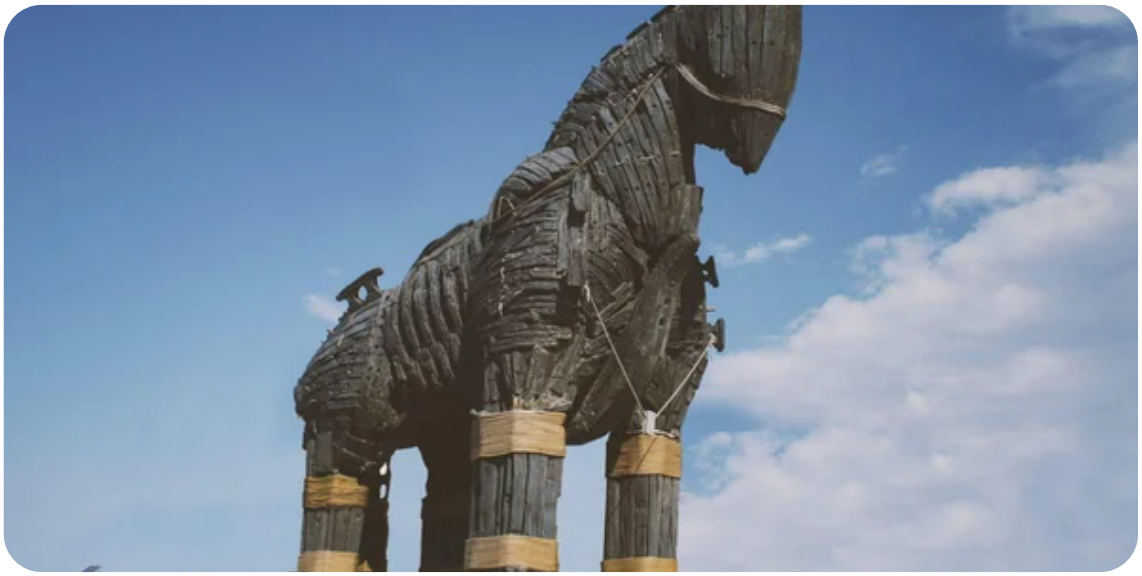
Sapienza, University of Rome
Course of Applied Computer Science and Artificial Intelligence (ACSAI)
3rd Year, 1st Semester

Cybersecurity



Notes by Leonardo Biason
Course taught by prof. Angelo Spognardi

Sapienza
Università di Roma

About these notes

Those notes were made during my three years of university at Sapienza, and **do not** replace any professor, they can be an help though when having to remember some particular details. If you are considering of using *only* these notes to study, then **don't do it**. Buy a book, borrow one from a library, whatever you prefer: these notes won't be enough.

License

The decision of licensing this work was taken since these notes come from **university classes**, which are protected, in turn, by the **Italian Copyright Law** and the **University's Policy** (thus Sapienza Policy). By licensing these works I'm **not claiming as mine** the materials that are used, but rather the creative input and the work of assembling everything into one file.

All the materials used will be listed here below, as well as the names of the professors (and their contact emails) that held the courses.

The notes are freely readable and can be shared, but **can't be modified**. If you find an error, then feel free to contact me via the socials listed in my website. If you want to share them, remember to **credit me** and remember to **not** obscure the **footer** of these notes.

Bibliography & References

[1] William Stallings, Lawrie Brown. (2018). *Computer Security Principles and Practice (Fourth Edition)*. Pearson

The "*Cybersecurity*" course was taught in the Winter semester in 2024 by prof. Angelo Spognardi (spognardi@di.uniroma1.it)

I hope that this introductory chapter was helpful. Please reach out to me if you ever feel like. You can find my contacts on my website. Good luck!

Leonardo Biason
→ leonardo@biason.org

Contents

Chapter 1
Introduction

Nowadays, there is a great attention regarding computer security and how to ensure privacy, mostly because of the rapid growth of internet and telecommunication systems. It has also become more important to secure data on machines, may they be connected to the internet or not. A word usually pops up when discussing computer security, and that is **cybersecurity**. Nowadays we have many definitions, but we'll stick to the one provided by the **NIST** (**National Institute of Standards and Technology**):

Definition

Cybersecurity

Prevention of damage to, protection of, and restoration of computers, electronic communications systems, electronic communications services, wire communication, and electronic communication, including information contained therein, to ensure its availability, integrity, authentication, confidentiality, and nonrepudiation.

In general, cybersecurity aims to ensure **computer security**, so deploying **confidentiality controls**, **integrity controls** and **availability controls** of the data and of the **computers' assets**, such as software, hardware and firmware. But what do we define as an **asset**?

Definition

Asset

An **asset** is an important object or collection of objects that are important for a specific entity, and that must be protected. Examples of assets may be the hardware, the software, the data or the network.

An example of used attack on the firmware the *meltdown attack*, which attacked the pipeline of the computer's CPUs; software attacks could be when there is a *man-in-the-middle*, so when eavesdropping happens; an instance of hardware attack could instead be the *voltage glitching*.

An information system is extremely complex, and works only thanks to the interaction of several elements. This is why protecting an information system is also complex. But we can't protect everything, so where should we put the priority?

The most important thing to protect is **data**, be it personal or company related. We can have a complete definition of what personal data is thanks to the GDPR (to make a parenthesis, privacy is something defined just recently. We could say that privacy is the right to decide who can access to your own data).

