Only three services differed from a stock Android installation: a camera application, a utility to control the robot's parameters and a proprietary application.

The Wi-Fi connectivity was also analysed, and it was possible to retrieve a list of Wi-Fi hotspots that are saved in the robot, listing the Wi-Fi names to which the robot has connected before. Details about the current Wi-Fi connectivity, including the IP and MAC addresses, were also retrieved.

The subsequent tests involved modifying the robot's configuration through its screen. Initially, we attempted to enable Android Debug Bridge (ADB) to allow remote shells to access the device. This was achieved effortlessly by tapping the Android version seven times to activate developer settings and then selecting the ADB checkbox. The subsequent test aimed to connect the robot to a hotspot of choice via Wi-Fi. The procedure for connecting to a Wi-Fi network is the same as on other devices. A Wi-Fi name is selected within the list of networks in reach, and credentials are entered if needed.

During the last test, a Certificate Authority certificate was installed on the device. These certificates are used to establish trust with servers and can be exploited in various attacks. To generate the certificate, the HTTP Toolkit was used, and it was then transferred to the robot's storage. The robot's browser can be used to access a website that serves the file. After downloading the file, it can be installed on the robot through the settings application, specifically in the Security and Credential section. All screen-related tests (threats TI1 and TT1) were successful, allowing for the retrieval of valuable system information and the implementation of necessary changes for future attack vectors. Enabling the ADB shell is a crucial achievement as it serves as a prerequisite for many other tests planned for the device, as demonstrated in Figure 5. These are the only tests aiming at the hardware attack surface.

Next, tests were conducted to address usage monitoring based on open ports. For this test, the robot must be reachable on the network by the device used to perform the test. This was not an issue as the robot is currently connected to an open Wi-Fi network in the building where it operates. The robot's IP address is required, but it can be obtained using Nmap. Executing Nmap with the flags '*-A -sV -sC*' allows for easy identification

of the robot's address among the list of connected devices on the Wi-Fi network, making use of the information displayed about the operating system and hostname. Once the target IP address is obtained, a full port scan is executed using the flag '*-p-*'.

Three scenarios were tested, each with different iterations, to ensure consistency. The test results indicate that the robot consistently has port 22/TCP open to manage the SSH server. The ADB protocol determines whether port 5555/TCP is open, meaning that the ADB service is running. Additionally, depending on the robot's current use, an extra port may be open. If no additional port is open, the robot is in standby mode and the video feed is not being transmitted. If the robot is in use, it opens a random port, typically above 50,000. Regardless of the call mode, both local and relay connections exhibit the same behaviour. This test has been proven to successfully determine whether the robot is in use or on standby to any device connected to the same Wi-Fi network.

Although the risk value of the threats was used to determine their order of execution, they are presented in this section in a more coherent manner by grouping similar tests together for easier comprehension. As ADB was successfully enabled in previous tests, it is now possible to attempt to establish a shell connection. This process requires the installation of the Android SDK Platform Tools provided by Google for Developers. After installation and a connection to the same network as the robot is established, it is possible to spoof an administrator and connect to the robot.

ADB lacks any security measures, such as authentication or authorisation by default. Although some other telepresence robot vendors have implemented their own protection methods, ADB is completely vulnerable in this device, and any device on the same Wi-Fi network can freely connect to it. Doing so involves running the command '*adb connect <IP>:5555*' and then '*adb shell*', which opens a new shell terminal on the robot. Once a terminal connection is established, ADB commands can be used to perform the same tests described earlier but with no interaction with the screen.

The command '*getprop*' provides technical information about the device, including hardware architecture, Android version, security patch updates, and the version of Ubuntu that is part of the hybrid operating system. Regarding network connectivity, the command '*dumpsys wifi*' provides a log of the Wi-Fi connectivity, including basic information such as the SSID of the currently connected Wi-Fi, MAC and IP addresses.

Furthermore, the details displayed include the BSSID of the networks in reach (the MAC addresses of the hotspots) and the complete log of Wi-Fi connections. Upon reading the contents of the file at '*\data\misc\wifi\wpa_supplicant.conf*', a list of saved Wi-Fi connections and their corresponding credentials are revealed. It is also possible to force the device to connect to another Wi-Fi hotspot, which is useful for performing network attacks. The command '*wpa_cli -p /data/misc/wifi/sockets/ -i wlan0*' can be used to list the identifiers of the networks in range, followed by 'select_network <ID>' and 'enable_network <ID>' to force the connection.

The Mitmproxy documentation [100] provided clear steps for installing a system's Certificate Authority certificate via ADB. To obtain a list of all applications installed on the device, identified by their package name, we used the command '*adb shell pm list packages*'. The command '*adb shell dumpsys package <package_name>*' can be used to retrieve the permissions granted to different applications.

Additionally, usage monitoring using the ADB shell was conducted to determine if the robot was streaming video. The initial test involved querying the application running on the screen, with the hope of detecting a difference between the robot being in standby mode and having an established call. As the results did not differ, a second test was conducted to obtain usage based on the executed services. The command '*top*' displays the processes in use. When a call is established, the processor experiences a significant increase in usage, particularly on the camera and audio servers, as well as the proprietary app that controls the robot's functions.

The following tests aim to breach the privacy of both the robot and its users. The initial set of tests attempted to gain access to the robot's camera, enabling an attacker to view its surroundings. There are numerous online sources detailing how to access the camera using ADB commands. The method is the same in all cases. It involves making three calls to the operating system: the first to start the camera activity, and the second and third to force focusing and capture a picture or video. However, this well-known method failed, and additional tests were conducted to try to achieve the desired outcome. Initially, the proprietary app was terminated as it was believed to be interfering with the camera, but even after its closure, images could not be captured. A camera application was then installed to capture photos in a traditional manner, but clicking on the shutter icon did not save any images.

The following test aimed to capture a screenshot of the device while the camera application was running. Using the command '*adb exec-out screencap -p > <filename>*', an image of the device's screen was taken. If an attacker tries to access the robot's camera, it is possible to launch the camera application and take a screenshot of the feed displayed on the screen. However, anyone in front of the robot would notice that the screen is displaying the camera feed. For this reason, the test "Access Cameras" is marked as partially successful. In addition, the screenshot utility can also be used during a call between the robot and a user. In that scenario, both the user's camera and the robot's cameras are shown on the screen. This poses a threat to the privacy of the system as taking a screenshot captures both the user's and robot's images.

A similar command has been proven to work on other Android devices, allowing for the capture of a video of a smartphone's screen. Applied to the telepresence robot, this would record a video of the screen instead of a static image. The command '*screenrecord*' was tested for this purpose, which, in combination with VLC player, can transmit the screen in real. Testing this command with different arguments, applications and scenarios was always unsuccessful.

By default, ADB does not provide a method for recording audio. However, it is theoretically possible to install another application to achieve this. A test was designed to install malware on the robot, which included the ability to record audio. The tools required to generate and control this malware are only available in Kali Linux, so this operating system was added to the testing equipment on an external hard disk. In this environment, the tool MsfVenom is used to generate code that is installed in the robot. The tool specifies the connection arguments and uses the payload '*android/meterpreter/reverse_tcp*'. This generates an Android package with the extension ".apk". The package can then be remotely installed on the robot using the ADB command '*install <APK_file>*'. Following installation, a listener was set up to connect the app to the controlling device upon its execution. We used the Metasploit tool with the same payload specified for MsfVenom.

Finally, the application needs to be executed. This is usually accomplished by manually opening it in the attacked device, but since the robot's screen doesn't allow to open applications, the ADB command "*monkey*" launched the app remotely. Upon launching the app, it automatically connects to the listener and opens a shell on the attacker's

device. This shell, specifically tailored for Android, offers custom commands that can interact with the robot or open a new shell on it, even if the ADB setting is disabled.

The custom commands that were useful for the designed tests were related to video and audio recording. The "*screenshare*" command enables an HTTP website on the device to stream the contents of the screen in real time. The '*record_mic*' command should record the audio captured by the robot's microphones. Finally, the commands '*webcam_snap*' and '*webcam_stream*' should capture images and videos using the robot's built-in cameras. However, all these commands failed. Although the installed malware can obtain other information about the device, it cannot access the cameras, audio, or screen.

In the last attempt, the same goal was attempted using TeamViewer, a software that allows remote control of Android devices. The program was designed to display the robot's screen on a controlling computer and simulate user input to take control of the device. After installing the application on both the robot and the computer, a PIN code is used to pair them. However, the connection failed to start, and remote control was not possible. These results show that access to the cameras or microphones has been restricted on this device. It is assumed that this behaviour is either a security measure or an accidental conflict with other background services being used simultaneously.

The following tests aim to compromise the filesystem, according to the threats of disclosing its information (TI5) and tampering with its files (TT_3). As an ADB shell was obtained by spoofing the administrator's identity, it will be used to access the filesystem. ADB provides typical Linux commands to interact with the files stored in the device. The '*ls*' command can be used to list files, '*cd*' to traverse folders, and '*cat*' to print file contents. Additionally, it supports the text editor Vi. These commands enable manual analysis of the files installed on the system by traversing key folders such as '*/data/ssh*'.

This process can be automated using Firmwalker, a utility that traverses the firmware file system to identify potentially interesting data from a cybersecurity perspective. The software searches the entire filesystem for hardcoded credentials or scripts that could potentially be exploited. The results revealed various certificates and key files, including those utilised for SSH connectivity. Lastly, it enumerated IP addresses, URLs, and emails referenced in the file system, which could be exploited to attack the Cloud

attack surface. However, it is important to note that this is beyond the scope of this thesis.

As part of the information disclosure process, we obtained the firmware responsible for the robot's features. This involved extracting the driver files and the controlling application for static analysis. To generate a list of installed applications, we used the command '*pm list packages*'. The robot's logic is managed by a single application that handles call connections and controls its physical features. To locate the path of the application's '.apk' executable file, the command '*pm path <package_name>*' was used. This file can then be retrieved using '*adb pull <path>*' to extract it to another device for its analysis.

The '*backup*' command can be used to retrieve other associated data, such as cache and configuration. This command enables the application to be cloned and installed in a virtual device created with Android Studio for dynamic analysis using device emulation. However, due to the inability to retrieve the backup file, only static analysis could be performed on the extracted application.

To access the source code, the extracted file needs to be decompiled. The Apktool utility, which specialises in reverse engineering Android apk files, was used for this purpose. The disassembly's success allowed for analysis of the app's behaviour. Despite the challenge of analysing over 300,000 lines of code, the process revealed how the app establishes connections and controls the robot's features, such as its movement motors, lights, and speakers.

The '*logcat*' command can be used to retrieve Android's logs. This log provides information about executed files and other event logs that offer insights into the system's behaviour. The extracted information was used to generate new tests based on the new possibilities that emerged from these files. Since most tests require access to an ADB shell to succeed, we focused on attempting SSH connectivity. The initial tests aimed to utilise the SSH information extracted from the file system. Despite retrieving the robot's SSH server's public and private DSA and RSA keys, they couldn't be used to force a connection. This was due to the SSH configuration that enforced the use of the Elliptic Curve Diffie-Hellman (ECDH) key exchange, resulting in the generation of new keys for each connection. Furthermore, the leak of keys in the '*authorized_keys*' file

could not be used to establish new connections due to the absence of private keys needed for the process. Nevertheless, it was possible to add custom keys by modifying the file. Once new RSA keys were generated using the utility PuTTYgen, the public key was added to the authorised keys using the '*echo <key_content> >> authorized_keys*' command. After adding the key, it was possible to gain access to the robot by using the newly generated private key to connect to the user "root" via SSH. This ensures that even if ADB is disabled, a potential attacker could still maintain access to the device.

Another method of taking control of the robot is through command execution. During the analysis of the file system, we found a file with a '.js' extension that was executed by the root user during the robot's boot-up process. This file was altered to include the execution of custom code. The file was modified to execute custom code using the '*execSync*' function, which allows shell commands to be executed with the specified shell path *('/system/bin/sh*'). This method presents a powerful attack vector as it enables the modification of files that are automatically executed by a root user.

Finally, the drivers obtained from the filesystem and reverse engineering of the control application have opened new attack vectors to attempt to gain physical control of the robot. After analysing the behaviour of the drivers, it was possible to replicate it by manually calling the different functions that control the robot's peripherals, thus granting full control over the robot. Some of the allowed actions include moving with its motors, adjusting lighting, checking battery levels, and utilising Text-to-Speech to read text aloud through its speakers.

As stated in the Threat Modelling section, it is necessary to analyse repudiation threats from a theoretical perspective due to the uncertainty surrounding the data sent from the robot to the cloud. The administration console displays recent user connections to the robot and the last SSH accesses. The robot's user manual documents that the installation of new SSH keys must be done through this console. However, it was discovered during the testing phase that if a malicious SSH key is installed in the robot using ADB commands, it will not be logged into the system. The administration cloud service does not show the malicious key or the SSH accesses where it is used. Our tests also revealed that the same behaviour applies to ADB shells, where accesses are not logged on the cloud's endpoint but only locally. For a more complete repudiation, even though they seem to not be transmitted to the cloud, the local logs stored in the robot were also

deleted. The ADB '*logcat -c*' command deletes all logs from the device. An administrator could notice that the logcat's buffer is empty, revealing some unauthorised access, but all the actions performed before wiping the logs are not traceable.

The following tests targeted various methods for denying service to different features of the telepresence robot's system. Based on the results of the threat modelling, the attempts focused on disrupting relay calls (TD_3), local calls (TD_4), administrative shells (TD_5) and the overall system (TD_1). Two main methods were tested: denying service through ADB commands and through network attacks that do not require a shell.

The initial tests attempted to deny service to the robot as a whole. To stop the robot from working, the simplest method is to reboot or power it off. This can be achieved using the '*reboot*' and '*reboot -p*' commands. It is important to note that these commands require a new shell connection on every reboot for a continuous denial of service. However, to ensure a permanent denial of service, the executable file discovered during the filesystem tests should be used. Custom code was injected into the file to create a function rebooting the system. The function is called from an asynchronous timer with a two-second delay to ensure all components are loaded. This code causes the device to enter a boot loop, rendering the robot unusable and denying service.

An alternative method to disable the robot's service using ADB commands is to uninstall the application that controls all of its features. The command '*uninstall <package_name>*' can be used for this purpose. But in this case, the damage is not reversible since it wasn't possible to back up the application. As this thesis aims to avoid irreversible damage, this test is only theoretical and has not been executed.

The tests carried out to disrupt the service of both local and relay calls are the same. When a call is established, the operating system creates a new process to manage it. To terminate the call, an ADB command was created to list all processes, filter them based on the '*ARGS*' property to isolate the desired one, and use the '*kill*' command to terminate the process. The robot automatically disconnects, causing the user controlling it to receive error connection messages for several seconds until the call is finished. Local and relay connections were successfully terminated using the same procedure.

Killing processes is also used to perform denial-of-service attacks on administrative shell connections.

