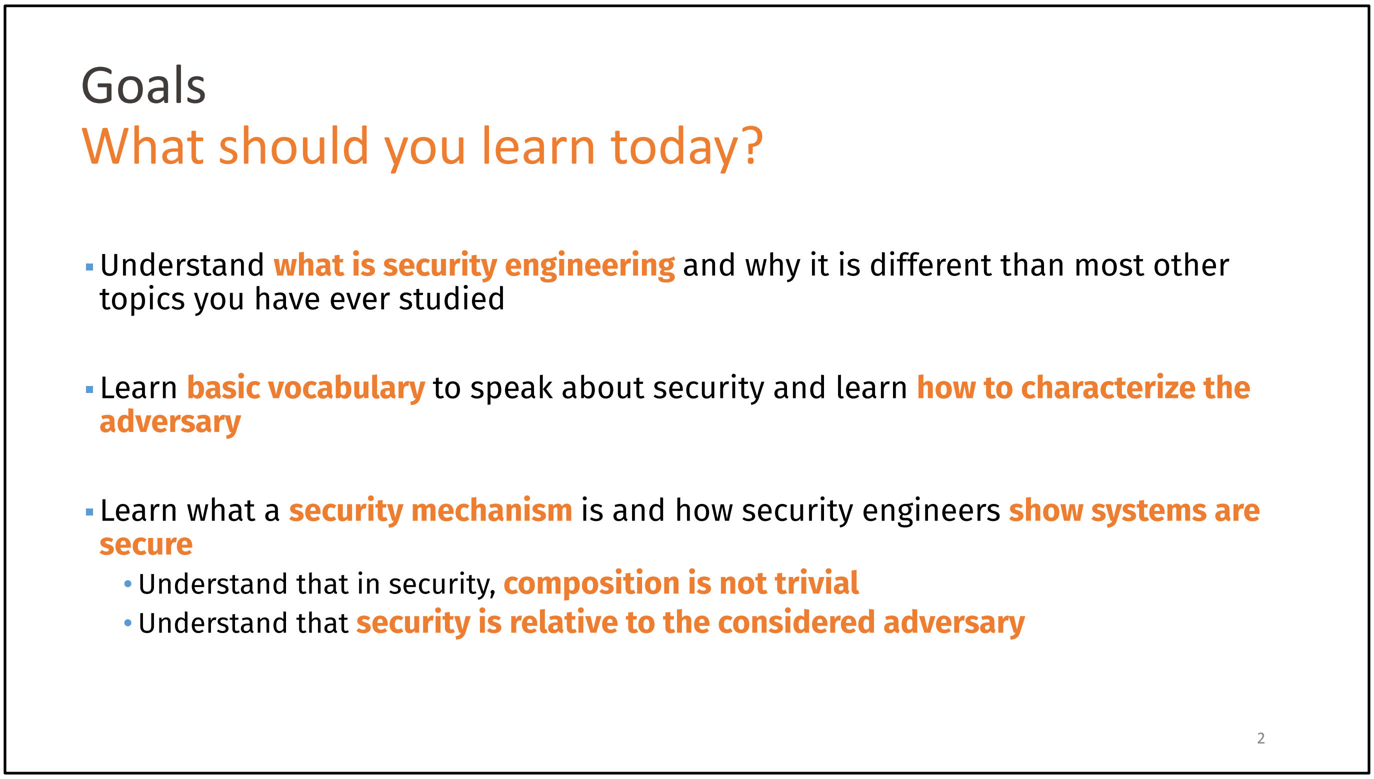
Computer Security and Privacy
What is computer security?
Definitions and basic vocabulary