In the cybersecurity scenery, we have multiple factors that are involved. Some of them have precise definitions. Here there is a list of some useful key concepts:

DEFINITION

Security Concepts

- **Threat agent** (or **adversary**): the entity who conducts detrimental activities;
- **Countermeasure**: a procedure which involves a device or special techniques that must impair the adversarial activity;
- **Risk**: a measure which denotes how much an entity is threatened by a given circumstance or event;
- **Threat**: any circumstance that may pose a risk to an entity by a malicious agent;
- **Vulnerability**: is something that can be exploited or triggered by an adversarial entity.

Threats from a malicious agent can have multiple forms. We mostly recognise 4 types of attacks:

- **Active attack**: an attempt to alter one or multiple resources of a system or affect their operations;
- **Passive attack**: an attempt to learn information from a system, without altering the system's resources;
- **Inside attack**: an attempt to alter one or multiple system resources from an entity which is within the security perimeter of the system. Such an entity is authorized to access the system resources, but it uses them for non-intended purposes (usually malicious ones);
- **Outside attack**: an attempt to alter one or multiple system resources by an entity located outside the security perimeter, which is not authorized to use the system's resources;

1.1 C.I.A.

Cybersecurity aims to provide 3 major key points, which are shared under the C.I.A. acronym: **Confidentiality**, **Integrity**, and **Availability**.

DEFINITION

Confidentiality

We define with **confidentiality** the avoidance of unauthorized disclosed information

Some tools that can enforce confidentiality are:

- **Encryption**, which is the transformation of a piece of information by using a secret, called **encryption key**, so that the message can be read again in its decrypted form only by using another secret, called **decryption key** (in some cases, the encryption key is the same as the decryption key). Clearly, the longer the key, the longer it may take for a malicious agent to guess it and break the encryption;

- **Access control**: the establishment of rules and policies that limit access to some assets to a restricted amount of users and/or systems with a "*need-to-know*";
- **Authentication**: the determination of the identity or role that a given user has. There are multiple ways to do this, and it usually employs a combination of:
  - the usage of a physical item that the user has;
  - the usage of an information that the user should know;
  - the usage of information regarding the user (such as the fingerprint, the face landmarks positions, etc.).
- **Authorization**: the action of determining if a user is allowed to access a system's information or data (in general, a system's assets), which is determined by an access control policy;
- **Physical security**: the establishment of physical barriers to protect the system's assets.

DEFINITION

Integrity

Integrity is the property that an asset has not been altered in an unauthorized way

Some ways for ensuring integrity are **backups** (the periodic copy of data), **checksums** (the mapping of a file or collection of files to a unique numerical value. By adding, removing, or modifying the data, the checksum changes as well), and **data correcting codes** (methods that allow detecting and quickly fixing eventual errors within data).

DEFINITION

Availability

Availability is the property that an asset can be accessed and modified in a timely fashion by only the users authorized to do so

Some tools that can enforce availability are **physical protections** over the system and **computational redundancies**, ensuring the availability of a given data or collection of data on multiple instances of the system to serve as fallbacks in case of failures.

1.2 Authentication

We said that one of the key concepts of cybersecurity is to detect who is authorized to perform certain actions and who's not. We also need a way to assess that such user is actually authorized. This is where authentication comes into play.

The NIST defines the concept of authentication as follows (from NIST SP-800-63-4):

DEFINITION

Authentication

Authentication is the process of **establishing confidence** in user identities that are presented electronically to an information system

Some examples of authentication could be biometrics (although this kind of authentication has a great margin of error) or the use of physical cards. There are two basic authentication requirements:

- **Uniquely identify the information** of some users, processes, or devices, who could be acting on behalf of someone else;
- **Authenticate** (or **verify**) if the identity of the users, processes, or devices is genuine.

After these basic requirements, we can also implement other secondary security requirements, such as asking for a 2FA (Two-Factor Authentication) code, enforcing a minimum length for the password or the requirement of some characters to be used, the encrypting of the password, etc...

How can we store passwords though? We can't store them in plaintext, since a breach into the system containing a batch of passwords would leak them, so we have to "hide" them. We can instead encrypt them through some function: this way, even if the password got leaked, then only the encoded version would be leaked, making it harder (if not impossible) to recover the true password. This is the case of **hash functions**.

DEFINITION

Cryptographic Hash Functions

A **hash function** is a function f which maps a string of any length n to a string o of finite length (it is made of a number L of bits) plus some padding p (so the following holds: $|o| = L + p$). Such function is a **one-way function**: the output can't be reverted.

$$f : n \longmapsto l$$

Hash functions have some key properties:

- the length $|o|$ of the output is constant, and has a fixed size, **independently from the input length**;
- it's **impossible** to revert;
- it's **efficient** to compute;
- although possible, it's **hard** to find two input values which will produce the same output.

These key properties make the hash function widely employed in the cybersecurity field. Hashing though may not be enough: UNIX instead takes a slightly more complicated approach.

The UNIX Legacy method has that, when storing a new password, UNIX takes the password and a **12-bits salt**, which is a 12 bits code, needed to slightly modify the password before hashing it. Why do we need the salt? Because if two users, for instance, had the same password, then the hash would be identical. With different salts instead, we slightly modify the password, making the hash different even if someone had the same

password. Once hashed, UNIX would save into a table the **user ID**, the **salt**, and the **hash code**.

Another reason why we should use salts is that we can slow down the brute forcing of a password, because for each brute force attempt to convert one password, we would need to also guess the right salt.

Nowadays, UNIX uses a loop of 1000 MD5 crypting routines, has no limitations on the input length, and produces a 128 bits hash value. Such hashes are stored in the `/etc/shadow` file (previously in `/etc/passwd`) in such a format: