Formal Methods for Secure Systems

Computer Engineering@UniPi

Giovanni Enrico Loni

27 febbraio 2024

Indice

0 Introduction

0.1 Outline of the course

1. Dependability
    - Building high reliable computer-based systems
    - Quantitative evaluation of dependability
    - Threat modeling and risk assessment
    - Malware analysis
    - Cybersecurity engineering
2. Formal methods for security
    - Formal methods applied to security
    - Case studies: Data confidentiality, Security protocols, Cyber-physical systems security

0.2 Computer-based systems

Computer-based systems are everywhere, and the services they offer are very diverses. From that, we can easy understand why the **dependability**, which is the ability of the system to deliver the expected service, is a critical aspect of these systems, in particular in a security point of view.

A system should (or must) be able to deliver the expected service, even in the presence of faults, errors, and **attacks**. This is the main goal of dependability, which is as important as the functionality of the system, pheraps even more. TO achieve that, we have **Formal Methods** that provide to us a set of techniques and tools to design, verify, and validate computer-based systems, in a rigorous and systematic way, even in presence of faults and attacks.

1 Basic concepts and terminology

All the concept and the terminology that will be presented in this section derives directly from the paper "Basic Concepts and Taxonomy of Dependable and Secure Computing" by Avizienis et al. (2004). This paper is a fundamental reference in the field of dependability and security and it is the basis for the definition of the concepts and terminology that will be used in the entire course, as suggested by the professor.

1.1 Dependability

We can give to **dependability** a simple definition: given a system, which is designed to provide a certain service, the dependability is the ability of that system to deliver the specified service also in presence of faults and malfunctions. In other words *dependability is that property of a computer-based system such that reliance can justifiably be placed on the service it delivers*. Note that the latter definition stresses the need for a justified reliance on the service, which is a key aspect of dependability.

1.1.1 Computer-based systems

A computer-based system is a system that includes a certain number of components: each of them can be interconnected and have its own functionality. The components can be hardware, software, humans and the environment in which the system operates.

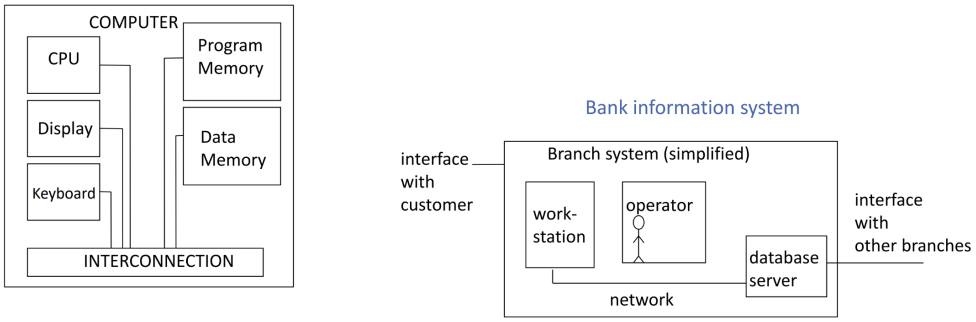

Figura 1.1: Computer-based system - C. Bernardeschi

1.1.2 Faults and Failures

We call a **failure** the inability of the system to deliver the expected service, and a **fault** the cause of that failure.

Example: if a cash machine delivers the wrong amount of money, we can say that the system has failed.

A fault causes an **error** in the state of the system, which lead to a **failure**. A failure can have different nature, such as physical, logical, human error or even as consequence of an attack.

Example: Logic Bomb. It's a piece of code that is inserted into a software system that will execute a malicious function when specified conditions are met.

```
legitimate_code();
if (date == "01/01/2020") {
    crash_system();
}
legitimate_code();
```

Computer Faults vs. Other Equipment Faults Computer faults differ from those of other equipment in several ways:

- **Subtler Failures**: computer failures are more subtle than outright crashes or sudden stops.
- **Information Storage**: computers store information in various ways, leading to a multitude of possible errors, both internally and externally.
- **Hidden Small Defects, Big Effects**: even small hidden defects can have significant impacts, especially in digital systems.
- **Complex Hierarchies**: computer systems are intricate hierarchies built upon hidden components.

1.1.3 Achieving Dependability

The dependability of a system can be achieved going trough a rigorous and engineered steps. Two main figures are involved, system and software engineers:

- **System engineers** are responsible for the design of the system, and they have to use analysis to model the dependability of the their design. From these, the software specifications are derived, and the possible changes to the system are evaluated, in order to accomodate software limitations;
- **Software engineers** are responsible for the implementation of the software, and they have to use the specifications to develop the software, and to test it in order to verify that it meets the requirements.

In general, it's crucial to understand that **dependability is not something that can be added to a system as an afterthought**. It must be considered from the very beginning of the design process, and it must be an integral part of the system, wusing a scientific and engineering approach.

1.2 The *system* entity

A simple but effective definition of a system is the following: a system is a entity that interacts with the environment and other systems; its boundaries are the common frontier between the system and the environment.

