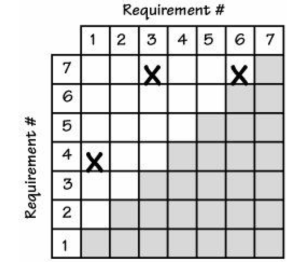
1. Introduction to Computer Security

Computer Security Courses @ POLIMI

Basic Questions

- What is a *secure system*?
- What is (computer) *security*?
- How do we engineer secure systems?

Basic Security Requirements



The so-called ***CIA Paradigm*** for information security states three requirements:

- **Confidentiality:** information can be accessed only by authorized entities.
- **Integrity:** information can be modified only by authorized entities, and only in the way such entities are entitled to modify it.
- **Availability:** information must be available to all the parties who have a right to access it, within specified time constraints.

"**A**" conflicts with "**C**" and "**I**": engineering problem.

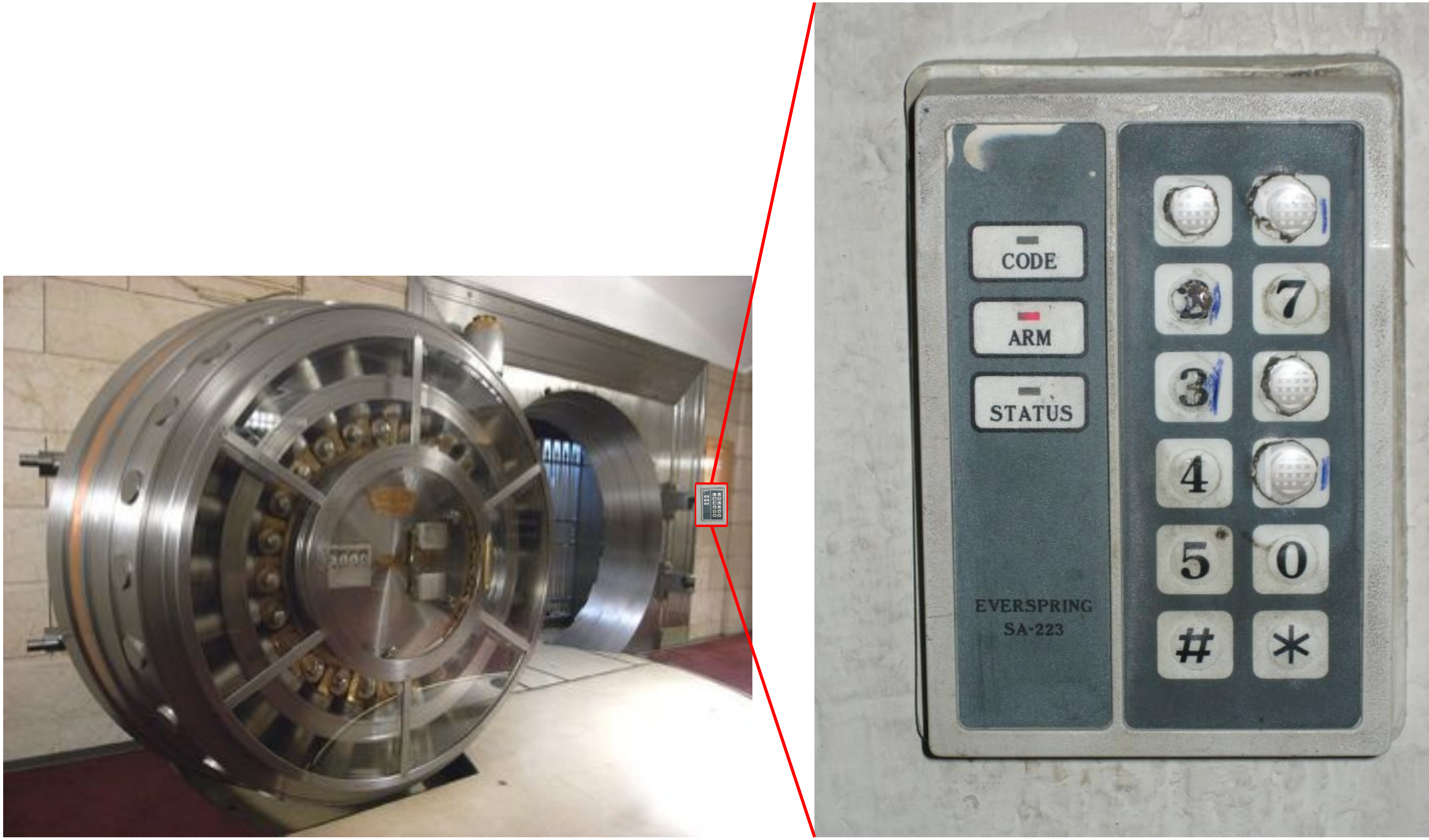
Security as an Engineering Problem

We need some concepts to frame it:

