Politecnico di Torino

Computer engineering

Master's thesis

Threat modeling on data sharing

Candidate:

Gianluca Turco

Advisor: **Prof. Fulvio Valenza**

Co-advisor: **Dott. Daniele Bringhenti**

Co-advisor: **Ing. Lucia Seno**

Academic Year 2022-2023

Contents

List of Figures

Chapter 1

Introduction

In recent years, there has been a marked increase in the number of people able to access the Internet and related services, becoming an integral part of a person's life. Millions of people around the world have access to the Internet, allowing them to interact with each other in ways never seen before. This phenomenon has created a knock-on effect, as the increase in users has led to the expansion of available online services, meeting the needs and expectations of modern consumers.

Undoubtedly one of the great advantages of the Internet is the possibility of sharing one's experiences quickly, immediately and without geographical boundaries. People can share projects, ideas and expertise, collaborating with colleagues all over the world through tools such as video conferencing, document sharing and project management platforms; but it is not only the business sector that benefits from this trend. Even in our free time, the Internet offers a wide range of platforms and social media that allow us to share our leisure experiences with the world. In a few simple steps you can register on a platform, customise your account and start communicating with others. While this expansion in the use of the Internet provides many benefits, it also brings with it new problems related to a person's privacy, which seems to be increasingly put at risk, since a great amount of personal data are public and can be accessed by anyone. Nowadays due to the trend of sharing everything to everyone, especially due to the success of social networks, it is very easy for an attacker to find a user's personal information and exploit it for their own convenience. Although the security systems of a service may be of a high standard an individual is never safe from cyberattacks, as any kind of information, even that which the user considers less important, could be exploited for malicious purposes. Moreover, with the development of new attacks, the risk of falling victim to them is increasing, making it essential to take security measures that can nullify or mitigate threats. In order to protect users, it is useful to develop new protection measures that can ensure user safety.

In this thesis, the approach used is the threat model, which is a means of analyzing the various components of the system, finding out if there are any weaknesses that can be exploited by malicious attackers to cause harm to users, and looking for countermeasures to defend against their attacks. The objective of this project is to develop a dedicated threat model for data sharing and to evaluate its effectiveness through appropriate validations.

The thesis is organised as follows:

- **Chapter 2**: it serves as a background, providing the theoretical concepts useful to enhance the understanding of the thesis work. In particular, it delves into the topics of the threat model and the data sharing system.
- **Chapter 3**: it defines the objective of this thesis and clarifies the motivation for undertaking this work.
- **Chapter 4**: it goes into the most important part of the whole project, which involves the complete development of the threat model for data sharing.
- **Chapter 5**: the tool used for model validation is introduced and its application is illustrated through various use cases.
- **Chapter 6**: it presents conclusions and considerations on the work done

Chapter 2

Background

In this chapter, the foundation is laid for the comprehensive exploration of various areas covered in this thesis. Here, it is provided a comprehensive overview of the fundamental and preliminary concepts that form the bedrock of the entire project. By delving into these concepts, the aim is to establish a solid understanding of the diverse domains and disciplines that intersect within the scope of this research.

2.1 Threat model

Threat modeling is an approach that enhances network security by identifying vulnerabilities, pinpointing targets, and creating countermeasures to prevent or mitigate the impact of cyber-attacks on the system [17]. As specified in [4], it can be applied to a wide range of things, including software, applications, systems, networks, distributed systems, Internet of Things devices, business processes and it typically includes:

- Description of the subject to be modeled.
- Assumptions susceptible to validation or scrutiny in the future, considering the evolving threat landscape.
- Potential threats to the system.
- Actions that can be taken to mitigate each threat.
- A method to validate the model and assess threats, as well as verify the effectiveness of the actions taken.

Threat modeling is a systematic approach that involves gathering, structuring, and evaluating all this information. When applied to software, it facilitates well-informed decision-making regarding application security risks. Apart from creating a model, typical threat modeling endeavors also yield a prioritized list of security enhancements for the application's concept, requirements, design, or implementation.

2.1.1 Historical overview

Throughout history, military strategists have employed threat trees and threat modeling as essential tools. One of the earliest known treatises on this subject is "The Art of War," written by the renowned Chinese general Sun Tzu around 512 BC. Over the centuries, the evolution of threat modeling has consistently emphasized enhancing military defensive preparedness.
Following the introduction of shared computing in the early 1960s, individuals started searching for ways to exploit security vulnerabilities for their own advantage. This led engineers and computer scientists to swiftly develop threat modeling concepts for information technology systems.
In 1994, Edward Amoroso introduced the concept of a "threat tree" in his book "Fundamentals of Computer Security Technology." [1] Threat trees, akin to decision tree diagrams, visually illustrate how potential threats to a computer system could be exploited. Concurrently, the NSA and DARPA independently employed a structured graphical representation known as the "attack tree" to demonstrate how specific attacks might be executed.
In 1998, Bruce Schneier made a significant contribution to the evolution of threat modeling for IT-systems through his paper titled "Toward a Secure System Engineering Methodology" [16], where he analyzed cyber risks using attack trees. In his analysis, the attacker's objective is depicted as the "root node," while the potential methods to achieve the goal are represented as "leaf nodes." This application of the attack tree enabled cybersecurity professionals to methodically assess multiple attack vectors against a defined target. Schneier's paper proved to be a seminal work in the field of threat modeling. In 1999, building on Schneier's work, Loren Kohnfelder and Praerit Garg developed another threat modeling methodology called STRIDE. This approach aimed to assist Microsoft security professionals in systematically analyzing potential attacks, that could be executed within specific segments of a computer system. In the year that STRIDE was introduced, Carnegie Mellon University's Software Engineering Institute introduced the OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) method. OCTAVE was the first risk-based

assessment methodology for IT security that aimed to strike a balance between considering both the technological and organizational aspects of potential threats while also addressing the practicalities of security practices. Its primary focus lies in effectively managing organizational risk.
In 2004, Frank Swiderski and Window Snyder pioneered the concept of employing threat modeling as a proactive approach to building secure applications. During the same year, Microsoft introduced the Threat Analysis and Modeling (TAM) tool. The tool was designed to enable non-security experts to input relevant information, such as business requirements and application specifications, in order to examine potential threats to the application being developed. Using data flow diagrams (DFD) and Schneier-style attack trees, the tool aided development teams in anticipating potential attacks and devising proactive security measures to implement.
In 2011, as cited in [20], ThreatModeler pushed the evolution of threat modeling toward automated threat analysis and introduced the concept of the process flow diagram (PFD). PFD provide a graphical representation of how users interact with different features of an application. Acting as an application "map," PFDs reflect the thought process of developers and potential attackers alike, as they analyze the application's functionality and potential vulnerabilities. In the subsequent year, ThreatModeler introduced the idea of distinguishing application threat modeling and operational threat modeling as distinct functions. Application threat modeling aids development teams in proactively integrating security requirements into their initial coding within an Agile development environment. On the other hand, operational threat modeling delves into the end-to-end data flow across an organization's infrastructure, providing a comprehensive understanding of the organization's risk profile at a broader level. In 2013 ThreatModeler produced the first commercial, automated, enterprise threat modeling tool.

2.1.2 Threat model terminology

In threat modeling, there are some basic terms described in [14] that it is best to know in order to better understand the topic:

1. **Threat Agent**: it refers to an individual or group with the potential to execute a specific threat. It is essential to ascertain the entities interested in capitalizing on a company's assets, comprehend the potential approaches they could employ against the company, and evaluate their competence to execute

such actions. Certain threats demand heightened expertise or resources, consequently increasing the caliber of threat actor required.

2. **Impact**: Impact quantifies the potential damage due to a specific threat, which may cause damage to tangible assets or manifest as obvious financial loss. Secondary losses could also be triggered by these attacks, which must be considered in the overall impact assessment.

3. **Likelihood**: It assesses the probability that a given threat will be carried out, influenced by a number of factors that include both the complexity of the execution of the threat and the potential advantage and benefit to be gained by the attacker.

4. **Controls**: Controls are protective measures or defenses implemented to prevent, identify, mitigate, or reduce potential threats to information, systems, or other valuable resources.

5. **Mitigations**: Mitigations include particular controls implemented to reduce the likelihood or consequences of a threat, while not eliminating it entirely.

6. **Data Flow Diagram**: The data flow diagram is a description of the stream of information within the system, showing where data flows in and out of each process.

2.1.3 Threat modeling methodologies

As mentioned earlier, there are numerous threat model methodologies [15], which are selected based on what types of threats one wants to model and for what purposes. As specified in [22], typically, threat modeling is implemented using one of these approaches:

1. **Asset-Centric**: In the asset-centric approach, the focus is on identifying and protecting valuable assets within the system. The goal is to understand which assets are the most critical and which threats could put them at risk. It involves analyzing how different threats might target and exploit assets, and then implementing security measures to safeguard those assets. One of the most well-known asset-centric methodology is OCTAVE, which focuses on identifying and protecting critical assets within an organization.

2. **Attacker-Centric**: In the attacker-centered approach, the main focus is on understanding potential attackers, their motivations and capabilities. It involves modeling the views and strategies of potential attackers, outlining the steps they might take to infiltrate the system, and identifying the vulnerabilities they might exploit. An example of attack-centric approach is STRIDE, which works by examining the system from the attacker's point of view and categorizing threats according to six types of possible attacks:

   - Spoofing.
   - Tampering.
   - Repudiation.
   - Information disclosure.
   - Denial of service.
   - Elevation of privilege.

   Another very famous attack-centric approach is PASTA (Process for Attack Simulation and Threat Analysis), which is designed to correlate business objectives with technical requirements, guiding teams to dynamically identify, count and prioritize threats through several steps:

   - Define objectives.
   - Define the technical scope of assets and components.
   - Application decomposition and identify application controls.
   - Threat analysis based on threat intelligence.
   - Vulnerability detection.
   - Attack enumeration and modeling.
   - Risk analysis and development of countermeasures.

3. **Software-Centric**: The software-centric approach concentrates on the application or system's architecture and design. It involves analyzing the software components, data flows, and interactions to identify possible vulnerabilities and security weaknesses within the codebase. VAST (Visual, Agile, and Simple Threat Modeling) is a software-centric approach that emphasizes simplicity, visual representations, and agility in the threat modeling process; the key characteristics of VAST include:

- Visual Representation: VAST utilizes visual models, diagrams, and flowcharts to represent the application's architecture and potential threats. This approach makes it easier for stakeholders to understand and communicate security risks effectively.
- Agile methodology: VAST is designed to be agile and lightweight, enabling organizations to integrate threat modeling seamlessly into their development process. The goal is to provide rapid feedback and iterative improvements to software application security.
- Simplicity: VAST focuses on simplicity, avoiding unnecessary complexity in the threat modeling process. It aims to quickly identify critical security issues, without getting bogged down in extensive documentation.

