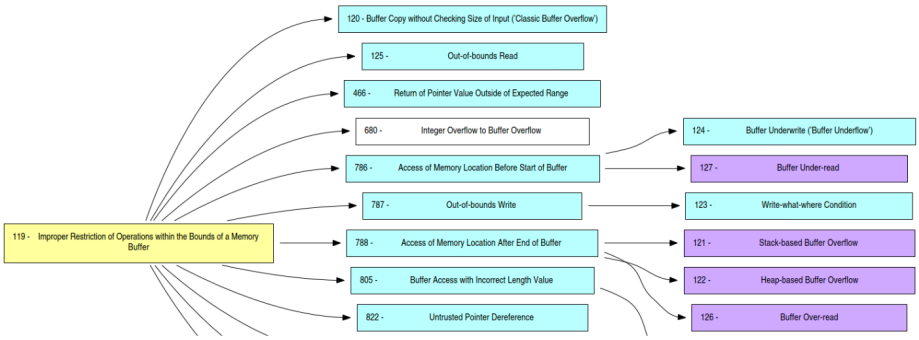
What is the underlying mistake? (CWE)

- Difference between bugs (SW mistakes) and vulnerabilities: a bug is an error of flaw that causes the software to produce an incorrect result of to behave in unintended ways, a vulnerability is a bug with violation of some security property (è un sottoinsieme dei bug)
- Each vulnerability may due to one or more mistakes → vulnerabilities can be grouped based on the type of mistake (weakness)

CWE: common weakness enumeration

- Is a list constructed one and for all and periodically revisited
- Has many hundreds of elements and a hierarchical structure
- Just a glimpse…→
- A vulnerability can be associated with one or more CWE, it is useful for understanding its nature quickly



- Examples: risk evaluation → successful exploitation of these vulnerabilities could allow an attacker to gain control of the device, escalate privileges or execute arbitrary code, some of these vulns are use of hard coded password (CWE 259), out fo bounds write (CWE 787), improper authentication (CWE 287), execution with unnecessary privileges (CWE 250)
- Immagine top CWE list of the year…arbitrary formula that combines count and severity of vulns of the last two years

How to identify them unambiguously? (CVE)

- How to know if two vulns are the same one or not

CVE: common vulnerabilities and exposures

- Every publicly known vulnerability has a unique ID (CWE-YYYY-NNNN..NN)
- It is extremely important to avoid any confusion
    1. Submission to a certain consortium (https://www.cve.org)
    2. Analysis followed by: assignment of a new CVE or see if it another instance of an existing CVE
- It is the standard the facto today
- Examples…
- CVE are associated only with widely used products, so vulns of hypothetical application created by us will not have any CVE
    - Vulnerabilities of home-made/special purpose applications will not be in any public database, discovering and assessing those vulns is up to their owners/users
- Sometimes, widely used products have vulns that do not have a CVE
    - Because it may be caused only by an insecure default configuration or the vendor claims that the vuln is actually a feature (an interned behavior)
- Vulnerabilities that are particularly interesting are sometimes given fancy names in the media (petit potam, goto fail, dragonblood…)
    - Is important to always try to refer to their CVE to avoid any inaccuracies
- The practical risk of a vulnerability depends on many factors (impact, difficulty of exploitation…) and it is hard to asses. Many vulns with a CVE have little to none practical risk
- CVE submissions are now receiving an excessively superficial scrutiny
- Many CVEs (categorized vulns) are not really relevant → the reports have to be analyzed carefully and we have to prioritize vulns that are really being exploited

What is the risk of a given vuln? (CVSS)

- Vulnerability severity depends on many factors: impact, difficulty of developing exploit, difficulty of injection (local or remote, with user intervention or automatic…), when there will be a patch, what workarounds can be used in the meanwhile
- Vulnerability score: there are several standards for quantifying severity by means of a single number, the most widely used (the de facto standard) is Common Vulnerability Scoring System (CVSS)

CVSS: common vulnerability scoring system

- Base score: quantifies intrinsic characteristics
  - Immutable: does not depend of the existence of exploit/patch
  - Independent of the context: does not depend on how many systems, their network exposure or how critical they are
- Full score: quantifies also those additional characteristics and more
- Usually when we talk about CVSS we mean the CVSS base score

CVSS base score

- Input: 8 categorical features (impact + exploit injection)
- Processing: translation of the categories to numbers, with arithmetic formulas + IF-THEN-ELSE
- Output: score in range 1-10 discretized in 5 categories (none 0.0, low 0.1-3.9, medium 4.0-6.9, high 7.0-8.9, critical 9.0-10.0)
- The formula to get the score is defined by a standard, for any given vulnerability is computed by NVD (national vulnerability database)
- NIST NVD (National institute for standards and technology) provides an CVSS base score and the values for input categories
  - Input categories are not our categories for impact/injection (that take a slightly different perspective)
- CVSS base score is
  - Immutable and not localized: CVSS is a method used to supply a qualitative measure of severity, is not a measure of risk
  - A vulnerability with low score in a system that is vital in my environment is very high risk for me
  - A vulnerability with high score can be not important at all or almost unreachable in my environment
- Vulnerability chaining: many attacks are executed by chaining multiple vulnerabilities, a low score vuln could enable very severe chains

NB quantitative does not mean objective and meaningful

- CVSS is just one of the many possible ways of distilling many properties in a single number (never use numbers blindly)

Which software is more secure?

- We cannot tell if one sw is more secure than another → there is not a good way to quantify how secure a software is, one can invent many quantification, but they will not be really meaningful
- The security of a software depends on many of its properties: quantifying each property is very hard, and combining those quantification in a single or few figures is even harder (certain thing simply cannot be measured (punto e basta)
- The vulns we know about a sw depend on many factors, like how many people are looking for them, on their effort and on thei technical skills. There could be many more highly talented people scrutinizing a certain software instead of another
- How to combine the known vulns, the ones that for which patch exists and the zero days vulns?
- A software can be considered less secure because it has more vulns, even if these vulns are much more harder to exploit than in other softwares
- The known vulns for one sw in this moment are usually different from the one that will be in the future (mi sembra abbastanza ovvio)
- How the vendor reacts to a vulnerability is much more important than the vulnerability (Matt Blaze)
  - How quickly does the vendor release patches, how interacts with vuln researchers…
- Android upgrades have problems: the time to patch tents to be long and not uniform (manufacturer, model, android version), end of life tent to be short and also not uniform (Manufacturer, model). The key reason is that there are many actors (Google, TCL providers, manufacturers)
- Apple/windows are much better from this point of view, end of life and time to patch are "good" and uniform, it is mostly because there is only one actor (tutto viene gestito da loro, per forza è più facile)