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

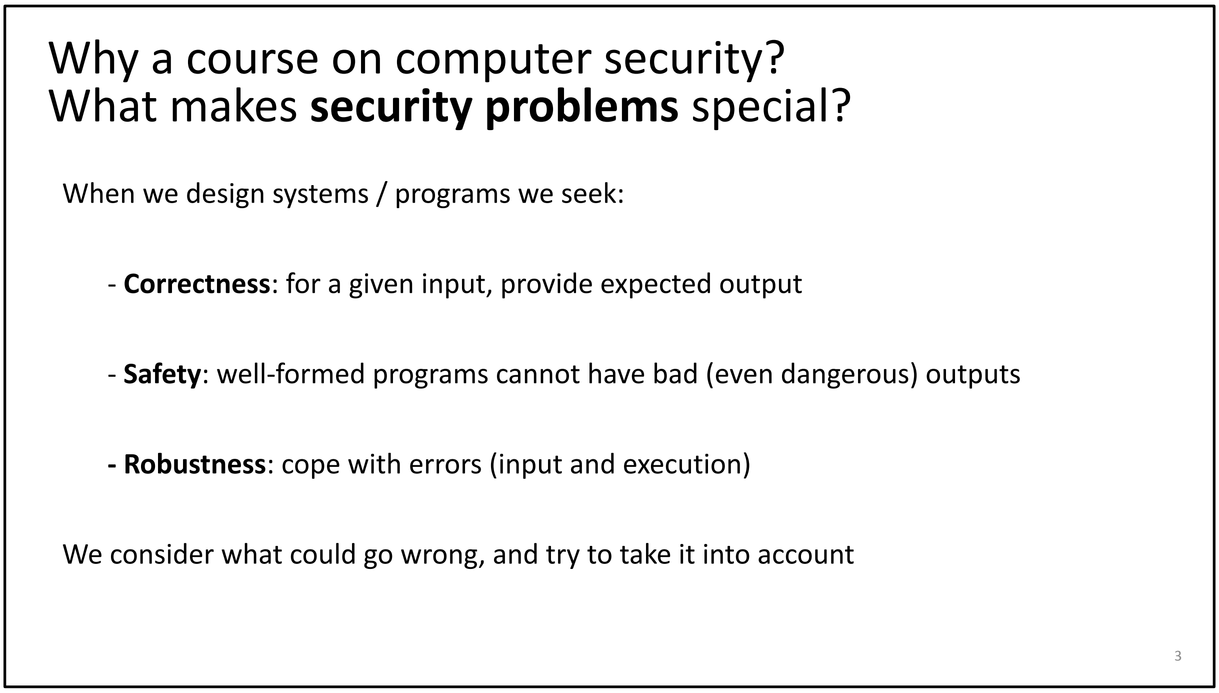
Some slides/ideas adapted from: Philippe Oechslin, George Danezis, Emiliano de Cristofaro, Gianluca Stringhini



Goals
What should you learn today?

- Understand **what is security engineering** and why it is different than most other topics you have ever studied
- Learn **basic vocabulary** to speak about security and learn **how to characterize the adversary**
- Learn what a **security mechanism** is and how security engineers **show systems are secure**
  - Understand that in security, **composition is not trivial**
  - Understand that **security is relative to the considered adversary**

2



Why a course on computer security? What makes **security problems** special?

When we design systems / programs we seek:

- **Correctness**: for a given input, provide expected output
- **Safety**: well-formed programs cannot have bad (even dangerous) outputs
- **Robustness**: cope with errors (input and execution)