4. **Value and Stakeholder-Centric**: The value and stakeholder-centered approach considers the business value of the system and stakeholder perspectives. The goal is to align threat modeling decisions with business objectives, regulatory requirements, and stakeholder interests. This approach helps prioritize security measures based on overall system value and stakeholder concerns. There are no purely value and stake holder-centric methodologies, but many of them, such as STRIDE and OCTAVE, can also integrate these elements to help guide security decisions and identify threats or to identify critical assets for the organization and better understand stakeholder expectations regarding security and data protection.
5. **Hybrid Approach**: A hybrid approach combines elements of one or more of the above methods, tailoring the threat modeling process to the specific needs of the organization or project. It may include a mix of asset-centric, attacker-centric, or other perspectives to achieve a comprehensive analysis of security risks.

2.1.4 How does threat modeling work?

Threat modeling involves identifying the various threat agents that can potentially harm an application or computer system. It adopts the viewpoint of malicious hackers to gain insight into the potential extent of damage they could inflict. This process enables a more profound comprehension and exploration of crucial aspects of the system. Usually, organizations perform threat modeling during the design phase (although it can also occur at other stages) of a new application to assist developers in identifying vulnerabilities and understanding the security implications of their design, code, and configuration decisions.

According to [3], the threat modeling process can be decomposed into four high level steps:

- Decompose the Application.
- Determine and Rank Threats.
- Determine Countermeasures and Mitigation.
- Validation.

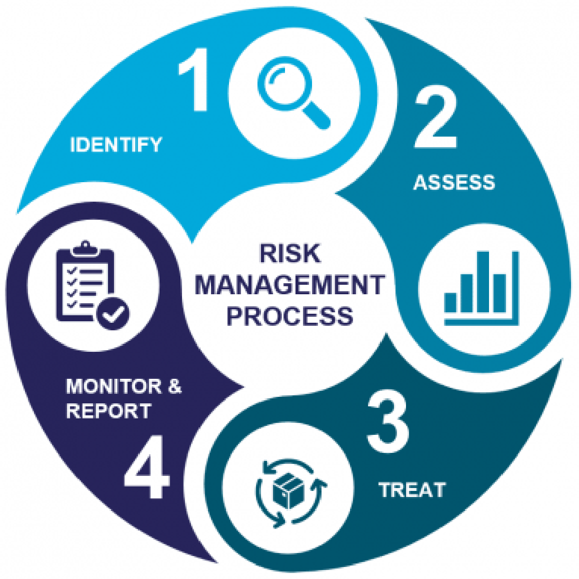


Figure 2.1: High level steps for threat modelling

Decompose the application

The goal of this phase is to understand the application and how it interacts with external entities. External dependencies refer to elements outside the application's code that could present a risk to the application. These elements are generally under the organization's influence, yet they might not be directly managed by the development team. This goal is achieved through information gathering, documentations and it involves:

- Creating use cases to understand how the application is used.
- Identifying entry points to establish potential locations where attackers might interact with the application.
- Identification of assets, i.e., elements or areas, that are potential targets for an attacker.
- Determining trust levels that denote the access privileges the application will extend to external entities.

Entry points establish the interfaces via which potential attackers can engage with the application or provide it with data and should be documented as follows:

1. ID: A unique ID assigned to the entry point.
2. Name: A descriptive name that identifies the entry point and its purpose.
3. Desciption: A textual description about the interactions and processes taking place at the entry point
4. Trust Levels: The level of access required by the entry point.

As written above, there are entry points, which show where data enter the system, but also exit points from which data leave the system. If there are exit points from components that handle sensitive data, the absence of security controls to safeguard confidentiality and integrity, may result in unauthorized exposure of this confidential information to unauthorized users. The target of attackers usually are assets, since if compromised they can lead to direct benefits for the attackers themselves or cause significant damage to the target organization. Assets are documented in the threat model as follows:

1. ID: A unique ID is assigned to identify each asset.
2. Name: A descriptive name that clearly identifies the asset.
3. Description: A textual description of what the asset is and why it should be protected.
4. Trust Levels: The level of access required to enter the entry point is documented here.

After identifying entry and exit points and all assets, it is necessary to define their relative Trust Level, to decide the access rights or privileges required for each component. A Trust Level is defined as followed:

1. ID: A unique number is assigned to each trust level.
2. Name: A distinctive label that enables the identification of external entities to which this level of trust has been granted.
3. Description: A textual explanation of the trust level, providing insights into the external entity to whom the trust level has been assigned.

After gathering all this information, an accurate model of the application can be created through the use of a data flow diagram. Through a graphical representation it shows the logical flow of data, identifying compromised components through critical points. Because of their hierarchical structure, DFDs are leveraged to decompose the application into subsystems and lower levels of subsystems. The higher-level DFD will help clarify the purpose of the application under investigation, while the lower levels will allow the focus to be on the specific processes that affect the data.

Determine and rank threats

The initial stage in identifying threats involves the implementation of threat categorization. Threat categorization encompasses the process of classifying potential

threats into distinct groupings according to shared attributes like attack type, motivation, impact, and additional factors. This classification procedure helps organize and group threats, improving understanding and facilitating effective management.
In the realm of threat analysis, it's essential to grasp the fundamental concept of risk. Risk is the potential for encountering losses, influenced by two key elements: the likelihood of an attack occurring and the possible impact or cost associated with that attack. Risk is calculated as:

(Probability that threat occurs) x (Cost to organization)

Hence, threat modeling is nothing more than a systematic approach that aims to identify and enumerate threats to an application environment, with the goal of minimizing risk. Typically, the threat identification procedure involves a series of steps. In the initial phase, all potential threats from the list of relevant threats to each component are examined. In subsequent iterations, threats are subjected to a more in-depth analysis, including examining attack paths, identifying triggers that make the threat exploitable, and identifying measures needed for mitigation. After assessing the prevailing threats, vulnerabilities, and attacks, a more focused threat analysis should include exploration of use and abuse cases. Through a comprehensive examination of use scenarios, potential weaknesses that could pave the way for a concrete threat can be identified. In addition, it is critical to identify abuse cases as well. These cases shed light on how existing protective measures might be bypassed or reveal areas without adequate protections. The result of the threat analysis is the determination of the types of threats posed to each component of the decomposed system.
Threats can be classified according to their level of risk. Evaluating the risk posed by the different threats identified allows a list of priority threats to be developed, aiding the formulation of a risk mitigation plan. Different criteria can be used to classify threats as high, medium or low risk. 2 different types of risk assessment models can be used:

- Subjective
- Qualitative

An example of the first one is DREAD, which facilitates the allocation of values to various factors that influence a threat. To determine where the threat stands in terms of priority, the threat analyst answers questions related to each risk factor, assigning each a score between 1 and 10:

1. **D**amage: What would be the extent of the damage in the event of a successful attack?
2. **R**eproducibility: How easy is it to reproduce an attack?
3. **E**xploitability: How much time, effort, and expertise is needed to exploit the threat?
4. **A**ffected Users: If a threat were exploited, what percentage of users would be affected?
5. **D**iscoverability: How easy is it for an attacker to discover this threat?

The DREAD model does not have widespread use across the industry, as its ratings are subjective.
The second category is a qualitative model, in which risk is measured by the probability of an attack occurring and the subsequent consequences of that attack. Probability can be assessed by the level of exploitability, which is influenced by the nature of the threat, system attributes, and any established preventive measures. To determine the ease of exploitation, it may be useful to answer the following questions:

- Can an attacker exploit this remotely?
- Does the attacker need to be authenticated?
- Can the exploit be automated?

The extent of the impact of attacks depends mainly on the potential for damage and its extent, which includes factors such as the amount of components that could be affected by a threat. As before there are some questions that help to understand the potential for damage due to an attack:

- Is it possible for an attacker to gain full control and manipulate the system?
- Can an attacker gain administration access to the system?

- Can an attacker crash the system?
- Can the attacker obtain access to sensitive information?

These examples help calculate overall risk values by assigning qualitative labels such as High, Medium and Low to the probability and impact factors. This approach assists in mitigating the risk of the ranking process becoming excessively subjective.

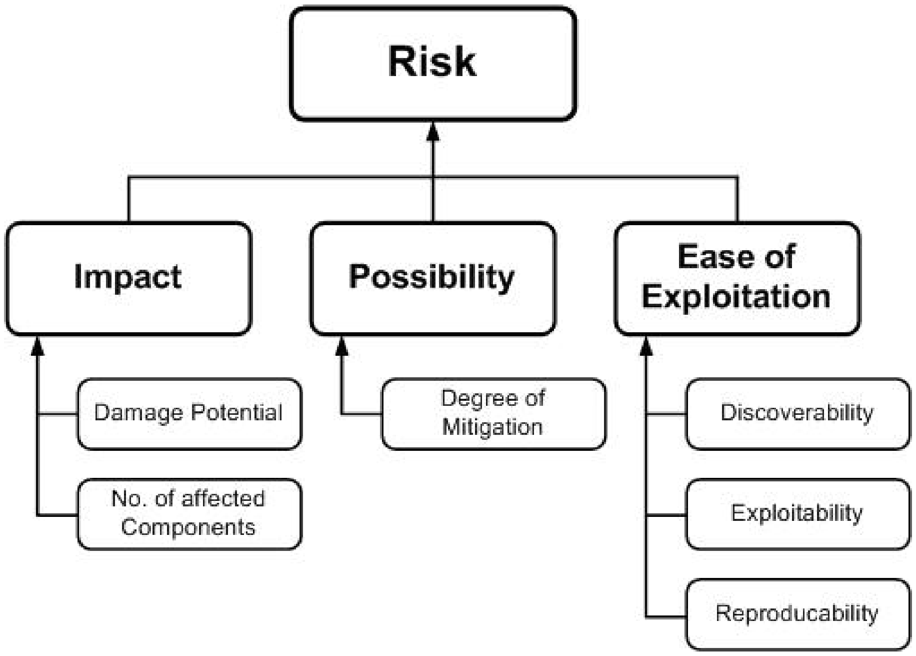


Figure 2.2: Risk factor

Countermeasures and mitigation

In this step, an attempt is made to identify whether there are any protective measures that are capable of preventing a threat from being implemented; at the end of the search it will be possible to classify 3 different types of threats:

1. **Non mitigated threats**: Vulnerabilities, i.e., those threats that cannot be countered and therefore will cause damage of a certain magnitude.