- **Vulnerabilities**
- **Exploits**
- Assets
- Threats
- Risks


IS THIS SECURE?
WHY DOES THIS "FEEL" SECURE?

The devil is in the details (1/2)



The devil is in the details (2/2)

Security door at some random airport.

Vulnerabilities *vs.* **Exploits**

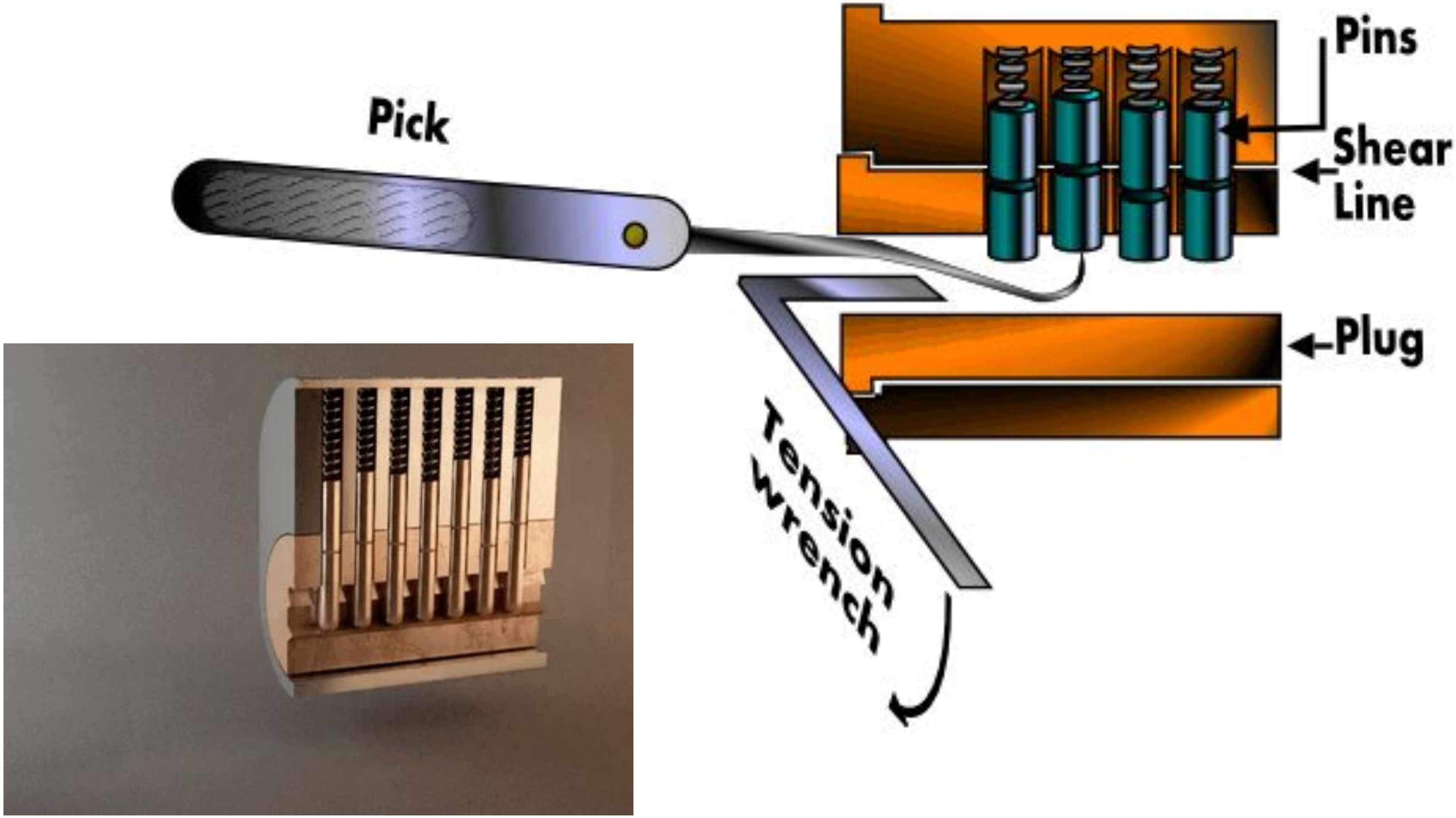
Vulnerability: something that allows to violate one of the constraints of the CIA paradigm.