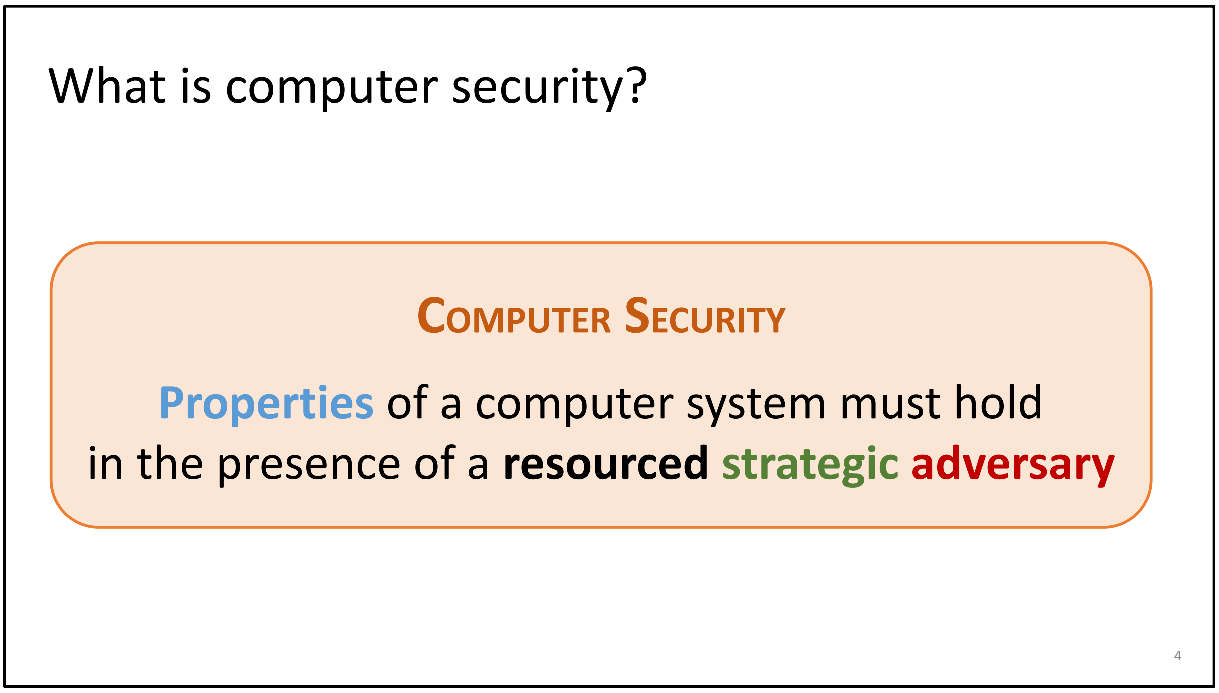
We consider what could go wrong, and try to take it into account

3

When we think about computer programs and how they should work, we usually think:

- They must be correct: they should perform the correct computations. That is, for a given input provide the expected output.
- They must be safe: their outputs should not create dangerous situations. If a program on an elevator fails, the result should not make the elevator fall
- They must be robust: if there is an error, the program should not crash, and still be correct and safe

Typically, these thoughts cover “expected” events: we think about the possible inputs the program will have, and we cover potential issues due to “errors” that happen due to random reasons, e.g., because programmers made a mistake, or an external event (e.g., unexpected change in weather creates an input on a sensor that is outside of the inputs you thought about, or the century changes and dates may be badly interpreted – like with the Y2K error https://en.wikipedia.org/wiki/Year_2000_problem).



What is computer security?