2. **Partially mitigated threats**: Threats that are partially mitigated by one or more countermeasures, but can still be exploited by an attacker to cause limited impact.

3. **Fully mitigated threats**: Threats that are completely nullified by countermeasures and therefore no longer pose a danger

Validation

The final and crucial step in the threat modeling process is the validation phase, with the primary aim being a comprehensive analysis of the work accomplished thus far. This phase involves a meticulous examination of various critical aspects. The first and most important is the examination of the accuracy of the model. It is essential that the model accurately represents the architecture, data flows and components of the system to ensure a true representation of reality. In addition, it is important to verify the correct classification of threats according to their severity, probability of occurrence and impact. This verification process ensures that threats accurately reflect their real relevance to system security. In addition, an assessment is performed to determine whether the countermeasures and mitigation strategies implemented are truly effective in addressing the identified threats.
All these validation checks are recorded in a validation activity log, which can serve as a valuable reference for future reviews and audits.

2.1.5 Advantages

Threat modeling offers an innovative and functional way to strengthen the security of a system or application, and as described in [18] and [19] has many benefits:

- Detect problems early in the software development life cycle.
- Spot design flaws that traditional testing methods and code reviews may overlook.
- Helps in making rational decisions about prioritizing threats
- Evaluate new forms of attack that you might not otherwise consider.
- Maximize testing budgets by helping target testing and code review.

- Identify security requirements.
- Enables remediation of problems before software release
- Keeps frameworks one step ahead of internal and external attackers
- Think about threats beyond standard attacks and security issues unique to your application.
- Highlight assets, threat agents, and controls to deduce components that attackers will target.
- Model the location of threat agents, motivations, skills, and capabilities to locate potential attackers in relation to the system architecture.

2.2 Data sharing

Sharing data has always been used, even before the advent of computers, but with the technological development of the last decade, sharing has been made much easier and more straightforward; the act of file sharing continues to revolutionise the way people collaborate, work and interact socially. With the ever-increasing number of tools and platforms available, fully understanding the mechanisms of file sharing has become essential to reap the full benefits. This allows users to access data efficiently, stay aligned with the latest content and ultimately promote greater productivity across a wide range of activities.
Besides the great advantages, data sharing also entails risks: in many cases, users may share personal information unintentionally or unknowingly. This data may be exploited by malicious persons for unauthorised purposes, including attacks or fraud. Therefore, it is crucial that users pay attention to what they share online and maintain conscious control over the permissions granted to apps. After providing a brief overview of data sharing regulations, the following sections will delve into an examination of the different data sharing mechanisms used by Google's systems. We will use specific services as illustrative examples, as these services were used as use cases in the development of the model presented in this thesis.

2.2.1 Data Sharing Regulations

Nowadays it is possible to share personal data very simply, but one must always bear in mind that laws or other regulations apply each time, depending on the platform where the data is published. A European law called the General Data Protection Regulation (GDPR) has been defined to regulate the privacy of citizens' personal data. This law applies to all organisations that handle personal data of individuals in the EU, so it also applies to companies outside the EU that handle data of European citizens. According to [2], under the GDPR, there are 3 main components:

1. **Data controller**:it is the individual or organisation that decides why and how personal data is collected and processed. It is responsible for ensuring that data processing complies with data protection laws and that people's rights to their data are respected.

2. **Data processor**: It carries out data processing activities in accordance with the instructions provided by the Data Controller and to comply with data

protection regulations.

3. **Data subject**: The data subject is the person to whom the data refer and whose rights and interests are protected by data protection and privacy laws. They possess the entitlement to receive information regarding the handling of their data, the privilege to retrieve their own data, and the prerogative to ensure their data is processed in compliance with legal prerequisites, encompassing consent and data security.

As described in [12], the GDPR is based on a set of fundamental principles aimed at ensuring the protection of personal data and privacy of individuals:

1. **Principle of lawfulness, fairness and transparency**: Personal data must be processed lawfully, fairly and transparently towards the data subject. A clear explanation of how the data will be used must be provided.
2. **Purpose limitation principle**: Personal data may only be collected for well-defined, explicit and lawful purposes and must not subsequently be processed in a manner incompatible with those purposes.
3. **Principle of data minimisation**: Personal data must be adequate, relevant and limited to what is necessary in relation to the purposes for which they were collected.
4. **Principle of data accuracy**: Personal data must be accurate and, if necessary, updated. Measures must be taken to ensure that inaccurate data are corrected or deleted.
5. **Principle of storage limitation**: Personal data should only be kept as long as necessary for the purposes for which they were collected.
6. **Principle of integrity and confidentiality**: Personal data must be managed in such a way as to guarantee their security, preventing unauthorised access or loss.
7. **Principle of accountability**: The data controller is responsible for adhering to data protection principles and must be able to demonstrate compliance. This involves record keeping and the implementation of appropriate security measures.

These principles must be respected by organisations, which define specific policies, procedures and documents that comply with the GDPR, to ensure proper management of users' personal data. According to [13] and [21] organisations provide licences, data sharing agreements, terms and conditions and waivers, to ensure clear rights for access, use and sharing of data.

Licences

A data sharing licence is a unilateral legal agreement that defines how data may be used, shared and re-used by others. Such licences clearly set out the conditions under which the data may be used, for instance by only allowing its use for non-commercial purposes or requiring that credit be given to the original author. Licences are often standardised and ready to use, and they are not limited in duration, but last as long as the agreements are fulfilled; one of the most widely used licences for sharing data is the Creative Commons, which due to its flexibility allows creators to choose from several options to customise the conditions of use of their works.

Contracts

Data sharing agreements are formal contracts or legal agreements that specify the terms, conditions and obligations relating to the sharing of data between two or more parties. These agreements then involve negotiation, where each party's legal team may be involved in drafting and reviewing the agreement to ensure that the terms are clear and legally binding. Data sharing agreements can be highly customised to meet the specific needs and concerns of the parties involved. Data sharing agreements can be comprehensive and cover a wide range of aspects beyond the simple use of data. They often include details on data ownership, data security measures, confidentiality, data management procedures, dispute resolution and more.

Waivers

In the context of data sharing, a waiver essentially means knowingly giving up specific rights or claims that one might otherwise have with respect to the shared data. With a waiver, the owner or provider of the data essentially declares that they will

not assert such rights or claims against those who receive or use the data.
What makes waivers particularly useful is that they are often very specific and precise. They state exactly what rights or claims are being waived and under what circumstances. For example, a data provider might decide to waive its copyright, but only to allow others to freely make copies and share the data.
These waivers play an important role as they bring clarity and transparency to data sharing agreements. This can be crucial especially when there are legal uncertainties or concerns. Furthermore, waivers act as a kind of legal safeguard, guaranteeing that no legal action will be taken in relation to the shared data. This guarantee is beneficial for all parties involved in the data sharing process.

Terms and conditions

Terms and conditions are a set of rules and provisions established by an organisation or digital platform. These documents are designed to define how users should behave when using a particular online service. They differ from licences in that they define general rules on the use of the platform and do not necessarily concern intellectual property rights or specific authorisations for the use of data.

2.2.2 Data sharing on Google systems

Data sharing has become a crucial component of our daily lives and of the worldwide economy, so global technology companies such as Google have taken a leading role in collecting, processing and sharing huge amounts of personal data. Data sharing in Google's systems also raises significant issues in terms of privacy, security and societal impact. In this chapter we will examine in detail how Google handles data sharing, analysing how this process takes place within some of its most used services, considering risks and benefits.

How Google collects user data

Google collects data in two different ways; in the first case, it retrieves the personal data provided by the user, which are those requested during the creation of a Google account, such as name, e-mail address, phone number, and others. The second case is much broader, data are collected through various tools, including Google analytics, which provides information on the use of its services by users. According to [8] and [9], the information collected is of various types and includes:

- Device information: They capture detailed data such as the type of computer or mobile device, its unique identifiers, operating system version and mobile network information.
- Log information: Data related to phone calls, IP address, device activity and cookies are stored directly automatically in server logs.
- Location data: When a user uses Google's services, information about his or her location can be managed. This location data can be determined with varying levels of accuracy through methods such as GPS, IP address and information from Wi-Fi access points and mobile phone signal towers.
- Unique application numbers: Some services have a separate application identifier. This identifier, together with the installation details, may be transmitted to Google when installing or uninstalling the service, or when the service periodically communicates with Google's servers, e.g. for automatic updates.
- Local conservation: Information can be collected and stored locally on the device through means such as web browser-based storage and data caching within the device.
- Cookies and anonymous identifiers: when a user visits a google service, one or more cookies or anonymous identifiers may be sent to his or her device. These are used to store user preferences or other data.

Google allows users to keep track of the personal data they collect, so that they can choose different privacy settings. As specified in [8], a user must be able to:

- Decide with whom to share information.
- Change the display of one's profile depending on the user who is viewing it.
- Review and manage specific categories of data associated with your Google account.
- Retrieve information from many Google services.

- Fully block cookies within the browser, though it may impact the proper functioning of certain services.
- Review and change a number of advertising preferences, including the option to remove specific advertising services.

Some data are stored for different periods and are processed separately according to their type or configuration. Personal data, of course, can only be changed or deleted by the user. However, other data, such as advertising information, are automatically deleted or anonymised after a certain duration. To avoid accidental or malicious deletion, data are securely removed, ensuring that they are only completely deleted from the servers after a predefined period of time, allowing for potential recovery.

Why Google collect user data

According to the official documentation [9] and [8], Google exploits the information collected for several purposes, but the main ones are:

- **Service improvement**: Google uses the personal data collected to improve the user experience and the quality of the services available. For example, this data is analysed to refine search algorithms, ensuring more relevant results, and to improve the effectiveness of email and cloud storage services.
- **Customising the user experience**: Personal data allow for greater customisation of the user experience, offering more related content and search results. One example is Google Play, which collects information on applications already installed on the device to provide recommendations based on one's preferences.
- **Targeted advertising**: The creation of user profiles by means of the data collected, enables the display of advertisements and ads closely related to the user's interests and habits. In the Google documentation it is specified that no sensitive data, private data and data contained in certain Google services such as Drive, Gmail and Photos are used, unless the user has provided the necessary consent.
- **Performance analysis**: Google collects technical data to evaluate the performance of its services, identify and solve various technical problems, and thus improve the optimisation of its products.