Four scenarios were considered, covering all possible connections. Both the attacker's shell and the shell to be denied can use either ADB or SSH protocols. Android only allows one ADB shell connection at a time. Therefore, if an attacker wants to deny the service of an ADB shell, they need to connect to the robot via SSH to perform the attack. Once connected, terminating the running process by filtering for the string '*adb*' instantly terminates the shell connection. After terminating the process, a new one is executed, and reconnection is possible.

To deny service for an extended period, the attacker must connect to the robot using ADB, preventing the legitimate administrator from connecting due to the limitation of one concurrent connection. Denial of an SSH shell can be accomplished by an attacker connected by ADB or SSH. When an SSH connection is established, a new child process is created to manage each shell in addition to the process responsible for the SSH server. This means that if there are two concurrent connections, there will be three processes managing SSH: one main process with the server and two child processes, each responsible for one shell. To avoid disconnection of the attacker, the 'kill' command should be used to terminate only the desired child process instead of terminating the main process. To achieve this, in addition to filtering for only ssh processes, further processing is necessary to select the specific one. These processes have their PID identifiers in chronological order, so it is important to correctly identify the other shell. If the attacker connected first, a command can be executed recursively to terminate SSH connections with a higher identifier. Otherwise, a command is executed to disconnect all SSH connections except for the one used for the attack, and then the recursive command is scheduled.

Additional tests were designed to achieve denial of service with network attacks, even if shell access is not granted. All of the proposed network attacks require being in the Wi-Fi range of the robot or client or connected to their networks on LAN. No method was found to deny the robot's service altogether but only to isolate it from its connectivity, thus denying its basic functions. This method was successful in disrupting the service of relay and local calls and administrative shells.

Two different approaches were tested, depending on the attacker's access to the network. In the first, the attacker is connected to or has access to the same Wi-Fi network. This is the scenario used for this research, as the robot analysed is connected to a public Wi-Fi. For the first scenario, the tool Bettercap in Kali Linux was used. The attack vector is based on ARP spoofing, where the attacking computer sends fake ARP packets linking its MAC address to the IP of the robot or client. Once the commands to set up ARP spoofing have been executed, it is possible to stop traffic on the target by setting the '*arp.ban on*' property. While this attack is being executed by Bettercap, the target device (either the client or the robot) is isolated from the network, interrupting current calls or shells. The four different scenarios tested, relay and local calls and ADB and SSH shells, were all successfully denied.

In the second approach, the attacker has no access to the network. In other words, the robot is connected to a Wi-Fi network whose credentials are unknown. This test uses the Aircrack-ng tool, which is also included in the Kali Linux operating system. Another requirement for this attack is a wireless network adapter that supports monitor mode. Once the set-up is configured, the command '*airodump-ng*' can be used to list all access points in range. This will locate the Wi-Fi network to which the robot is connected and retrieve the hotspot details. Using the hotspot's MAC address and Wi-Fi channel, the same utility can be used to monitor devices connected to the hotspot. As there may be several devices connected to the same hotspot, the MAC address of the robot must be filtered. This can be done based on previous tests, by proximity based on the strength of the signal, or by collecting all MAC addresses and filtering based on the manufacturer of the device. Tools such as the previously used Fing or the online MACVendors can be used to determine the manufacturer from a given MAC address.

With both MAC addresses (robot and access point), it is possible to launch a Wi-Fi de-authentication attack using the '*aireplay-ng*' command. This sends disassociate packets, forcing the robot to disconnect from the Wi-Fi hotspot and resulting in denial of service for calls and administrative shells. The process explained above was aimed at isolating the robot from the network, but it is also possible to isolate the client running the same test using the client's MAC address instead. The choice depends on the network topology of the attacker.

A combination of the two previous attacks can be used to perform shell information disclosure (TI_6) tests. Man-in-the-middle can be used to expose information flowing from the robot to the client, in this case, administrative shell commands. This method can only be used if the attacker has access to the Wi-Fi network where the robot or client is located. Figure 6 shows the network topology of a local connection used for local calls or shell connections. The left part of the figure shows the normal topology when no attack is performed, and the right part shows the scenario that was achieved by this attack.

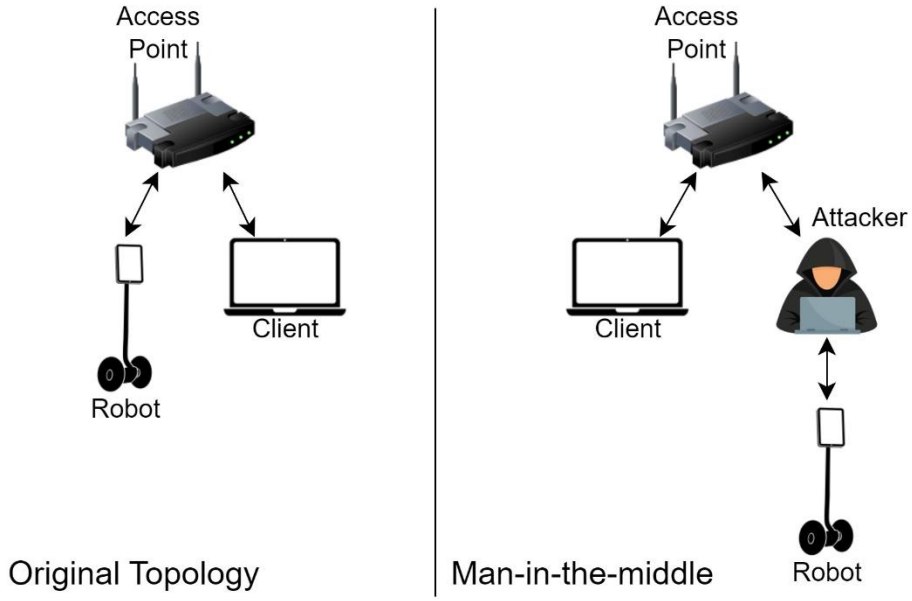

Figure 6. Man-in-the-middle attack.

The first step in the test was to create a Wi-Fi hotspot for the robot to connect to. This test was carried out in Kali Linux, where the 'Linux Wifi Hotspot' utility was installed. Once executed, its configuration menu can be used to enter the credentials of the Wi-Fi to be created, and the "gateway" checkbox must be activated so that the clients connecting to it can access the Internet. Once the hotspot is up and running, it is necessary to force the robot to connect to it. This approach builds on the previous tests by using a Wi-Fi de-authentication attack to force the robot to disconnect from the Wi-Fi. Since the robot cannot connect to its previous access point, it tries to connect to the same Wi-Fi network served by a different access point. If the new connection is within range of the robot, it will successfully connect to the compromised access point.

With this new network topology, it is possible to capture and analyse the network traffic between the robot and the client using Wireshark. Different iterations of this test were

able to capture traffic generated by SSH and ADB shells, as well as local and relay calls. The traffic captured from the ADB shell showed all interactions in clear text. It was possible to retrieve all commands sent by the administrator along with the robot's responses. In terms of SSH, the captured traffic was encrypted using Elliptic Curve Diffie-Hellman Key Exchange, so none of the attempts to decrypt the content were successful. All traffic in the intercepted calls is encrypted, either using TLS for commands, status and logs, or the UDP protocol with an encrypted payload for camera feeds. This encryption prevented any information from being retrieved from the connection and ensured privacy.