- Examples:
  - Mechanical mismatches of pins in physical locks
  - software that fails to check the size of attachments

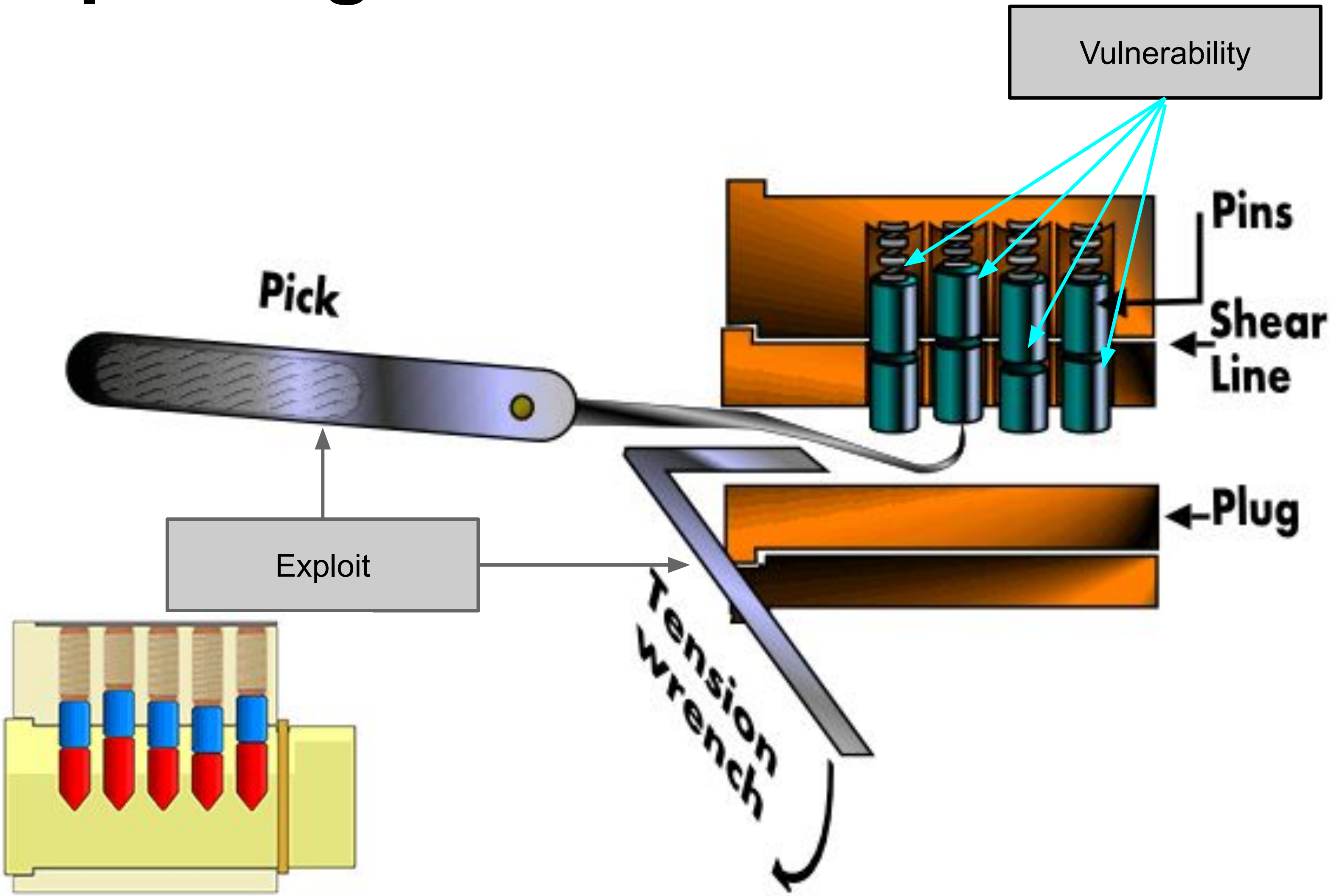
Exploit: a *specific way* to use one or more vulnerabilities to accomplish a specific objective that violates the constraints.

- Example:
  - Lockpicks and lock picking techniques
  - A large attachment leveraging the missing check

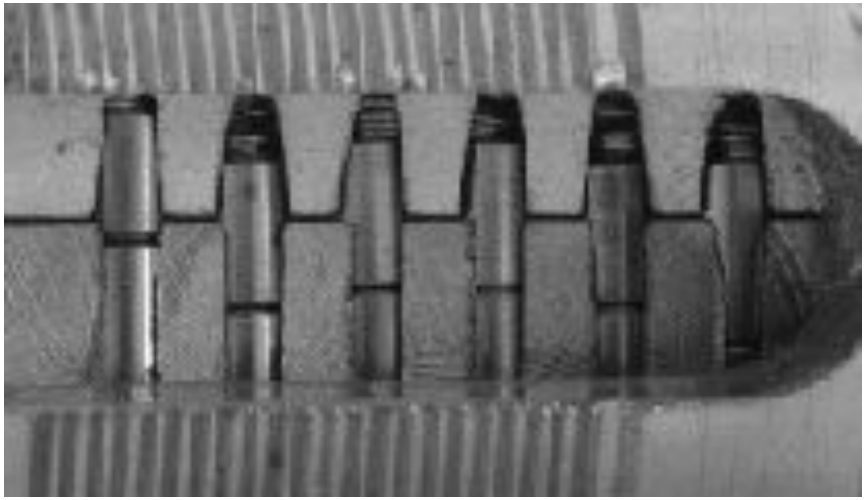
Exploiting a Vulnerable Lock

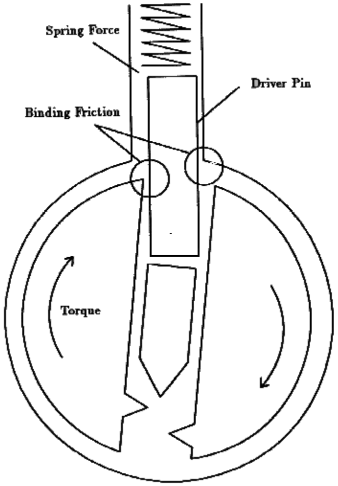


Exploiting a Vulnerable Lock



The Devil is in the Details

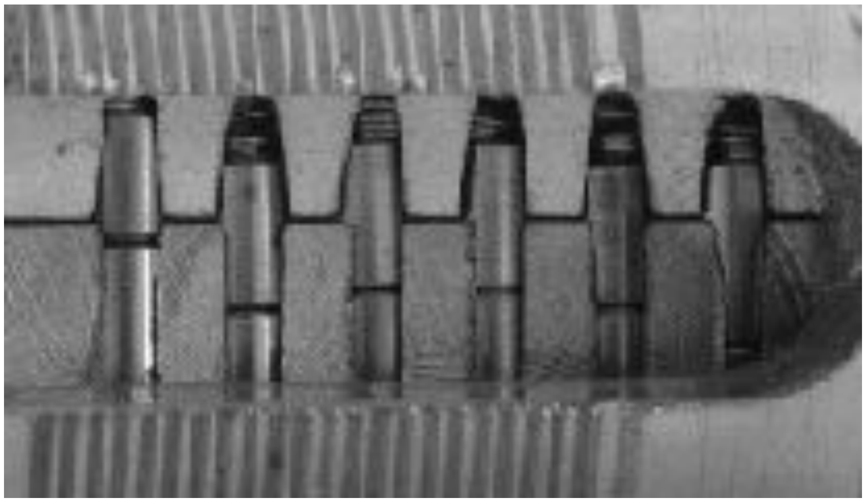


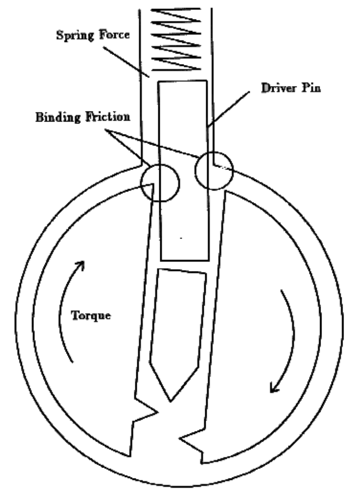


(a cheap lock)

How can we “fix” this vulnerability ?

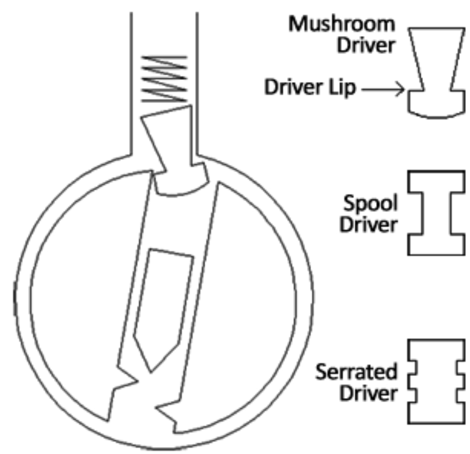
The Devil is in the Details





(a cheap lock)

A better lock:
- no feedback to the attacker about the correctness of the position of each pin
- less room to experiment with movements
- attacker must know exactly the key position of all pins and apply torque in a very specific way (more pins in different positions)



A strawman software vulnerability

```
int i;
unsigned short s;

i = atoi(argv[1]);// parse command line parameter as int

if (i == 0) {     // check
    printf("Invalid number: value must be > 0\n");
    return -1;
}
s = i;
if (s == 0) {     //security check
    printf("Access GRANTED!\n");
}
```

An exploit for the vulnerability

```
$ gcc -o ex1 ex1.c
$ ./ex1 0
Invalid number: value must be > 0
$ ./ex1 10
$ ./ex1 65536  <~ exploit = the number "65536"
Access GRANTED!
```

Exploit vs. Vulnerability

```
$ gcc -o ex1 ex1.c
$ ./ex1 0
Invalid number: value must be > 0
$ ./ex1 10
$ ./ex1 65536  <~ exploit = the number "65536"
Access GRANTED!
```

Vulnerability:

- we check input on **`int i`** with **`if (i == 0)`**
- **`int i`** is guaranteed to be encoded in at least 32 bit (standard C)
- but **`unsigned short s`** can be encoded in 16 bits only
- then we (implicitly) convert an **`int`** to an **`unsigned short`**
- and do our "authentication check" on **`s`**

- **TODO:** can you find a **different** exploit for the **same** vulnerability?

Security as an Engineering Problem

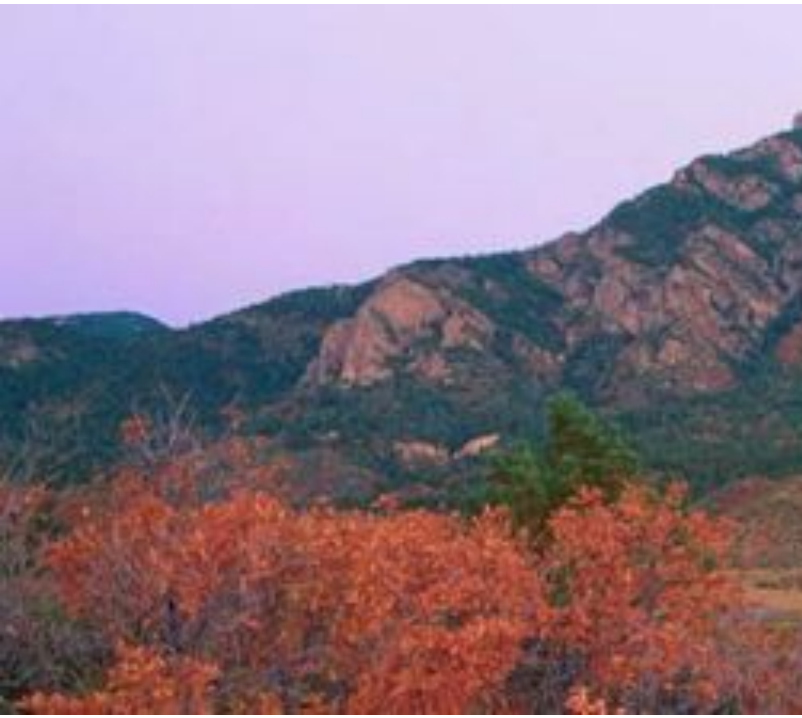
We need some concepts to "solve" it:

- Vulnerabilities
- Exploits
- **Assets**
- **Threats**
- Risks

Security Level =/=> Protection Level

Is this Secure? It Seems Safe...

The Devil is in the Details



"The Cheyenne Mountain nuclear bunker is a Cold War hardened installation with **North American Aerospace Defense Command (NORAD) centers** and associated computer systems [...]"
http://en.wikipedia.org/wiki/Cheyenne_Mountain_nuclear_bunker

Assets and Threats

Asset: identifies what is valuable for an organization.

In this course, we focus on IT assets.

- Examples:
  - hardware (e.g., laptops, computers, phones)
  - software (e.g., applications, operating system, db)
  - data (e.g., data stored in a db)
  - reputation (think about social media)

Threat: Potential violation of CIA; circumstance potentially causing a CIA violation

- Examples:
  - Denial of service (e.g., *software* or *hardware* unavailable),
  - identity theft (e.g., unauthorized access to *software/data*),
  - data leak (e.g., unauthorized release of *data*).

Attacks and Threat Agents

Attack: is an ***intentional*** use of one or more exploits with the objective of compromising a system's CIA.

- Examples:
  - attaching a "malicious" PDF file to an email,
  - picking a lock to enter a building.

Threat Agent: whoever/whatever may cause an attack to occur.

- Examples:
  - malicious software or individual attaching a file
  - thief trying to enter a building

Attackers, Hackers, ...

Mass media created false myths and controversies around these and other words.

Hacker: someone with an *advanced understanding* of computers and computer networks, and willingness to learn "everything."

(see https://datatracker.ietf.org/doc/html/rfc1983, http://www.catb.org/jargon/html/H/hacker.html)

Black hats: malicious hackers.

Attacker != hacker

Security Professionals (white hats)

- Identifying vulnerabilities.
- Developing exploits.
- Developing attack-detection methods.
- Develop countermeasures against attacks.
- Engineer security solutions.

Essential parts of the skillset of a security professionals (also known as *"ethical hackers"*).

Security as an Engineering Problem

No system is invulnerable. So, how do we solve this problem?

- Vulnerabilities
- Exploits
- Assets
- Threats
- **Risks**

Security as "Risk Management"

Risk: statistical and economical evaluation of the exposure to damage because of the presence of vulnerabilities and threats.

$$\text{Risk} = \underbrace{\text{Asset} \times \text{Vulnerabilities}}_{\text{controllable variables}} \times \underbrace{\text{Threats}}_{\text{independent variable}}$$

Security: balance the

(reduction of vulnerabilities + damage containment)

vs.

(cost)

Security *vs.* Cost Balance

Direct costs

- Management
- Operational
- Equipment

Indirect costs (more relevant)

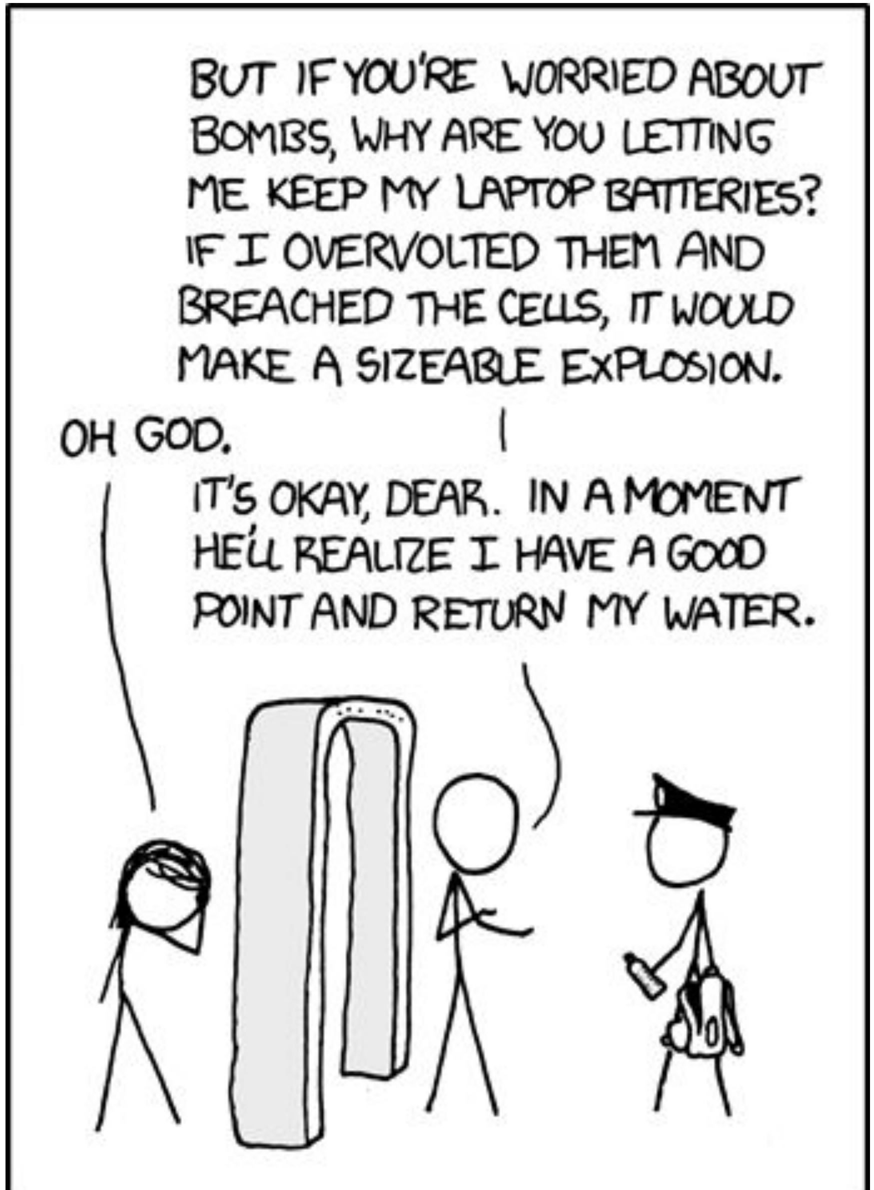
- Less usability
- Slower performance
- Less privacy (due to security controls)
- Reduced productivity (users are slower)

More money ⇏ More security

Throwing more money at the problem will not necessarily solve it

Examples

- Very expensive, unconfigured firewall
  - Better not to have it
- Complex authentication that slows down users
  - Users will write passwords on stickies
- Airport security
  - ...


BUT IF YOU'RE WORRIED ABOUT BOMBS, WHY ARE YOU LETTING ME KEEP MY LAPTOP BATTERIES? IF I OVERVOLTED THEM AND BREACHED THE CELLS, IT WOULD MAKE A SIZEABLE EXPLOSION.
OH GOD.
IT'S OKAY, DEAR. IN A MOMENT HE'LL REALIZE I HAVE A GOOD POINT AND RETURN MY WATER.

Trust and Assumptions

- We must set boundaries.
- Part of the system will be ***assumed*** secure
  - == trusted element.
- Examples:
  - Can we trust the security officer?
  - ...the software we just installed?
  - ...our own code?
  - ...the compiler?
  - ...the BIOS?
  - ...the hardware?
- "chicken and egg" type of problem.

Paper

Ken Thompson, "Reflections on Trusting Trust", in Communications of the ACM (1984), and ACM Turing Award Lectures: The First Twenty Years 1965-1985 (1987)

TL;DR: trojanized compilers.

Bootstrapping again, in a trusted fashion:
https://bootstrappable.org/

Conclusions

Security is a complex *engineering problem* of balancing conflicting requirements.

A system with *limited vulnerabilities* but with a *high threat level* may be *less secure* than a system with *many vulnerabilities* but with *low threat level*.

Attackers, hackers, pirates, ..., are very distinct concepts and should not be confused.

Security is a cost.