- **Account security**: The data is used to improve the security and reliability of the services offered. This is done by monitoring user activities to identify potentially fraudulent behaviour that may pose a risk to other users.

Google cannot share data collected from users as it wishes, but there are numerous restrictions that must be respected. In fact, it must operate in accordance with data protection laws to ensure that personal information is treated legally, fairly and transparently. The sharing of such information with external entities can only take place under specific conditions:

- **With the user's consent**: Personal data is only provided to third-party companies or services if the user provides explicit permission. This means that a user must always be informed about how such data is used, while retaining the possibility of giving or withdrawing consent at any time.
- **To domain administrators**: If a user's Google account is managed by a domain administrator, that administrator will have certain privileges related to the user's account. These privileges include the ability to:

    1. Suspend or close the user's account.
    2. Receive user account information as required by applicable laws, regulations, legal processes, or governmental requests.
    3. View statistics regarding the user's account, such as statistics related to installed applications.
    4. Limit the user's ability to delete or modify information or adjust privacy settings.
    5. Access or store information stored within the user's account.
    6. Change the user's account password.

- **For external treatment**: The data is shared with companies affiliated with Google or their trusted partners for processing in line with the privacy policy, while implementing appropriate security measures.
- **For legal reasons**: If deemed necessary, personal data may be disclosed to external parties for:

1. Adhere to the current Terms of Service.
2. Protection of Google's rights, property or security, to the extent necessary or permitted by law.
3. Comply with all relevant laws or regulations, legal procedures or government requirements.
4. Identifying, preventing or solving fraudulent activities or technical and security problems.

2.2.3 Google and third party sharing

When a user authorises access to their personal data by a third-party application, Google uses the OAuth protocol for authentication and authorisation, [7]. This ensures compliance with the 'least privilege' policy, which means that only the specific data requested by the application, for which authorisation has been granted and for which the user has given consent, can be accessed. Consequently, even if a third-party application were to somehow circumvent Google's security measures in order to access the user's data, it would not be able to retrieve any data other than that explicitly requested.
Initially, the third-party application must contact Google's API Console to obtain credentials, such as an ID, so that these are known only to the two parties involved. When it wishes to access private data stored by Google, it must obtain an access token, each of which can confer different types of privileges. Typically, access requests are sent, which often require an authentication phase on the part of the user; in this step, the user receives detailed information on what data is being shared and for what purposes, helping him to decide whether to deny or approve the request. If at least one authorisation is granted, an access token and a list of associated scopes are returned as a response. Scopes allow an application to request access only to the necessary resources, giving users control over the level of access they provide to the application. if authorisation is denied by the user, an error is returned. After acquiring the token, the third-party application transmits it to a Google API via an HTTP request header. Upon receipt of the token, Google performs checks to confirm its validity and its association with the user whose data is being requested. Once these checks are completed, the request is processed and the requested data is provided. It is essential to bear in mind that access tokens are only valid for the specific operations specified in the request header. Moreover, for security reasons, they have a limited lifetime. Therefore, when they expire, it is necessary to request an updated token or to repeat the OAuth authorisation process.
Google suggests using incremental authorisation as a best practice for gradually accessing a user's data or resources, avoiding requesting all authorisations at once. This approach involves initially requesting only the authorisations needed for a specific operation and, if necessary, allowing the user to grant additional authorisations at a later stage.

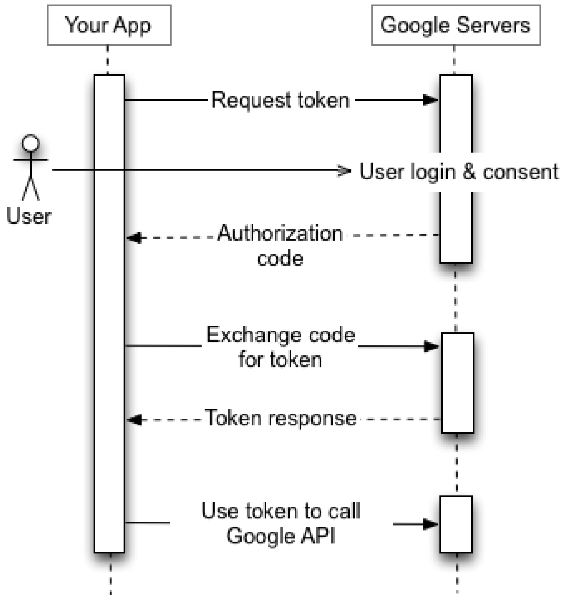


Figure 2.3: OAuth 2.0

The process for smartphones is quite similar, although there is an additional element involved: backup applications are frequently employed to facilitate the authorization process. These applications, often referred to as 'authorization managers' or 'app managers,' provide enhanced functionality for overseeing and managing the permissions granted to individual applications installed on the device. These app managers empower users to inspect and modify the permissions assigned to installed applications, offering them greater control and a more detailed level of customization for each app. It's worth noting that not all devices and operating systems come equipped with these features as part of their native functionality. For instance, certain Android devices may come with a pre-installed app management application, while others might necessitate the installation of third-party apps to access this added functionality. In contrast, on iOS devices, permission management is predominantly handled through the system settings.
The data sharing process within the Google suite generally follows a similar pattern, although with slight variations depending on the specific type of service and, consequently, the nature of the data involved. For this reason, we will look at some use cases to better understand the data that can be shared and the areas that can be authorised. In particular, we will focus on Google services such as Google Calendar,

YouTube and Google Photos.

Google Calendar

Google Calendar is a calendar application developed by Google, designed to simplify the management of your time and commitments; a calendar is a collection of related events, together with additional metadata such as summary, default time zone, location and so on. Each calendar is identified by an ID that corresponds to an email address and they may have multiple owners. An event is an object associated with a specific date or time range. Events are identified by a unique ID. In addition to a start and end date and time, events contain other data such as summary, description, location, status, reminders, attachments and so on. All event-related information can only be shared by Google to third-party applications or services if the user has granted the relevant authorisations. For this reason, it is often the user's responsibility to ensure what data will be shared and how it will be used. As mentioned earlier, to precisely determine the granted level of access, it's essential to identify the specific authorization areas and select the most limited one to prevent the sharing of unnecessary data[6]:

- https://www.googleapis.com/auth/calendar: Read/write access to calendars.
- https://www.googleapis.com/auth/calendar.readonly: Read-only access to calendars.
- https://www.googleapis.com/auth/calendar.events: Read/write access to events.
- https://www.googleapis.com/auth/calendar.events.readonly: Read-only access to events.
- https://www.googleapis.com/auth/calendar.settings.readonly: Read-only access to settings.
- https://www.googleapis.com/auth/calendar.addons.execute: Run as a Calendar add-on.

In addition to proper management of authorisations, one of the tasks of a good user is to be careful what he or she publishes. As we know on Google Calendar it is possible to make certain events public, putting personal information at risk, such as:

- **Personal information**: name, address, telephone number, e-mail address.
- **Event details**: dates and places attended.
- **Contacts**: names, e-mail addresses of your contacts.

Such sensitive data can be exploited by an attacker to carry out certain types of attacks, which we will analyse in the threats section.

You Tube

You Tube is a video sharing platform operated by Google, which allows people from all over the world to upload, view and share videos on a wide range of topics. In addition, the platform offers interaction features such as comments, ratings and the possibility to subscribe to favourite channels. Google collects data by monitoring user activities, such as searches, views, interactions, from which it can deduce a great deal of information. When a third-party application needs access to private data, it consistently relies on the OAuth 2.0 protocol as the underlying mechanism. While the process remains constant, what varies are the specific scopes granted to define the authorized resources to be accessed[11]:

- https://www.googleapis.com/auth/youtube: Manage your YouTube account.
- https://www.googleapis.com/auth/youtube.channel-memberships.creator: Display a list of your current active channel members, their current level and when they became members.
- https://www.googleapis.com/auth/youtube.force-ssl: View, edit and permanently delete YouTube videos, ratings, comments and subtitles.
- https://www.googleapis.com/auth/youtube.readonly: View your YouTube account.
- https://www.googleapis.com/auth/youtube.upload: Manage your videos on YouTube.
- https://www.googleapis.com/auth/youtubepartner: Viewing and managing resources and related content on YouTube.

- https://www.googleapis.com/auth/youtubepartner-channel-audit: Display private information of your YouTube channel relevant during the verification process with a YouTube partner.

Similar to other Google services, an attacker can obtain a user's specific personal data, such as name, profile picture, comments, favourite videos, behavioural patterns and interests, by observing their interactions.

Google Photos

Google Photos is a photo and video storage and sharing service that allows users to upload, organise and share their multimedia content. As in previous cases, Google only shares users' information after obtaining explicit consent; the authorisation protocol follows the same steps, with differences in the scopes provided, which are specific to Google Photos[10]:

- https://www.googleapis.com/auth/photoslibrary.readonly: Read access only; lists collection items and all albums, accesses all media items and lists user-owned albums, including those that have been shared with them. For albums shared by the user, the sharing properties are only returned if the photoslibrary.sharing scope has also been granted.
- https://www.googleapis.com/auth/photoslibrary.appendonly: Write access only; access to upload bytes, create media items, create albums and add enrichments. Allows the creation of new media only in the user's own collection and in albums created by the app.
- https://www.googleapis.com/auth/photoslibrary.readonly.appcreateddata: Read access to multimedia elements and albums created by the developer.
- https://www.googleapis.com/auth/photoslibrary.edit.appcreateddata: Edit access only; access to edit album titles, cover photos and media descriptions.
- https://www.googleapis.com/auth/photoslibrary.sharing: Access to call sharing; access to create an album, share it, upload multimedia items to it and participate in a shared album.
- https://www.googleapis.com/auth/photoslibrary: Provides access to the photoslibrary.appendonly and photoslibrary.readonly scopes, but since incremental access is considered an important best practice, it is not recommended.

Like other Google services, Google Photos offers different visibility settings. Users can choose to keep their albums private, share them publicly via a link or grant access to specific people via email or Google accounts. However, when albums are shared, photos and videos become accessible, potentially revealing all associated personal data, including dates, times, locations, people involved, comments and descriptions'.