The same network topology described in the previous attack can be used to perform shell tampering, an attack derived from the TT_6 threat. The tests performed attempt to modify the shell commands from a network perspective, which would mean interacting with the shell in an unauthorised way. As SSH is encrypted and secure, we didn't find any tests that would succeed in this type of attack. For the ADB connection, several tests were run to try to make this attack successful. Previous literature has shown that these attacks can be performed using Burp Suite. This program can't access TCP traffic by default, as it usually targets HTTP and HTTPS connections, but Burp-Non-HTTP-Extension can be installed to include this feature. After trying various configuration methods, we were unable to capture the traffic using this program.

The next approach was to use custom Python code, including the Scapy and NetfilterQueue utilities. The program developed was able to capture the traffic and, in theory, manipulate its contents to alter the commands, but when the tests were run, the contents of the shell weren't actually altered. An additional test used Ettercap to accomplish this purpose. An additional test used Ettercap for this purpose. First, a filter was created with instructions on how to manipulate the TCP packets, with strict rules specifying the changes that would be made. This filter was then compiled using the command '*etterfilter <input_filter> -o <output_file>*'. This filter was then used in the '*ettercap*' command, specifying the interface used. An ARP attack was then selected, although no devices were spoofed, as otherwise, the ADB commands would not be successfully modified. After this attack was executed, the ADB shell displayed modified commands according to the specified filters, proving the success of this test.

The last test attempted to retrieve information from the cloud provider, under the threat TI_4. During the development of this thesis, we noticed that the cloud service is not properly protected against user enumeration. In the test carried out, going to the login page of the cloud provider and clicking on “Forgot password” allows entering an email to reset the password. If the email exists, a confirmation message is displayed, otherwise an error message appears. This behaviour allows an attacker to check whether an email is registered on the robot's online platform. This information isn't useful on its own, but when combined with social engineering, it can form part of an attack vector. Although this type of attack is not considered in this thesis, it is usually good practice not to reveal whether an email is registered with a service.

Figure 7 shows an overview of the test results. The tests or categories that were successful are marked in green; since access to the cameras was only partially successful, they are marked in yellow; and the unsuccessful tests are marked in red. Finally, the grey colour is used for the “Control App Uninstallation” tests, as they couldn't be carried out in order not to destroy the robot, as stated in the limitations section.

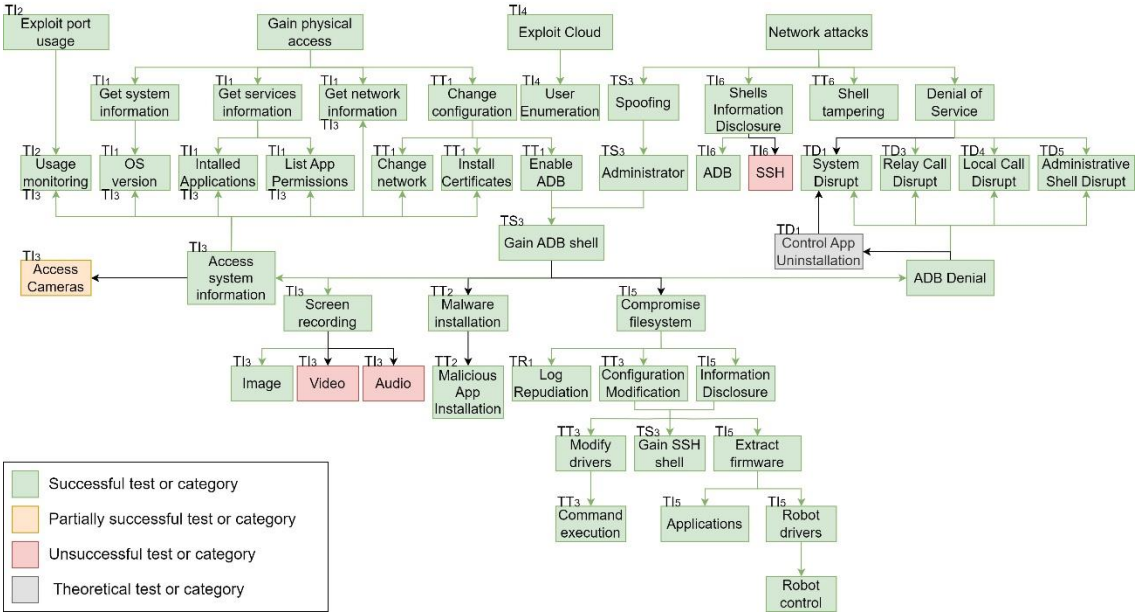

Figure 7. Test tree results.

7 Reporting

The results of the execution section have been documented so that the details can be shared with both the robot manufacturer and the Creativity Matters research group[1], who own the robot used for this research. The content of the report includes the data documented in this thesis and additional information with the exact steps performed in each of the tests so that they can be replicated. Due to the limitation in this thesis to safeguard the privacy of the manufacturer by not revealing the brand and model of the robot, these details can't be appended to this document. We have considered that the information included in the execution section is the appropriate amount to give an overview of the work carried out while ensuring that identification of the robot is not possible. However, a brief summary of the results, vulnerabilities discovered and the tools used to map them to the threat analysis is given in Table 12. This summary follows the structure used to share the results of the tests attempted on the device, grouped by the threat from which they originate.

Table 12. Report summary.

Threat	Information Disclosure (TI_1)	**Success**	Yes
Details	Information retrieved from the screen of the robot: Android version, security patch level, installed applications, application permissions, saved Wi-Fi hotspots and IP and MAC addresses.		
Method	Input in robot's screen.		
Vulnerabilities	Lack of authentication on the touchscreen's settings.		

Threat	Tampering (TT_1)	**Success**	Yes
Details	Settings altered through the screen of the robot: ADB shell, Wi-Fi hotspot used and Certificate Authority certificate installation.		
Method	Input in robot's screen, HTTP Toolkit.		
Vulnerabilities	Lack of authentication on the touchscreen's settings.		

[1] Creativity Matters research group at TalTech IT College. URL: https://cm.taltech.ee/about

Threat	Information Disclosure (TI_2)	**Success**	Yes
Details	Monitor the robot's usage based on open ports.		
Tools	Nmap.		
Vulnerabilities	Exposure of information.		

Threat	Information Disclosure (TI_3)	**Success**	Yes
Details	Information retrieved from establishing a shell with the operating system. Requires the success of threat TS_3. Information retrieved: Usage monitoring, Android version, security patch level, installed applications, application permissions, saved Wi-Fi hotspots, saved Wi-Fi passwords, IP and MAC addresses, screen capturing, Malware installation, filesystem access, log repudiation and denial of service. Failed to record video from the screen and audio through the microphones. Partially succeeded in capturing images from the camera.		
Tools	Android SDK Platform Tools (ADB).		
Vulnerabilities	Inadequate ADB shell privileges.		

Threat	Information Disclosure (TI_5)	**Success**	Yes
Details	Retrieval of the filesystem, controlling application and drivers. Requires the success of threat TS_3. Enabled physical control of the robot.		
Tools	Firmwalker, Apktool, Android Studio, Android SDK Platform Tools (ADB).		
Vulnerabilities	Inadequate ADB shell privileges.		

Threat	Spoofing (TS_3)	**Success**	Yes
Details	Spoofing an administrator to establish a shell to the robot's system.		
Tools	Android SDK Platform Tools (ADB).		
Vulnerabilities	Lack of authentication on ADB shells.		

Threat	Information Disclosure (TI_6)	**Success**	Yes
Details	Interception of shell traffic using man-in-the-middle attack.		
Tools	Kali Linux, Linux Wifi Hotspot, Bettercap, Aircrack-ng, Fing, MACVendors, Wireshark.		
Vulnerabilities	Lack of encryption on ADB shells.		