1.2.1 System's properties

A system has a **function**, which is the service that it provides to the environment, and it's described by its own functional specification. It also has a **behavior**, visualized as the sequence of states that the system goes through during its operation, and it's how the system implements its function. Then there is the **structure**, which is the way the system is organized, and it's described by its own structural specification.

From the user point of view, the system has a **delivered service**, which is the result of the interaction between the user and the system, that is the behavior of the system as perceived by the user. Obviously, the user can be seen as another system that interacts with the system under consideration.

1.2.2 System's requirements

First of all, we need to define the problem that the system has to solve, and then we have to define the requirements that the system has to meet, and then we have to define both functional and dependability requirements. Pay attention to the difference between the system's function and the system's specification: the former is the service that the system provides, while the latter is the solution implemented to provide that service. In the end, we define the **correctness** of the system, which is the ability of the system to deliver the specified service.

1.3 Dependability tree

Take in consideration the following dependability tree:

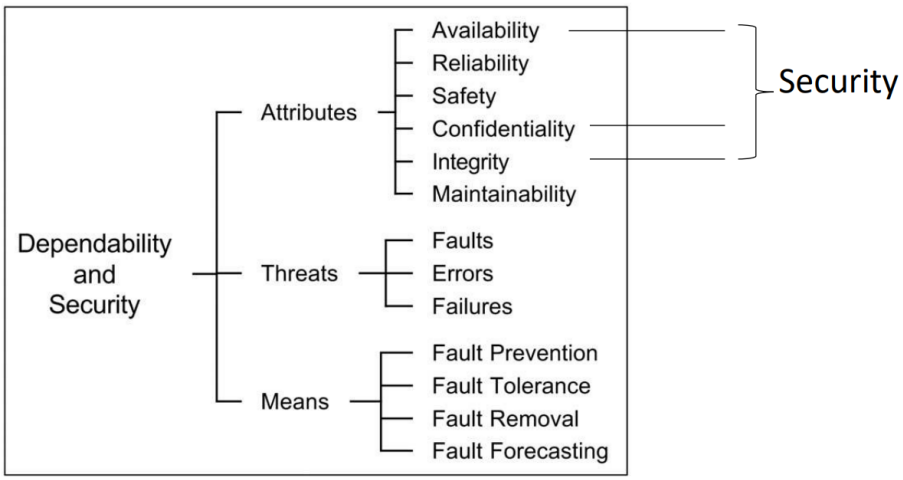

Figura 1.2: Dependability tree - Avizienis et al., 2004

1.3.1 Threats to dependability

As we said before, a **correct service** is delivered if the service is delivered in accordance with the system's specification. When this doesn't happen, we have a **service failure**, which is one the possible states of the system:

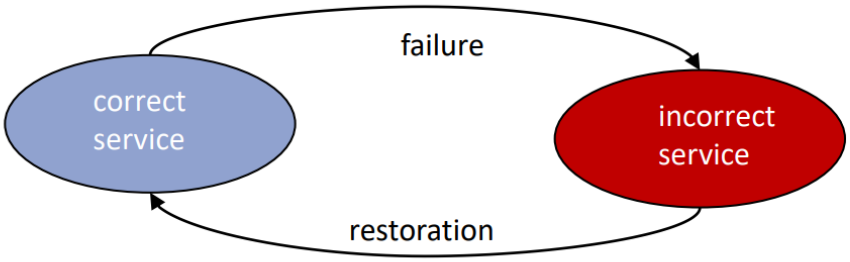

Figura 1.3: Service failure - C. Bernardeschi

We call **service outage** the period during which the system is not able to deliver the service, a **service degradation** the period during which the system delivers a service that is not in accordance with the specification, such as a subset of the services. We also recall the **chain of threats** to dependability: **faults** causes **errors**, which lead to **failures**: note that many errors don't cause failures because they don't reach the external state of the system.

Faults can be **dormant**, that is they are present in the system but they don't cause errors, and **active**, that is they cause errors, and thay can be **external** or **internal**: from the latter we can extract the definition of **vulnerability**, which is the property of the system that allows an external agent to cause a fault.

Example: Trapdoor. It's a hidden entry within a system that allows an attacker to bypass security measures.

```
username = read_username();
password = read_password();
if (username == 'dummy_user'){
    //note that the password is not checked
    grant_access();
}
if (username.isValid() && password.isValid()){
    grant_access();
}
```

From the past example we can learn that, having in mind the dependability tree, to achieve security only the authorized actions have to be allowed, and the confidentiality and the integrity of the data have appears in case of improper or unauthorized actions.

1.3.2 Dependability attributes

The dependability of a system can be described by a set of attributes, measurable and quantifiable in terms of probabilities:

- **Availability**: the readiness for correct service;
- **Reliability**: the continuity of correct service;
- **Safety**: the absence of catastrophic consequences on the user and the environment;
- **Confidentiality**: the absence of unauthorized disclosure of information;
- **Integrity**: the absence of improper system state alterations;
- **Maintainability**: the ability to undergo modifications and repairs.

We also briefly present the concept of **trust** between systems, that express the dependance of dependability of the system A from the dependability of the system B.