2.2.4 Data sharing threats

As we have discussed so far, even apparently harmless personal information, when combined, can provide attackers with enough data to carry out potentially dangerous attacks. By analysing the information shared by a user, it is possible to accumulate enough details to launch attacks, some of which are classified as social engineering attacks. These attacks revolve around psychological manipulation, inducing victims to perform specific actions or reveal confidential information. One of the most widespread attacks within this category is phishing, which aims to acquire additional private data to facilitate other malicious actions. This attack involves various steps:

1. The attacker identifies the intended target and collects information about him, such as e-mail address, name, occupation and any other information that could make the phishing message credible.
2. The attacker creates a message that appears to come from a trusted source, such as a known company or service. This may be an e-mail, a text message, a telephone call or even a forged web page.
3. In the case of an e-mail or web attack, the attacker sends the message to the victim, pretending to be a legitimate service or requesting immediate action. It may contain a fraudulent link or a malicious attachment that aims to obtain victim's personal information.
4. If the victim clicks on the link or opens the attachment, they may be redirected to a fraudulent website that looks identical to the original one and be asked to enter their login credentials or other personal information.
5. Once the victim has provided the requested information, the attacker uses it for fraudulent purposes.

In the more generic case of phishing, attackers target a wide audience, sending generic messages in the hope that someone will fall into the trap; this method is known as generic phishing. However, there are other types of phishing, which aim at specific targets:

- **Spear Phishing**: Attackers conduct extensive research on their victims and adjust the messages to make them appear highly legitimate. This form of targeted phishing is particularly dangerous, as messages often imitate trusted sources and include convincing personal details.
- **Whaling**: Whaling is a variant of phishing mainly targeted at high-profile individuals, such as executives of companies or organisations. Attackers try to deceive these high-profile figures in order to gain access to sensitive information or to perform malicious actions, capable of causing significant damage.

An attack closely associated with phishing is spoofing, which increases the credibility of messages in the eyes of victims. Spoofing consists of making a message or communication appear to come from a source different from the real one, creating an appearance of authenticity. Various types of spoofing are closely related to phishing:

- **Email Spoofing**: Attackers can manipulate the email address so that it appears to come from a legitimate source. For example, a forged e-mail address may use a zero instead of the letter 'O', or replace an upper-case 'I' with a lower-case 'L'.
- **Caller ID Spoofing**: Attackers can falsify the phone number displayed on the recipient's screen during the call and pretend to be a customer service agent to collect personal information.
- **Website Spoofing**: Attackers can create websites that imitate an existing site by slightly altering the domain names. They usually distribute links to these sites using e-mail spoofing.
- **IP Spoofing**: Attackers may impersonate a trusted person or entity in their communications, making it appear that the message is from a known and trusted person.

- **DNS Spoofing**: Attackers can manipulate DNS records to make it appear that a malicious website is a legitimate site. As a result, victims can be redirected to a fraudulent website when they try to access the authentic site.

Phishing and spoofing, when executed effectively, provide cybercriminals with the additional information they need to perform subsequent malicious actions, without making the victim suspicious. In fact, many attacks share common characteristics with phishing and spoofing techniques, often incorporating them in their initial stages:

- **Identity theft**: Personal information acquired through phishing attacks can be exploited for identity theft. In these attacks, the perpetrator assumes the identity of the victim, enabling them to engage in malicious activities in their name, with possible legal repercussions.
- **Unauthorised access attacks**: This is one of the most damaging scenarios, in which the victim is tricked into revealing his or her account credentials. In this way, the attacker gains access to the account, allowing him to take malicious actions such as altering account settings, publishing unwanted content and stealing additional sensitive information, potentially opening the door to several types of highly dangerous attacks. In some cases, when a user uses a simple and weak password, attackers can avoid acquiring the victim's access credentials. Using personal information as a starting point, they can initiate a brute force attack, which involves several steps: first, a list of potential passwords is generated, then scripting automatically tests each one on the list for a login attempt. The process culminates when all passwords are exhausted or when a positive match is obtained that allows access to the account. The effectiveness of a brute-force attack depends on several factors, including the complexity and length of the password, the computing power of the attacker's hardware, and the speed of login attempts.
- **Malware attacks**: Phishing can be used to distribute malware, for instance through malicious attachments in e-mail messages or links to malicious websites; on the other hand, spoofing can trick people into believing that malware comes from trusted sources, inducing them to download it. As a result, attackers can introduce viruses into a user's account or device, gaining control, stealing additional information or demanding a ransom to restore access to data.

- **Cyberstalking**: Cyberstalking is not simply considered an attack, but constitutes criminal behaviour, often involving the use of phishing and spoofing techniques. In particular, spoofing can be used to hide one's identity and impersonate trusted persons to approach victims or send threatening messages.

Chapter 3

Problem Statement

The previous chapter emphasised the complexity inherent in data sharing. As we have observed, although many tools strive to ensure secure and reliable data management, they do not always succeed in safeguarding users from threats. Consequently, there is an urgent need for more accurate and adaptable solutions, that can respond to the many variables encountered in this vast digital realm. Hence the desire to create a threat model capable of analysing how a user shares his data and assessing whether this information can be retrieved by an attacker and exploited for malicious purposes.This model not only helps identify the various threats that can be executed by exploiting shared data, but also assesses whether the user has the necessary defences in place to prevent such attacks from succeeding.

The thesis work involved an extensive theoretical phase, followed by the development of the model. Initially, a review of various threat models covering a wide range of topics was conducted to ensure a proper understanding of the model's construction. In the course of the study, the focus shifted to the Google suite, with a detailed analysis of data sharing mechanisms in four specific services: Google Calendar, YouTube and Google Photos. Although these services share many processes, they differ in the types of data they handle. For each of these services, an investigation was conducted to identify the most exposed data and the potential threats that could arise if this data fell into the wrong hands. These theoretical foundations formed the basis of the threat model for data sharing. Once finalised, the model was revised and represented using Prolog, a programming language based on logic programming. Prolog allowed us to assess the validity of the system by incorporating various use cases and evaluating their security properties using the language's query capabilities.

Chapter 4

Solution Design

In this chapter, we will delve into the design of a solution based on a threat model that specifically addresses data sharing. Having gained an understanding of the fundamentals of threat modelling, as well as the potential threats and vulnerabilities related to data sharing in the previous chapter, we will now focus on the next step: the conversion of this analysis into tangible actions aimed at improving the security of a system or application.
Exactly as in [5], the model is characterised by 2 fundamental aspects:

- The analysis conducted is static, which implies that every time the system undergoes changes, the analysis must be re-run.
- Temporal aspects or events are not taken into consideration during the analysis.

A threat model makes it possible to abstract the components of a system under analysis and interpret the various connections between them according to predetermined relationships. A model is composed of various elements including:

- **Entities**: Entities represent components or actors within the system that can influence or be influenced by threats.
- **Threats**: Threats represent events or situations that could change a property, which can alter the security state of an entity.
- **Properties**: Properties represent the security knowledge of entities or their state, which can change based on actions taken. The effective management of these properties are essential to preserve the overall security of the system.

- **Relations**: Relations delineate the interactions between entities, threats and properties within the system. They help identify how threats may affect the properties and entities involved.
- **Rules**: Derivation rules conform to the theoretical principles of first-order logic. After providing a description of the system under consideration in accordance with the syntax and semantics of the model, these rules allow new information on the security status of the system and its entities to be derived mechanically.

4.0.1 Entities and threats

In the threat model, entities are categorized into four macro-categories, each of which is further subdivided into various elements. The first category pertains to data adequacy, which assess the quantity and quality of information available to an attacker for executing an attack. When an attacker possesses limited or insufficient personal data, such as only a first and last name, to carry out a potentially harmful attack, it falls under the category of insufficient data. Conversely, if the data acquired is adequate to facilitate an attack, the user may be at risk if protective measures are not in place. For example, information like the victim's first name, last name, date of birth, and email address can enable basic attacks. When additional personal information complements the aforementioned data, they aid in constructing a comprehensive victim profile and executing successful attacks. Lastly, there's sensitive data, including credentials and financial information, which, if in the hands of attackers, can lead to severe and substantial damage.

The second category concerns the visibility of personal data, which plays a crucial role in ensuring the security of users. When information is public and accessible to anyone, it falls into the category of **public visibility**. Sometimes, users may decide to restrict the visibility of certain data exclusively to a select group of chosen people, classifying it as **confidential visibility**. As a last option, there is **private visibility**, which applies only to private data intended to be accessible only by its owner.

The third category concerns the types of users that can be targeted in attacks on our system. The experience level of a user can significantly influence the outcome of an attack, leading to its classification into three distinct categories. In applying the model, assumptions were made about the visibility of specific data and potential security measures in place based on the user's level of experience. **Beginner users** are more likely to adopt a superficial approach, potentially exposing some personal information publicly or not paying adequate attention to sensitive data. Moreover, given their limited knowledge and awareness of potential risks, they are more susceptible to deception, making them an easy target in the eyes of an attacker. As for **intermediate users**, they are assumed to take more precautions to limit the public exposure of their data, keeping it mostly with confidential or private visibility. Compared to inexperienced users, they have a greater awareness of risks and protective measures, which makes them less vulnerable to attacks. Users who are highly experienced and actively keep themselves informed of the latest data threats are considered **expert users**. Consequently, their personal data is assumed to be exceptionally protected, making it extremely difficult for an attacker to deceive them and acquire exploitable information to conduct an attack.

The last macro-category focuses on protection measures; however, in order to understand the specifics, it is essential to first highlight the practical attacks that an attacker might deploy in a data sharing system, since the countermeasures chosen depend on the specific threats. At first, when an attacker has only limited personal information, his initial approach is to attempt to manipulate the user into revealing further personal information or to exploit inadequate security measures, if any. For this purpose, the most frequently used attacks are **brute force**, **phishing** and **spoofing**. Should these attacks be successful, an attacker would already be able to acquire additional personal information or even sensitive data. This, in turn, could facilitate the execution of new attacks that could cause even more serious damage, such as:

- More accurate phishing attacks like **spear phishing** or **whaling**.
- **Identity theft**.
- **Malware attacks**.
- **Unauthorised access**.
- **Cyberstalking**.