Threat	Tampering (TT_3)	**Success**	Yes
Details	Tampering files in the operating system to modify the robot's drivers, command injection and installing an SSH backdoor. Requires the success of threat TS_3.		
Tools	PuTTYgen, custom code, Android SDK Platform Tools (ADB).		
Vulnerabilities	Inadequate filesystem privileges.		

Threat	Denial of Service (TD_1)	**Success**	Yes
Details	Disruption of the overall system. Requires the success of threat TS_3.		
Tools	Android SDK Platform Tools (ADB).		

Threat	Denial of Service (TD_3)	**Success**	Yes
Details	Disruption of service of relay calls.		
Tools	Kali Linux, Bettercap, Aircrack-ng, Fing, MACVendors, Android SDK Platform Tools (ADB).		

Threat	Denial of Service (TD_4)	**Success**	Yes
Details	Disruption of service of local calls.		
Tools	Kali Linux, Bettercap, Aircrack-ng, Fing, MACVendors, Android SDK Platform Tools (ADB).		

Threat	Denial of Service (TD_5)	**Success**	Yes
Details	Disruption of service of administrative shells		
Tools	Kali Linux, Bettercap, Aircrack-ng, Fing, MACVendors, Android SDK Platform Tools (ADB).		

Threat	Information Disclosure (TI_4)	**Success**	Yes
Details	User enumeration on the cloud provider.		
Tools	None.		
Vulnerabilities	Exposure of information.		

Threat	Repudiation (TR_1)	**Success**	Yes
Details	Log and usage repudiation. Requires the success of threat TS_3.		
Tools	Android SDK Platform Tools (ADB).		
Vulnerabilities	Inadequate ADB shell privileges.		

Threat	Tampering (TT_2)	**Success**	Yes
Details	Installation of malware capable of accessing the device's features.		
Tools	Kali Linux, MsfVenom, TeamViewer, Metasploit, Android SDK Platform Tools (ADB).		

Threat	Tampering (TT_6)	**Success**	Yes
Details	Tampering of shell commands resulting in arbitrary code execution.		
Tools	Kali Linux, Burp Suite, Burp-Non-HTTP-Extension, Python, Scapy, NetfilterQueue, Ettercap		
Vulnerabilities	Lack of encryption and integrity checks on ADB shells.		

Potential improvements to enhance the cybersecurity of the analysed telepresence robot are also included in the report. These improvements are detailed in the Discussion section.

8 Discussion

The success of some of the tests carried out on the robot during the development of this thesis reveals weaknesses that affect the robot. Most of the weaknesses arise from the ability to enable ADB shells and the lack of security around this protocol. Although ADB is not enabled by default, it can be easily enabled on screen, along with changing various settings and revealing confidential information. The nature of these robots involves human interaction, so in most use cases, this interaction with the screen is easily achieved. For all the suggestions to improve the robot's security, it is specified who should apply the changes since the issues can be fixed by the manufacturer or the robot's users (those with administrative rights).

Suggested improvements to increase the security of the robot include protecting the screen with password authentication so that only authorised users can access the robot's settings. This setting should be included in a future software update, fixed by the manufacturer. From a user perspective, a temporary fix would be the installation of an "AppLock" program in charge of securing an application (in this case, the settings app) with password protection. If those changes are applied, the risks associated with disclosing information and modifying settings through the screen will no longer exist.

Another recommendation to secure ADB is to disable it if possible. ADB lacks authentication and encryption, while the SSH server installed in the device provides the same functionality in a secure way. For this purpose, ADB can be removed entirely while maintaining its functionality. This setting is disabled by default, but it can easily be enabled in the robot's settings. Modern Android devices include a message highlighting the risks of this setting and discouraging users from enabling it. The manufacturer could update the software, including a similar approach or directly removing the option in the settings. From a user's perspective, there is no fix other than ensuring this setting is disabled at all times.

Usage monitoring based on ports can be avoided only from the manufacturer's side, by updating its behaviour to use a fixed port and leaving it open regardless of the robot's

usage. This change would only solve this attack vector, but usage could still be monitored by analysing network traffic on a device having access to the Wi-Fi network. From a user perspective, there is no method to fix this security weakness.

The user enumeration on the cloud service can be easily fixed by changing the code not to reveal whether the email is registered or not and instead give a generic message. Again, this action must be performed by the manufacturer since users cannot control the behaviour of the website used to manage and connect to the robots.

Finally, the robot should be connected to a private Wi-Fi network, ideally isolated from the rest of the user's devices, to limit the number of network attacks that can exploit the system. This is a security measure that the robot's administrator should adopt. If these suggested improvements were applied, the success of the designed tests would be as shown in Figure 8.

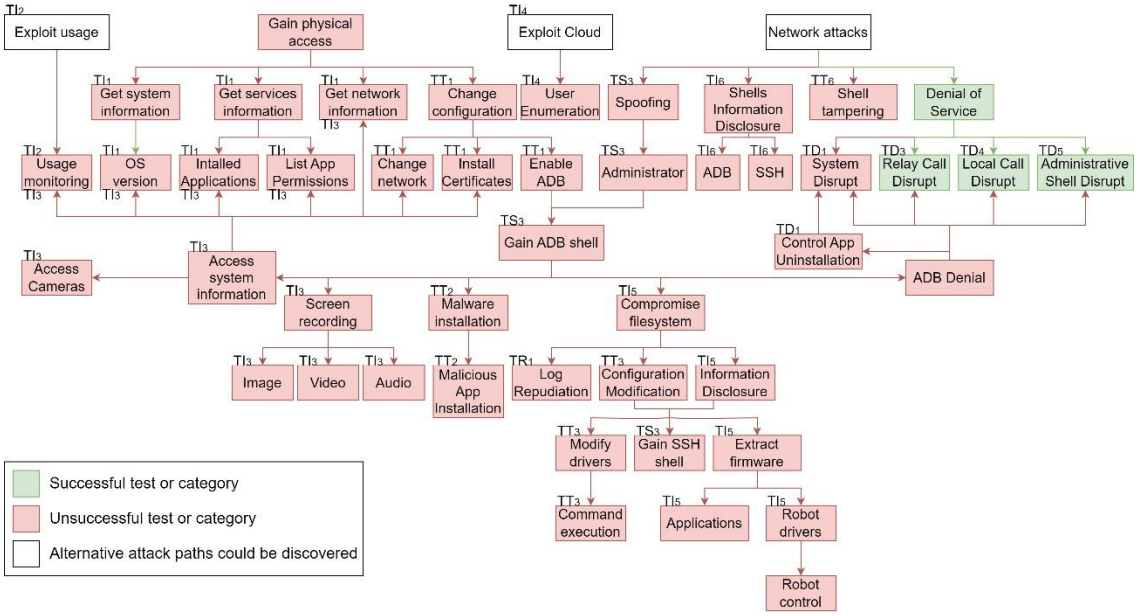

Figure 8. Test tree with improvements applied.

There are several caveats to the approach taken in this research. The methodology followed for penetration testing relies entirely on the data-flow diagram designed. This diagram was designed by combining information about the robot gathered from online documentation and through active methods. Even though the diagram was designed with the highest possible accuracy, any error in its representation would lead to

incorrect addressing of the threats affecting the robot. If the threats were modified, different tests would have to be designed, which would change the results of the work.

Secondly, the attacks carried out were based on various sources, documentation and hacking forums. This approach was taken in order to obtain an up-to-date database of available attacks and methods, although some of the sources do not originate from academic publications. For this reason, several sources were cross-checked in an attempt to achieve the highest possible coverage of attacks. As there are a large number of hacking tools for each method, it is possible that some attack methods have been overlooked. Although special emphasis was placed on the unsuccessful tests in an attempt to discover new attack vectors that would be successful, it is possible that some methods that were not tested would have been successful. Also, the availability of newer tools could change the results of the unsuccessful tests, leading to different results than those in this research.