To mitigate or counter such attacks, various entities within the system must act as protective measures for other components. As already mentioned, the experience

level of users plays an important role, as increased **awareness** prepares them for potential threats. For instance, in the case of phishing or spoofing attacks, users can take simple precautions, such as checking the sender's e-mail address or being careful to click on links in suspicious messages to avoid falling victim to deception. Another effective countermeasure is the use of **antispam filters**, which can identify suspicious e-mails through specialised algorithms and automatically delete them or move them to a designated spam folder.
The use of a hard-to-guess, **strong password** is considered more of a best practice than an isolated security measure, but is nevertheless of considerable importance. To further enhance security, **two-factor authentication** (2FA) can be used. 2FA requires an additional level of access, guaranteeing safety even if the user's credentials are stolen. Within the system, it is essential to have a **monitoring** system capable of detecting unauthorised access to sensitive information at an early stage. Fast detection enables immediate reaction, preventing such data breaches from causing significant damage. **Antivirus** software plays a key role in detecting potential viruses or malware and neutralising threats before they can damage system components. To achieve a high level of protection, it is essential to keep both antivirus and **software update**, as they often include bug fixes and patches to resolve known vulnerabilities. Developers are constantly introducing security enhancements to strengthen the system's resistance against emerging threats.

4.0.2 Properties and relations

As already introduced in this chapter, properties reflect the security status of entities, which may vary within the system. For the realisation of the threat model on data sharing, the following basic properties were considered suitable and relevant for our analysis:

- **Compromised** Compromised(A, B): it indicates that an entity A has been compromised by an attack B.
- **Vulnerable** Vulnerable(A, B): it indicates that an entity A is vulnerable to an attack B.
- **Exposed** Exposed(A, B): it indicates that the data B has been exposed due to an attack B.

- **Damaged** Damaged(A, B): It indicates that data A has been damaged by an attack B, which implies that A may be lost or encrypted.
- **Recovered** Recovered(A): it indicates that A data has been recovered, usually after A has been damaged.
- **Public** Public(A): it indicates that a given data A has public visibility, allowing anyone to access it.
- **Confidential** Confidential(A): it indicates that a given data A has confidential visibility, allowing access only to users chosen directly by the owner to be part of a restricted group.
- **Private** Private(A): it indicates that a given data A has private visibility, allowing only those with credentials to access it.
- **Name** Name(A, B): it indicates the name A of a user B.
- **LastName** LastName(A, B): it indicates the last name A of a user B.
- **Username** Username(A, B): it indicates the username A of a user B.
- **Email** Email(A, B): it indicates the email address A of a user B.
- **BirthDate** BirthDate(A, B): indicates the date of birth of user A.
- **Credentials** Credentials(A, B): it indicates the credentials A of a user B.

After defining the basic properties, relations were developed, which can be classified into two types. There are relations between entities, which can influence how these entities respond to attacks:

- **Control** Control(A, B): it means that an entity A has control over an entity B. For example, a user A can control a data B
- **AccessTo** AccessTo(A, B): It implies that an attacker has gained access to a restricted group defined by a user A, allowing him to acquire a confidential data B visible only to members of the group.
- **Dupllicated** Duplicated(A): data A is duplicated.

Then there are relations between entities and attacks, which allow us to represent which entities are vulnerable to specific attacks and how they can protect each other:

- **Protect** Protect(A, B, C): it indicates that entity A is employed to protect entity B from an attack C.
- **Monitor** Monitor(A, B, C): it indicates that entity A is employed to monitor entity B against an attack C.
- **SecurityFailure** SecurityFailure(A, B, C): it indicates that an antispam filter A can no longer protect a user B against a phishing attack C if combined with a spoofing attack, against which it is not secure.

High-level properties were introduced, derived from the combination of the previous ones. These high-level properties serve to simplify the formulation of the derivation rules, which are essential for the proper functioning of the threat model:

- **Protected** Protected(A, B): Indicates that an entity A is protected from an attack B, assuming an entity C is responsible for this protection and is not compromised.
- **Monitored** Monitored(A, B): Indicates that an entity A is monitored with respect to an attack B, assuming an entity C is responsible for this monitoring and is not compromised.
- **Secure** Secure(A, B): Indicates that an entity A is secure from an attack B, when A is not vulnerable to B or is protected from B.
- **Detected** Detected(A, B): it indicates that if an entity A is compromised by an attack B and A is being monitored for that attack, then B is detected.

Similar to high-level properties, threats are defined within the model. By combining properties and relationships, it becomes possible to assess the feasibility of a particular type of attack against an entity:

- **Phishing** Phishing(A): it indicates that an attacker can execute a phishing attack against an entity A.

- **Spoofing** Spoofing(A): it indicates that an attacker can execute a spoofing attack against an entity A.
- **MalwareAttack** MalwareAttack(A): it indicates that an attacker can execute a malware attack against an entity A.
- **BruteForce** BruteForce(A): it indicates that an attacker can execute a brute force attack against an entity A.
- **UnauthorisedAccess** UnauthorisedAccess(A): it indicates that an attacker can execute an unauthorised access attack against an entity A.
- **IdentityTheft** IdentityTheft(A): it indicates that an attacker can execute an identity theft attack against an entity A.

4.0.3 Rules

After providing a description of the system in accordance with the syntax and semantics of the model, it becomes possible to apply the derivation rules. These rules facilitate the evaluation of the system's performance in different scenarios and the identification of potential vulnerabilities, allowing new knowledge to be gained about the security status of the system and its component entities. For example, in cases where the compromise of an entity is confirmed, these rules make it possible to deduce which other entities may be at risk. The rules identified for the threat model will now be explained, starting with the introduction of the rules defined through the combination of relations and properties to obtain high-level properties. When an entity A, which is not compromised, protects a user B against an attack C, B is protected:

$$Protect(A, B, C) \land \neg Exposed(A, \_) \land \neg SecurityFailure(A, B, C)$$
$$\rightarrow Protected(B, C).$$

When an entity A, which is not compromised, monitors a user B for an attack C, B is monitored

$$Monitor(A, B, C) \land \neg Exposed(A, \_) \rightarrow Monitored(B, C).$$

When a user A is not vulnerable to an attack B, or A is protected by B, A is considered secure from B:

$$\neg Vulnerable(A, B) \lor Protected(A, B) \rightarrow Secure(A, B).$$

When a user A is monitored for an attack B and the data C, held by A, is damaged by that attack, B can be detected:

$$Monitored(A, B) \land Damaged(C, B) \land Control(A, C) \rightarrow Detected(C, B).$$

By analysing the different entities and their respective states for each user, it is possible to determine whether an individual is susceptible to one or more of the attacks outlined in the model.
When the first name B, last name C and email D or the username and the email D of user A have been exposed, the attacker has enough information to carry out a phishing attack:

$$\begin{gathered}((Name(B, A) \land LastName(C, A) \land Exposed(C, \_)) \lor Username(B, A)) \\ \land Exposed(B, \_) \land Email(D, A) \land Exposed(D, \_) \rightarrow Phishing(A).\end{gathered}$$

When the email B of user A has been exposed, the attacker has enough information to perform a spoofing attack:

$$Email(B, A) \ \wedge \ Exposed(B, \_) \ \wedge \ Control(A, C) \ \wedge \ Exposed(C, \_) \\ \rightarrow Spoofing(A).$$

When user A is the victim of a phishing attack and is not secure against it, the attacker is able to carry out a malware attack:

$$Phishing(A) \wedge \neg Secure(A, phishing) \rightarrow MalwareAttack(A).$$

When the date of birth B, email C and name D of a user A are exposed, the attacker could have enough information to carry out an identity theft attack.

$$BirthDate(B, A) \wedge Exposed(B, \_) \wedge Email(C, A) \wedge Exposed(C, \_) \wedge \\ Name(D, A) \wedge Exposed(D, \_) \rightarrow IdentityTheft(A).$$

When personal data C and the first or last name of a user A have been exposed, the attacker could have enough information to carry out a brute force attack:

$$(Name(B, A) \vee LastName(B, A)) \wedge Exposed(B, \_) \wedge \\ BirthDate(C, A) \ \wedge Exposed(C, \_) \rightarrow BruteForce(A).$$

When the credentials B of a user A are exposed, the attacker can carry out an unauthorised access attack:

$$Credentials(B, A) \land (Exposed(B, phishing) \lor Exposed(B, identityTheft) \\ \lor Exposed(B, bruteForce)) \to UnauthorisedAccess(A).$$

Then, there is a set of rules that seeks to identify what conditions exist for an entity to be compromised within the system.
The first rule is intended to express how the degree of visibility of data alone is critical to its security. If a data A has public visibility, it will be compromised; similarly, if A has confidential visibility, but the attacker is able to access the restricted group of the user B who owns the data, it will be compromised. The definition of the property compromised is always characterised by the attack that led to the violation of the entity. In this context, where the entity corresponds to a data, the state of compromise depends on the condition of the data itself, rather than on a particular targeted attack; to preserve the structure of the property, a placeholder attack is defined to indicate the exposure of the data:

$$Public(A) \lor (Confidential(A) \land AccessTo(B, A)) \\ \to Exposed(A, visibilityExposure).$$

The next rules emphasise that when users are compromised or not secure from an attack, all their data are at risk, because they can easily be compromised. When a user A, which controls a given data B, is compromised by a phishing attack C, B will also be compromised:

$$Control(A, B) \land Compromised(A, phishing) \to Exposed(B, phishing).$$

When a user A, which controls a given data B, is compromised by an identity theft attack C, B will also be compromised:

$$Control(A, B) \land Compromised(A, identityTheft) \rightarrow Exposed(B, identityTheft).$$

When a user A, which controls a given data B, is compromised by an unauthorised access attack C, B will also be compromised:

$$Control(A, B) \land Compromised(A, unauthorisedAccess) \rightarrow Exposed(B, unauthorisedAccess).$$

When a user A is the victim of a brute-force attack and is not secure against it, the attacker can obtain user A's B credentials:

$$Credentials(B, A) \land BruteForce(A) \land \neg Secure(A, bruteForce) \rightarrow Exposed(B, bruteForce).$$

The next rules determine when a user is compromised. When data is compromised and the user possessing the data is vulnerable or unable to protect himself from an attack, the user will also be compromised.
When a user A is the victim of a phishing attack, in the absence of security measures against it, A becomes compromised:

$$Phishing(A) \land \neg Secure(A, phishing) \rightarrow Compromised(A, phishing).$$

When user A is the victim of an unauthorised access attack and is not secured against it, A is compromised:

$$UnauthorisedAccess(A) \land \neg Secure(A, unauthorisedAccess).$$
$$\rightarrow Compromised(A, unauthorisedAccess).$$

When a user A is the victim of an identity theft attack and is not secured against it, A is compromised:

$$IdentityTheft(A) \land \neg Secure(A, identityTheft)$$
$$\rightarrow Compromised(A, identityTheft).$$