After evaluating the results, it was discovered that the method that PatrIoT uses for risk scoring, Lightweight Risk Scoring, is excessively lightweight. We realised that the formula used to compute the risk gives the "Simplicity" variable a very high weight over the final score. This results in overvaluing specific threats merely based on how easy they are to be performed. As an example, a threat with the values one for Impact, two for Coverage and three for Simplicity would have a risk score of 2,4. Another threat with values of three for Impact, two for Coverage and two for simplicity would have a risk score of 2,2. This means that a threat with the highest impact but a medium simplicity has a theoretically lower risk than another with the lowest impact but the highest simplicity. We believe this risk scoring method is not adequately balanced. The risk scoring was based on a theoretical perspective, and the values were assigned with the highest possible partiality. However, since the risk scoring method only relies on three values to compute the scores, the values might differ depending on the people who assign the values. This is also tied to the lack of collaboration by the manufacturer, which would have been ideal for a person with full knowledge of the robot, and the company would have decided on the variable values since the risks require organisational context. Another approach would have been to perform vulnerability scoring using a standard like the Common Vulnerability Scoring System (CVSS) [101]. With this, assessing the risk values based on the vulnerabilities that create those risks would be possible. For all the reasons previously mentioned, the risk scoring used in

this thesis has been shown to have some margin to be improved. However, since these risk scores were only used to establish the order of the exploitation of the threats, all threats with a high impact were analysed, and there were several iterations on the threats, meaning that their order didn't affect the results. We believe the choice of the Lightweight Risk Scoring was not ideal, but at the same time, it didn't affect the results. Using an alternative method to score the vulnerabilities or risks found and prioritise the exploits to be tested relies on future work.

In addition, an appropriate validation method for this research, other than empirical testing, couldn't be found. The results were empirically validated during the exploitation phase. This shows that the elicitation of the threats was adequate and did indeed pose a cybersecurity risk to the robot. Due to the novelty of the topic, it was not possible to compare it with other previous research for additional validation. Had the vendor agreed to collaborate on this issue, their security analysis could have been compared to this work to assess its validity. This lack of collaboration also limited the attack surface, as the cloud service was not analysed. Previous literature has shown successful attacks on IoT devices on the cloud surface [102], so this area needs to be analysed for the completeness of the robot penetration test.

It is also worth mentioning that the company that manufactures the robot showed a complete lack of collaboration at all project stages. They never answered when they were first contacted for cooperation on this thesis. More prominent companies usually offer different ways of reporting vulnerabilities, from a single email point of contact to bug bounty programs. Even though it is a generalised issue for small companies, we believe a simple contact method to report vulnerabilities found is an easy yet effective way to collect cyber security issues. The manufacturer should make an effort to implement some mechanism for this purpose.

Finally, this work was limited in time and resources. Although we consider the coverage of the tests to be adequate, some threats were not analysed. From the elicitation of all threats, tests for those with the lowest risk score were not considered in the exploitation phase. As explained above, the threats that weren't analysed had a low impact and were the most complex. Although we believe the results wouldn't be too different when attempting to exploit them, they could be included in future research along with the cloud attack surface to achieve the full penetration test.

9 Summary

The aim of this research was to investigate the potential cybersecurity threats and vulnerabilities affecting the telepresence robot under study. We employed a four-step methodology based on PatrIoT [44], consisting of Reconnaissance, Threat Modelling, Exploitation and Reporting. During the Reconnaissance phase, information about the device and the behaviour of the whole system was collected. Based on this information, a data-flow diagram was created in the Threat Modelling phase, showing the behaviour and information flow between the components of the telepresence robot system. This diagram was used to identify the threats affecting the system using STRIDE threat modelling. The threats were then prioritised according to their risk score. In the Exploitation phase, we used the discovered threats to plan tests to exploit the robot. These tests were then executed, opening up new attack vectors and generating additional tests. After several iterations in which all the planned tests were executed, the results were documented in the Reporting phase. This report contains details of the tests performed, including successful exploits, unsuccessful attempts and a list of improvements that can be made to improve the cybersecurity of the telepresence robot.

The main findings of the research can be summarised in four points corresponding to the proposed research questions. This research was able to identify the potential cybersecurity threats affecting the telepresence robot during the threat modelling phase, thus answering the first research question (RQ1). This thesis also showed how systematic penetration testing can be adapted to the telepresence robot through the methodology applied to conduct the research, answering the second research question (RQ2). A method for selecting the tests to be performed was also shown, using the risk score of the threats to prioritise and select the tests to be performed (RQ3). The improvements that can be made to enhance the cybersecurity of the telepresence robots are shown in section 8, based on the results of the penetration tests. This answers the final research question (RQ4) and successfully achieves the objectives of this thesis.

These findings can contribute to a deeper understanding of the security implications associated with telepresence robots. They can also be used by telepresence robot manufacturers to improve the security of their products. The results also include information about the robot's configuration, which could be used by owners to ensure greater security when using their product.

This research was limited in two ways: by not analysing the system's cloud service due to a lack of cooperation from the service provider and by not analysing low-risk threats due to limited resources. Further research can continue with different objectives, starting from overcoming the limitations described above. The low-risk threats could be included in the penetration testing process together with the cloud network analysis if the service provider allows it. However, future work could also find an additional method to validate the current research, ideally by comparing it to the existing work of the robot's development team.

References

[1] M. Lei, I. M. Clemente, H. Liu and J. Bell, “The acceptance of telepresence robots in higher education,” *International Journal of Social Robotics,* vol. 14, p. 1025–1042, 2022.

[2] A. Velinov, S. Koceski and N. Koceska, “A review of the usage of telepresence robots in education,” *Balkan Journal of Applied Mathematics and Informatics,* vol. 4, p. 27–40, 2021.

[3] N. D. Xuan Hai, L. H. Thanh Nam and N. T. Thinh, “Remote Healthcare for the Elderly, Patients by Tele-Presence Robot,” in *2019 International Conference on System Science and Engineering (ICSSE)*, 2019.

[4] R. Bevilacqua, A. Cesta, G. Cortellessa, A. Macchione, A. Orlandini and L. Tiberio, “Telepresence robot at home: a long-term case study,” in *Ambient Assisted Living: Italian Forum 2013*, 2014.

[5] A. G. Bacudio, X. Yuan, B.-T. B. Chu and M. Jones, “An overview of penetration testing,” *International Journal of Network Security & Its Applications,* vol. 3, p. 19, 2011.

[6] I. Rae and C. Neustaedter, “Robotic telepresence at scale,” in *Proceedings of the 2017 chi conference on human factors in computing systems*, 2017.

[7] I. Priyadarshini, “Cyber security risks in robotics,” in *Cyber security and threats: concepts, methodologies, tools, and applications*, IGI Global, 2018, p. 1235–1250.

[8] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno and H. J. Chizeck, “To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots,” *arXiv preprint arXiv:1504.04339,* 2015.

[9] E. Fosch-Villaronga and T. Mahler, “Cybersecurity, safety and robots: Strengthening the link between cybersecurity and safety in the context of care robots,” *Computer Law & Security Review,* vol. 41, p. 105528, 2021.

[10] N. G. Hockstein, C. G. Gourin, R. A. Faust and D. J. Terris, “A history of robots: from science fiction to surgical robotics,” *Journal of robotic surgery,* vol. 1, p. 113–118, 2007.