When a user B is protected against phishing by a standard antispam filter, but is not secure against spoofing, the combination of the two attacks can bypass the protection, which means that the countermeasure has failed:

$$A = antispamFilter \land C = phishing \land Spoofing(B) \land$$
$$\neg Secure(B, spoofing) \rightarrow SecurityFailure(A, B, C).$$

The last two rules concern the potential loss or encryption of data due to a malware attack and the subsequent potential for data recovery.
When a user A suffers a malware attack and is not protected against it, private or confidential data B may be damaged:

$$(Private(B) \lor Confidential(B)) \land Control(A, B) \land MalwareAttack(A) \land$$
$$\neg Secure(A, malwareAttack) \rightarrow Damaged(B, malwareAttack).$$

When an attack B can be detected for a data A, which is damaged, and A has been duplicated, A can be recovered:

$$\boxed{Detected(A, B) \wedge Duplicated(A) \rightarrow Recovered(A).}$$

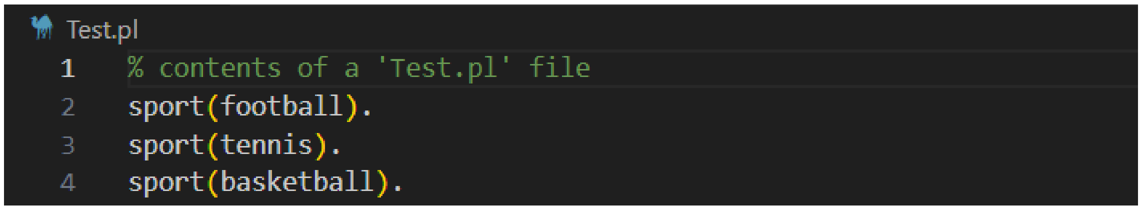
Chapter 5

Validation

The phase following the completion of the model's design involves the review and validation of the work done. This review is discussed in the current chapter, which addresses several use cases to validate different aspects of the newly created threat model for data sharing.
In order to perform this review, a tool is needed that allows for the definition of a model, the inclusion of use cases and the execution of queries. In this case, the tool chosen is XSB, a software that supports Prolog, a logic programming language known for its declarative programming capabilities.

5.0.1 Prolog rappresentation

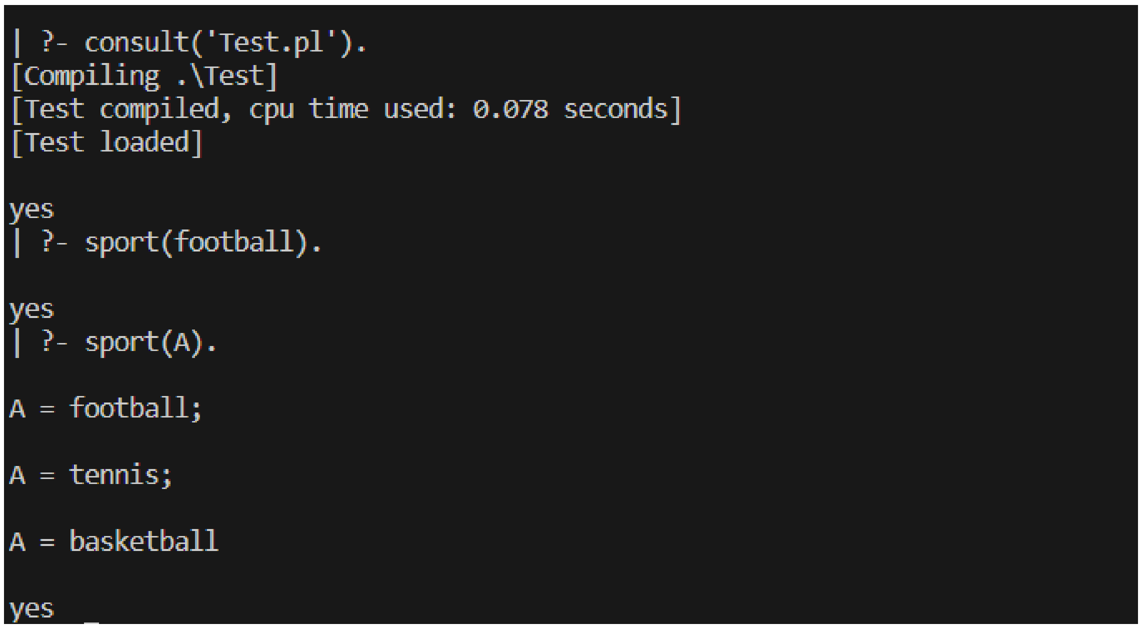
A Prolog programme comprises various elements: facts, rules and queries. A fact represents a unit of knowledge that the programmer considers significant, such as 'football is a sport' or 'tennis is a sport'. Here are some examples of facts described using the syntax of the language: A database is enriched with a set of facts using



```
Test.pl
% contents of a 'Test.pl' file
sport(football).
sport(tennis).
sport(basketball).
```

Figure 5.1: Definition of facts in prolog

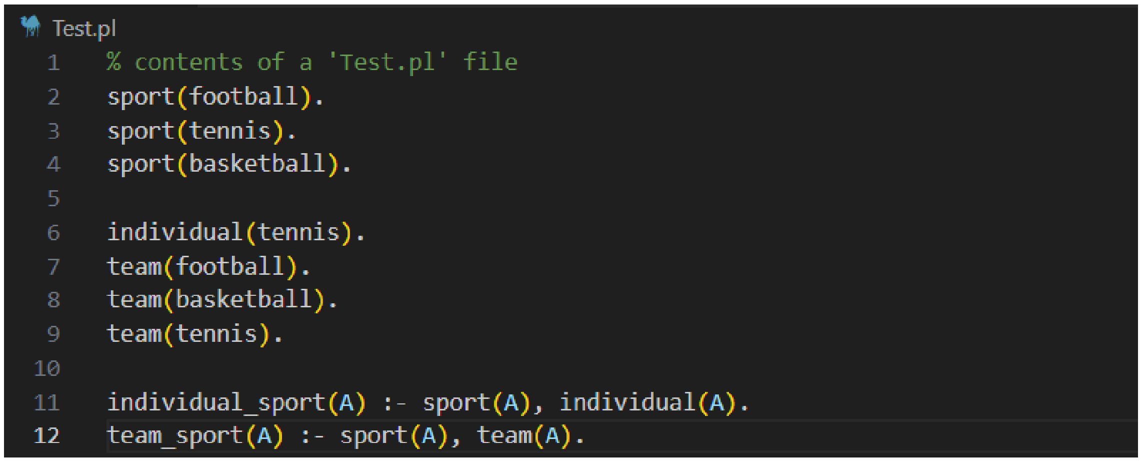
the query command, which adds to the rule engine's knowledge. A query serves as a method of asking questions. It is constructed from one or more 'goal', which are essentially questions that Prolog wants to answer by matching them with known values. The answers depend on the facts that have previously been entered into the system. Generally, Prolog shows the first solution when a query is successful, and the semicolon command ";" is used to check whether other matches have been found.



```
| ?- consult('Test.pl').
[Compiling .\Test]
[Test compiled, cpu time used: 0.078 seconds]
[Test loaded]

yes
| ?- sport(football).

yes
| ?- sport(A).

A = football;

A = tennis;

A = basketball

yes  _
```

Figure 5.2: Queries in prolog

The previous query identifies the values of A for which we can establish the association of A with a sport. In general, the main function of a rule engine is to evaluate statements using the available facts and rules. When it comes to specific, direct queries, it returns 'true' or 'false'. In more complex queries involving variables, it provides a set of values that satisfy the query.
A rule is an expression that establishes a connection between facts and specific criteria. It consists of a head, which indicates a predicate, and a body, which contains the prerequisites that must be fulfilled for the rule to be valid. When Prolog is queried, it tries to fulfil the conditions within the body of the rule and, if successful, performs the actions related to the rule.



Test.pl

```prolog
% contents of a 'Test.pl' file
sport(football).
sport(tennis).
sport(basketball).

individual(tennis).
team(football).
team(basketball).
team(tennis).

individual_sport(A) :- sport(A), individual(A).
team_sport(A) :- sport(A), team(A).
```

Figure 5.3: Definition of rules in prolog

The image above shows two simple rules, the result of which depends solely on the conditions enclosed in them. Since they are connected by a logical 'and', both conditions must be true to satisfy the rule.

5.0.2 Use case 1: Google Calendar

The first use case outlines a data sharing service, in which the different components and properties are articulated using the terminology of the developed model. The objective is to assess whether the exposed data poses a threat to the user and to evaluate the user's ability to defend himself against potential attacks.

In this use case, we will analyse how an attacker can exploit user vulnerabilities and gain access to sensitive information owned by the victim through a Google service such as Google Calendar. Sensitive data is classified as private, as it is only visible to the owner and requires private credentials for access. It is essential to consider the user's level of knowledge and awareness for each data-sharing use case, as this factor affects data visibility, potential vulnerabilities and user protections. We make assumptions about the level of visibility of the data, so we consider the victim's name, last name and e-mail to be public, which means they are exposed to an attacker. Information such as the location or time of an event is considered confidential information and therefore only visible to a small group of people chosen by the user. Others personal information are private, which means that no one should have access to them.

$$
\begin{array}{l}
Public(mario).\\
Public(rossi).\\
Public(mariorossi@gmail.com).\\
Confidential(eventLocation).\\
Confidential(eventTime).\\
Private(password).\\
Private(sensitiveInfo).\\
\\
Control(user, mario).\\
Control(user, rossi).\\
Control(user, mariorossi@gmail.com).\\
Control(user, eventLocation).\\
Control(user, eventTime).\\
Control(user, sensitiveInfo).\\
Control(user, password).\\
\\
Name(mario, user).\\
LastName(rossi, user).\\
Email(mariorossi@gmail.com, user).\\
Credentials(password, user).
\end{array}
$$

The user may be subject to various threats, being vulnerable to attacks such as phishing, brute force and unauthorised access attempts. The victim has no protective measures against phishing, but imposes limits between access attempts to deter brute force attacks and uses a password to prevent unauthorised access.

$Vulnerable(user, phishing).$
$Vulnerable(user, bruteForce).$
$Vulnerable(user, unauthorisedAccess).$

$Protect(attemptLimit, user, bruteForce).$
$Protect(password, user, unauthorisedAccess).$

Based on the definitions provided above, we can use some derivation rules to extract further information on potential attacks and the consequent security measures against them. The attacker has access to a significant amount of personal information, enabling the execution of various attacks. If a brute force attack would be successful, the attacker would gain access to the password, rendering it ineffective in protecting the user from unauthorised access attempts. However, not only does the attacker not have enough information to perform a brute force attack, but according to the high-level properties the user is secure from it. Furthermore, it can be deduced from the same rules that the user is not safe from phishing, but is secure from unauthorised access attacks.

$Name(mario, user) \land LastName(rossi, user) \land Exposed(rossi, phishing) \land$
$Exposed(mario, phishing) \land Email(mariorossi@gmail.com, user) \land$
$Exposed(mariorossi@gmail.com, phishing) \rightarrow Phishing(user).$

$\neg Secure(user, phishing).$
$Secure(user, bruteForce).$
$Secure(user, unauthorisedAccess).$

When the user becomes the victim of a phishing attack, because he or she does not have adequate protection, he or she becomes compromised. This compromise results in the exposure of one's data, particularly sensitive and confidential information, which in turn creates opportunities for more serious attacks. For example, the password that previously offered a level of protection is no longer effective, making the user vulnerable to unauthorised access attempts.

$$Phishing(user) \land \neg Secure(user, phishing) \rightarrow Compromised(user, phishing).$$

$$Control(user, eventLocation) \land Compromised(user, phishing) \\ \rightarrow Exposed(eventLocation, phishing).$$

$$Control(user, eventTime) \land Compromised(user, phishing) \\ \rightarrow Exposed(eventTime, phishing).$$

$$Control(user, sensitiveInfo) \land Compromised(user, phishing) \\ \rightarrow Exposed(sensitiveInfo, phishing).$$

$$Control(user, password) \land Compromised(user, phishing) \\ \rightarrow Exposed(password, phishing).$$

$$\neg Secure(user, unauthorisedAccess).$$

5.0.3 Use case 2: You Tube

The second scenario under consideration concerns another Google service, You Tube. Each You Tube account includes a description and contact information accessible to all users. As in the previous case, a potential attacker could use this data and exploit any identified vulnerabilities in the target's security to launch attacks aimed at stealing sensitive information. It is assumed that a user's first name, last name, email address and profile description are publicly accessible within the channel information, thus making them exposed to potential attackers. In contrast, other personal data, such as watch history, password and phone number, remain private and thus safe from intrusion.

$Public(mario).$
$Public(rossi).$
$Public(profileDescription).$
$Public(mariorossi@gmail.com).$
$Private(watchHistory).$
$Private(phoneNumber).$
$Private(password).$

$Control(user, mario).$
$Control(user, rossi).$
$Control(user, profileDescription).$
$Control(user, mariorossi@gmail.com).$
$Control(user, watchHistory).$
$Control(user, phoneNumber).$
$Control(user, password).$

$Name(mario, user).$
$LastName(rossi, user).$
$Email(mariorossi@gmail.com, user).$
$Credentials(password, user).$

The user could potentially face multiple threats, being susceptible to both phishing and malware attacks. However, it is important to note that the user is protected from phishing attempts by the presence of a standard antispam filter.

$Vulnerable(user, phishing).$
$Vulnerable(user, malwareAttack).$

$Protect(antispamFilter, user, phishing).$

Based on the definitions provided above, the system appears to be safe and adequately protected. Although an attacker has sufficient data to launch a phishing attack, the user is protected by the protective measures implemented by the spam filter. Likewise, while the user remains vulnerable to a potential malware attack, the attacker does not have the conditions to take advantage of it.

$Name(mario, user) \land LastName(rossi, user) \land Exposed(rossi, phishing) \land Exposed(mario, phishing) \land Email(mariorossi@gmail.com, user) \land Exposed(mariorossi@gmail.com, phishing) \rightarrow Phishing(user).$

$Protected(user, phishing) \rightarrow Secure(user, phishing).$
$\neg Secure(user, malwareAttack).$

$\neg MalwareAttack(user).$

By introducing a single alteration into the system and changing the initial description accordingly, the dynamics between the system components lead to a chain of events, creating a potential attack vector for the attacker. In our case, we assume that the user is also vulnerable to a spoofing attack. The combination of spoofing and phishing enables a more sophisticated attack, potentially bypassing defensive measures such as a standard antispam filter, and making the victim vulnerable to phishing attacks.

$Vulnerable(user, spoofing).$
$\neg Secure(user, spoofing).$

$Spoofing(user) \land \neg Secure(user, spoofing)$
$\rightarrow SecurityFailure(antispamFilter, user, phishing).$

$\neg Secure(user, phishing).$

From the moment an attacker carries out a phishing attack and the victim has no way to defend himself, the user will become compromised. This means that all sensitive data controlled by the user will be exposed and thus obtained by the attacker. The phishing attack can also be exploited to convince the victim to click on a malicious link, allowing a malware attack to be executed. At this point, all sensitive data will not only be compromised, but may also be at risk of loss or encryption.

$$Phishing(user) \land \neg Secure(user, phishing) \rightarrow Compromised(user, phishing).$$

$$Control(user, watchHistory) \land Compromised(user, phishing)$$
$$\rightarrow Exposed(watchHistory, phishing).$$

$$Control(user, phoneNumber) \land Compromised(user, phishing)$$
$$\rightarrow Exposed(phoneNumber, phishing).$$

$$Control(user, password) \land Compromised(user, phishing)$$
$$\rightarrow Exposed(password, phishing).$$

$$Phishing(user) \land \neg Secure(user, phishing) \rightarrow MalwareAttack(user).$$

$$Private(watchHistory) \land Control(user, watchHistory) \land$$
$$MalwareAttack(user) \land \neg Secure(user, malwareAttack)$$
$$\rightarrow Damaged(user, malwareAttack).$$

$$Private(phoneNumber) \land Control(user, phoneNumber) \land$$
$$MalwareAttack(user) \land \neg Secure(user, malwareAttack)$$
$$\rightarrow Damaged(user, malwareAttack).$$

$$Private(password) \land Control(user, password) \land$$
$$MalwareAttack(user) \land \neg Secure(user, malwareAttack)$$
$$\rightarrow Damaged(user, malwareAttack).$$

Chapter 6

Conclusion

This thesis aims to investigate the challenges and threats associated with data sharing in today's digital landscape. In particular, the focus of our examination is the threat model, a security approach that enhances the understanding of potential threat scenarios and helps to devise effective security strategies.
The primary objective of this work is to build a comprehensive threat model that effectively captures the complexities of data sharing systems. The model introduced is designed to represent the interconnections between system components, the properties and potential relationships between these properties and the various threats. Every aspect of a user is taken into account to ascertain vulnerability to specific threats or the ability to avoid such threats by using suitable protective measures. The result is a model that simplifies the assessment of a user's risks, enabling the timely implementation of appropriate countermeasures.
To verify the functionality of the model, it was implemented in Prolog for testing purposes, involving two use cases relating to Google services. By entering the components and properties of the system, it was possible to analyse and determine the user's level of protection.

In the future, there are several directions to improve the advancement of threat modelling. This could involve advanced analysis techniques, such as exploiting artificial intelligence to improve the detection of anomalous behaviour, or increasing automation to speed up threat analysis and improve overall efficiency.

Bibliography

[1] Edward G. Amoroso. *Fundamentals of computer security technology*. Pearson College Div, 1994.

[2] Data Compliant. *Data sharing under GDPR: What you need to know*. URL: https://datacompliant.co.uk/data-sharing-under-gdpr-what-you-need-to-know/ (visited on 08/03/2023).

[3] Larry Conklin. *Threat Modeling Process*. URL: https://owasp.org/www-community/Threat_Modeling_Process (visited on 08/03/2023).

[4] Victoria Drake. *Threat Modeling*. URL: https://owasp.org/www-community/Threat_Modeling (visited on 08/03/2023).

[5] Rodrigo Vieira Steiner Fulvio Valenza Elisa Karafili and Emil C Lupu. "A hybrid threat model for smart systems". In: *IEEE Transactions on Dependable and Secure Computing* (2022/10/11).

[6] Google. *Google Calendar Documentation*. 2023. URL: https://developers.google.com/calendar/api/guides/overview.

[7] Google. *Google Identity*. 2023. URL: https://developers.google.com/identity/protocols/oauth2?hl=it.

[8] Google. *Google Privacy and terms*. 2023. URL: https://policies.google.com/privacy/archive/20141219?hl=it#infocollect.

[9] Google. *Google Privacy and terms*. 2023. URL: https://policies.google.com/privacy?hl=it.

[10] Google. *GooglePhotos Documentation*. 2023. URL: https://developers.google.com/photos/library/guides/overview?hl=it.

[11] Google. *YouTube Documentation*. 2023. URL: https://developers.google.com/youtube/v3.

[12] Antonio Lioy. *Privacy protection*.

[13] Policy National Research Council, Board on Research Data Global Affairs, and Paul E. Uhlir Information. *For Attribution: Developing Data Attribution and Citation Practices and Standards: Summary of an International Workshop*. National Academies Press, 2012.

[14] Owasp. *Threat Modeling Cheat Sheet*. URL: https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html#threat-modeling-terminology (visited on 08/03/2023).

[15] RiskOptics. *Top Threat Modeling Methodologies*. 2022. URL: https://reciprocity.com/blog/top-threat-modeling-methodologies/.

[16] Bruce Scheneier. *Fundamentals of Computer Security Technology*. 1998. URL: https://www.schneier.com/wp-content/uploads/2016/02/paper-secure-methodology.pdf.

[17] Simplilearn. *What is Threat Modeling: Process and Methodologies*. URL: https://www.simplilearn.com/what-is-threat-modeling-article (visited on 08/03/2023).

[18] synopsis. *Threat Modeling*. URL: https://www.synopsys.com/glossary/what-is-threat-modeling.html# (visited on 08/03/2023).

[19] ThreatModeler. *The Ultimate Guide To Threat Modeling*. URL: https://threatmodeler.com/the-ultimate-guide-to-threat-modeling/ (visited on 08/03/2023).

[20] threatmodeler. *The evolution of threat modeling*. 2016. URL: https://threatmodeler.com/evolution-of-threat-modeling/.

[21] *Three types of agreement that shape your use of data*. URL: https://blog.ldodds.com/2020/02/21/three-types-of-agreement-that-shape-your-use-of-data/ (visited on 08/03/2023).

[22] windriver. *What is Threat Modeling*. URL: https://www.windriver.com/solutions/learning/threat-modeling (visited on 08/03/2023).