[11] IFR, “World Robotics Report 2022,” 2022.

[12] M. Minsky, “Telepresence,” 1980.

[13] M. Desai, K. M. Tsui, H. A. Yanco and C. Uhlik, “Essential features of telepresence robots,” in *2011 IEEE Conference on Technologies for Practical Robot Applications,* 2011.

[14] K. M. Tsui, M. Desai, H. A. Yanco and C. Uhlik, “Exploring use cases for telepresence robots,” in *Proceedings of the 6th international conference on Human-robot interaction*, 2011.

[15] D. Geffen, “World's First Hospital to Introduce Remote Presence Robots in ICU,”

Newswise, 2005.

[16] FACT.MR, “Telepresence Robot Market Is Expected To Accelerate At A Whopping 17% CAGR, Reaching US 1.6 Billion by 2033,” 2023.

[17] J. P. Hansen, A. Alapetite, M. Thomsen, Z. Wang, K. Minakata and G. Zhang, “Head and gaze control of a telepresence robot with an hmd,” in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, 2018.

[18] M. Zhang, P. Duan, Z. Zhang and S. Esche, “Development of telepresence teaching robots with social capabilities,” in *ASME International Mechanical Engineering Congress and Exposition*, 2018.

[19] C. Liu, C. Ishi and H. Ishiguro, “Auditory scene reproduction for tele-operated robot systems,” *Advanced Robotics*, vol. 33, p. 415–423, 2019.

[20] G. Zhang and J. P. Hansen, “Telepresence robots for people with special needs: A systematic review,” *International Journal of Human–Computer Interaction*, vol. 38, p. 1651–1667, 2022.

[21] D. G. Macharet and D. A. Florencio, “A collaborative control system for telepresence robots,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012.

[22] G. Zhang and J. P. Hansen, “Accessible control of telepresence robots based on eye tracking,” in *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, 2019.

[23] S. J. Badashah and others, “Telepresence Robot Using Raspberry Pi,” *Journal of Algebraic Statistics*, vol. 13, p. 2165–2172, 2022.

[24] A. Kosugi, M. Kobayashi and K. Fukuda, “Hands-Free collaboration using telepresence robots for all ages,” in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion*, 2016.

[25] B. Hine, C. Stoker, M. Sims, D. Rasmussen, P. Hontalas, T. Fong, J. Steele, D. Barch, D. Andersen, E. Miles and others, “The application of telepresence and virtual reality to subsea exploration,” in *Proceeding of the 2nd Workshop on: Mobile Robots for Subsea Environments, International Advanced Robotics Program (IARP), MJ Lee and RB McGee (eds), Monterey, CA*, 1994.

[26] P. Ardanuy, C. Otto, J. Head, N. Powell, B. Grant and T. Howard, “Telepresence enabling human and robotic space exploration and discovery: antarctic lessons learned,” *Space 2005*, p. 6756, 2005.

[27] C. R. Stoker, “Telepresence in the human exploration of Mars: Field studies in analog environments,” *Proc. Vision 21: Interdisciplinary Science and Engineering in the Era of Cyberspace*, p. 23–34, 1993.

[28] B. P. Singh, “Telepresence Robots Market Research Report 2022, Size, Share, Trends and Forecast to 2027,” LinkedIn, 2022. [Online]. Available: https://www.linkedin.com/pulse/telepresence-robots-market-research-report-2022-size-share-singh.

[29] H. M. Z. A. Shebli and B. D. Beheshti, “A study on penetration testing process and tools,” in *2018 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, 2018.

[30] M. Bishop, “About Penetration Testing,” *IEEE Security & Privacy*, vol. 5, pp. 84-87, 2007.

[31] G. Weidman, Penetration testing: a hands-on introduction to hacking, No starch press, 2014.

[32] UK National Cyber Security Centre, “Threat Modelling,” *Risk management,* 2023.

[33] P. S. Shinde and S. B. Ardhapurkar, “Cyber security analysis using vulnerability assessment and penetration testing,” in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave),* 2016.

[34] R. E. Johnson, “Frameworks=(components+ patterns),” *Communications of the ACM,* vol. 40, p. 39–42, 1997.

[35] D. Avison and G. Fitzgerald, Information systems development: methodologies, techniques and tools, McGraw-Hill, 2003.

[36] T. Wilhelm, *Professional Penetration Testing: Volume 1: Creating and Learning in a Hacking Lab (Vol. 1, pp. 26). Burlington: Syngress,* 2009.

[37] S. Shah and B. M. Mehtre, “An overview of vulnerability assessment and penetration testing techniques,” *Journal of Computer Virology and Hacking Techniques,* vol. 11, p. 27–49, November 2014.

[38] M. Alhamed and M. M. H. Rahman, “A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions,” *Applied Sciences,* vol. 13, 2023.

[39] M. R. Albrecht and R. B. Jensen, “The Vacuity of the Open Source Security Testing Methodology Manual,” in *Security Standardisation Research: 6th International Conference, SSR 2020, London, UK, November 30–December 1, 2020, Proceedings 6,* 2020.

[40] E. A. Morse and V. Raval, “PCI DSS: Payment card industry data security standards in context,” *Computer Law & Security Review,* vol. 24, pp. 540-554, 2008.

[41] OWASP, “OWASP Web Security Testing Guide”.

[42] OWASP, “OWASP Mobile Application Security Testing Guide”.

[43] K. Scarfone, M. Souppaya, A. Cody and A. Orebaugh, “Technical guide to information security testing and assessment,” *NIST Special Publication,* vol. 800, p. 2–25, 2008.

[44] E. Süren, F. Heiding, J. Olegård and R. Lagerström, “PatrIoT: practical and agile threat research for IoT,” *International Journal of Information Security,* vol. 22, p. 213–233, November 2022.

[45] M. Berner, *Where’s My Car? Ethical Hacking of a Smart Garage,* 2020.

[46] G. Rubbestad and W. Söderqvist, *Hacking a Wi-Fi based drone,* 2021.

[47] C. Torgilsman and E. Bröndum, *Ethical hacking of a Robot vacuum cleaner,* 2020.

[48] M. Rak, G. Salzillo, C. Romeo and others, “Systematic IoT Penetration Testing: Alexa Case Study.,” in *ITASEC,* 2020.

[49] A. Konev, A. Shelupanov, M. Kataev, V. Ageeva and A. Nabieva, “A survey on threat-modeling techniques: protected objects and classification of threats,” *Symmetry,* vol. 14, p. 549, 2022.

[50] K. Tuma, G. Calikli and R. Scandariato, “Threat analysis of software systems: A systematic literature review,” *Journal of Systems and Software,* vol. 144, p. 275–294, 2018.

[51] S. Hussain, A. Kamal, S. Ahmad, G. Rasool and S. Iqbal, “Threat modelling methodologies: a survey,” *Sci. Int.(Lahore),* vol. 26, p. 1607–1609, 2014.

[52] K. Wuyts and W. Joosen, “LINDDUN privacy threat modeling: a tutorial,” *CW Reports,* 2015.

[53] K. L. DistriNet Research Unit, “LINDDUN.ORG”.

[54] T. UcedaVélez, “What is PASTA Threat Modeling?,” 2021.

[55] F. Den Braber, I. Hogganvik, M. S. Lund, K. Stølen and F. Vraalsen, “Model-based security analysis in seven steps—a guided tour to the CORAS method,” *BT Technology Journal,* vol. 25, p. 101–117, 2007.

[56] P. Saitta, B. Larcom and M. Eddington, “Trike v. 1 methodology document [draft],” *URL: http://dymaxion. org/trike/Trike v1 Methodology Documentdraft. pdf,* 2005.

[57] I. Yaqoob, S. A. Hussain, S. Mamoon, N. Naseer, J. Akram and A. ur Rehman, “Penetration testing and vulnerability assessment,” *Journal of Network Communications and Emerging Technologies (JNCET) www. jncet. org,* vol. 7, 2017.

[58] S. Mehta, G. Raj and D. Singh, “Penetration Testing as a Test Phase in Web Service Testing a Black Box Pen Testing Approach,” in *Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 2,* 2018.

[59] S. Huckins, “R7-2017-01: Multiple Vulnerabilities in Double Robotics Telepresence Robot,” Rapid7, 13 March 2017. [Online]. Available: https://www.rapid7.com/blog/post/2017/03/13/r7-2017-01-multiple-vulnerabilities-in-double-robotics-telepresence-robot/. [Accessed 16 February 2024].

[60] D. Regalado, “Watching You through the Eyes of,” Zingbox, 2018.

[61] M. Bereza, “Call an Exorcist! My Robot’s Possessed!,” McAfee, 05 August 2020. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/call-an-exorcist-my-robots-possessed/. [Accessed 17 Februry 2024].

[62] OWASP, “OWASP Top 10 Internet of Things,” 10.

[63] MITRE, *Common Weakness Enumeration (CWE).*

[64] OWASP, *OWASP Firmware Security Testing Methodology (FSTM),* 2020.

[65] O. W. A. S. P. Project, *OWASP Web Security Testing Guide.*

[66] O. W. A. S. P. Project, *OWASP Mobile Security Testing Guide.*

[67] P. Mell, K. Scarfone and S. Romanosky, “Common vulnerability scoring system,” *IEEE Security & Privacy,* vol. 4, p. 85–89, 2006.

[68] OWASP, “OWASP Risk Rating Methodology”.

[69] L. Kohnfelder and P. Garg, “The threats to our products,” *Microsoft Interface, Microsoft Corporation,* vol. 33, 1999.

[70] Z. Abuabed, A. Alsadeh and A. Taweel, “STRIDE threat model-based framework for assessing the vulnerabilities of modern vehicles,” *Computers & Security,* vol. 133, p. 103391, 2023.

[71] D. Overstreet, H. Wimmer and R. J. Haddad, “Penetration testing of the amazon echo digital voice assistant using a denial-of-service attack,” in *2019 SoutheastCon,* 2019.

[72] R. Khan, K. McLaughlin, D. Laverty and S. Sezer, “STRIDE-based threat modeling for cyber-physical systems,” in *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe),* 2017.

[73] K. Tuma and R. Scandariato, “Two architectural threat analysis techniques compared,” in *Software Architecture: 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24–28, 2018, Proceedings 12*, 2018.

[74] M. Howard and S. Lipner, The security development lifecycle, vol. 8, Microsoft Press Redmond, 2006.

[75] Angry IP Scanner, “Angry IP Scanner,” [Online]. Available: https://angryip.org/. [Accessed 19 March 2024].

[76] G. Lyon, “Nmap,” [Online]. Available: https://nmap.org/. [Accessed 19 March 2024].

[77] Fing Limited, “Fing,” [Online]. Available: https://www.fing.com/. [Accessed 19 March 2024].

[78] MACVendors, “MAC Vendors,” [Online]. Available: https://macvendors.com/. [Accessed 19 March 2024].

[79] T. Perry, “HTTP Toolkit,” [Online]. Available: https://httptoolkit.com/. [Accessed 19 March 2024].

[80] S. Tatham, “PuTTYgen,” [Online]. Available: https://www.puttygen.com/. [Accessed 19 March 2024].

[81] Google, “Android SDK Platform Tools,” [Online]. Available: https://developer.android.com/studio/releases/platform-tools. [Accessed 19 March 2024].

[82] C. Smith, “Firmwalker,” [Online]. Available: https://github.com/craigz28/firmwalker. [Accessed 19 March 2024].

[83] R. Wiśniewski and C. Tumbleson, “Apktool,” [Online]. Available: https://ibotpeaches.github.io/Apktool/. [Accessed 19 March 2024].

[84] Google, “Android Studio,” [Online]. Available: https://developer.android.com/studio. [Accessed 19 March 2024].

[85] Rapid7, “Metasploit,” [Online]. Available: https://www.metasploit.com/. [Accessed 19 March 2024].

[86] Rapid7, “Msfvenom,” [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/msfvenom/. [Accessed 19 March 2024].

[87] TeamViewer AG, “TeamViewer,” [Online]. Available: https://www.teamviewer.com/. [Accessed 19 March 2024].

[88] L. Akash, “Linux Wifi Hotspot,” [Online]. Available: https://github.com/lakinduakash/linux-wifi-hotspot. [Accessed 19 March 2024].

[89] Wireshark Foundation, “Wireshark,” [Online]. Available: https://www.wireshark.org/. [Accessed 19 March 2024].

[90] S. Margaritelli, “Bettercap,” [Online]. Available: https://www.bettercap.org/. [Accessed 19 March 2024].

[91] C. Devine, “Aircrack-ng,” [Online]. Available: https://www.aircrack-ng.org/. [Accessed 19 March 2024].

[92] Ettercap Project, “Ettercap,” [Online]. Available: https://www.ettercap-project.org/. [Accessed 19 March 2024].

[93] PortSwigger, “Burp Suite,” [Online]. Available: https://portswigger.net/burp. [Accessed 19 March 2024].

[94] Josh, “Burp-Non-HTTP-Extension,” [Online]. Available: https://github.com/summitt/Burp-Non-HTTP-Extension. [Accessed 19 March 2024].

[95] P. S. Foundation, “Python,” [Online]. Available: https://www.python.org/. [Accessed 19 March 2024].

[96] P. Biondi, “Scapy,” [Online]. Available: https://scapy.net/. [Accessed 19 March 2024].

[97] M. Fox, “NetfilterQueue,” [Online]. Available: https://pypi.org/project/NetfilterQueue/. [Accessed 19 March 2024].

[98] DistriNet, “LINDDUN PRO Tutorial,” 2023.

[99] B. Cannoles and A. Ghafarian, “Hacking experiment by using usb rubber ducky scripting,” *Journal of Systemics*, vol. 15, p. 6671, 2017.

[100] Mitmproxy, “Install System CA Certificate on Android Emulator,” [Online]. Available: https://docs.mitmproxy.org/stable/howto-install-system-trusted-ca-android/. [Accessed 3 March 2024].

[101] K. a. M. P. Scarfone, “An analysis of CVSS version 2 vulnerability scoring,” *2009 3rd International Symposium on Empirical Software Engineering and Measurement*, pp. 516--525, 2009.

[102] T. Stroeven and F. Söderman, *Cybersecurity Evaluation of an IP Camera*, 2022.

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

I Diego del Rio Manzanas

1. Grant Tallinn University of Technology free licence (non-exclusive licence) for my thesis "Securing Remote Connectivity: A Systematic Penetration Testing Analysis of a Telepresence Robot" supervised by "Shaymaa Mamdouh Khalil".
   1.1. to be reproduced for the purposes of preservation and electronic publication of the graduation thesis, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright;
   1.2. to be published via the web of Tallinn University of Technology, incl. to be entered in the digital collection of the library of Tallinn University of Technology until expiry of the term of copyright.
2. I am aware that the author also retains the rights specified in clause 1 of the non-exclusive licence.
3. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights, the rights arising from the Personal Data Protection Act or rights arising from other legislation.

10.05.2024

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.