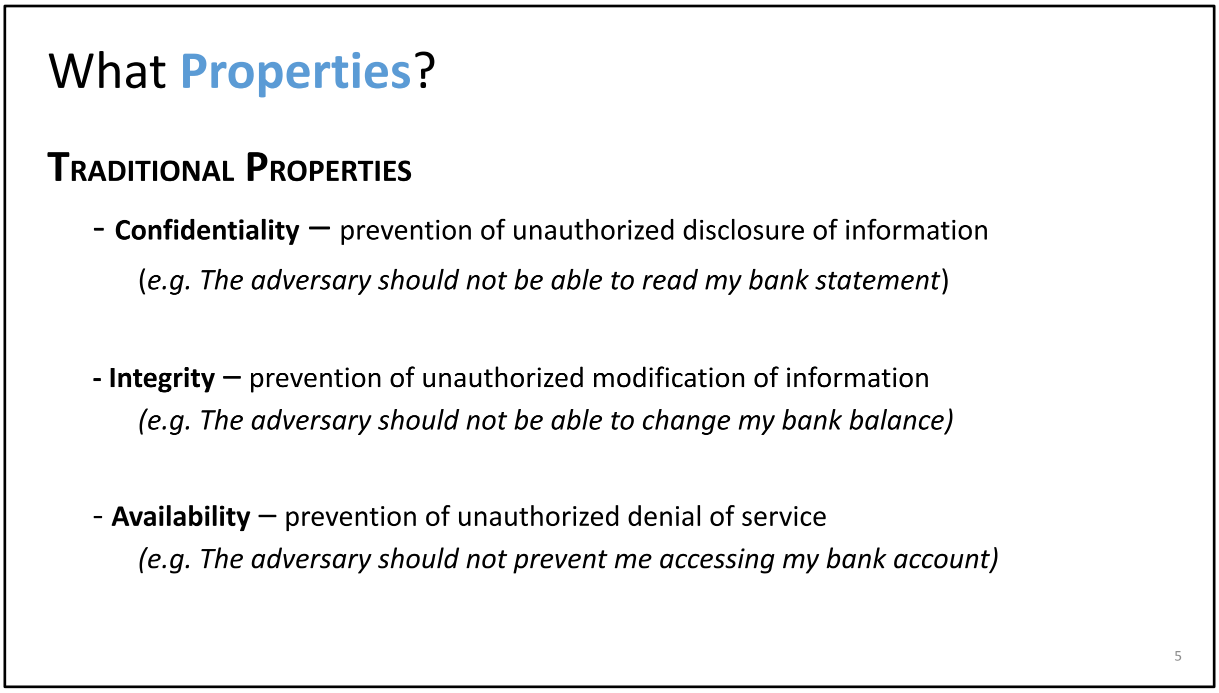
COMPUTER SECURITY

Properties of a computer system must hold
in the presence of a **resourced** **strategic** **adversary**

When we work on security problems however, the events that will bring problems do not happen at random, but are provoked on purpose by this entity we call we call the **adversary**. This adversary will make sure that the worst-case event happens if that would allow them to get value.

Thus, as security engineers we need to think out of the box about what can go wrong and how can it go wrong; and not discard events that have low likelihood of happening, because if it is possible the adversary will do it.

When we think about security we consider that any property **must hold in presence of such an adversary**.



What **Properties**?

Traditional Properties

- **Confidentiality** – prevention of unauthorized disclosure of information
  (*e.g. The adversary should not be able to read my bank statement*)

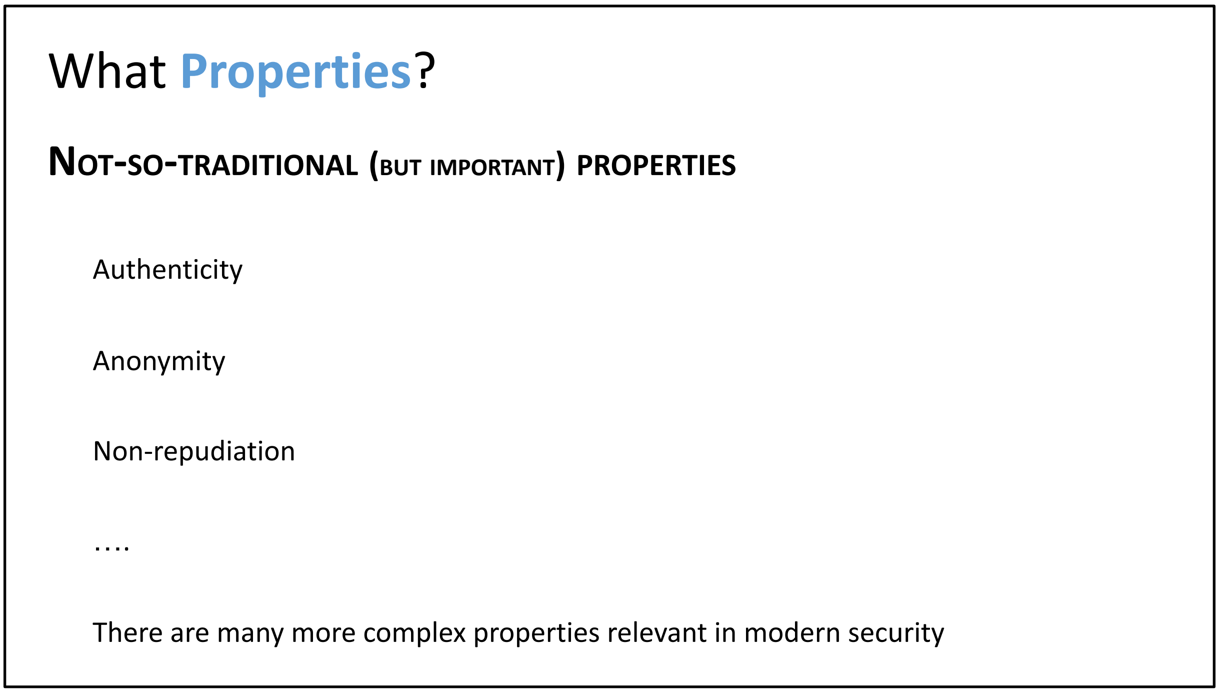
- **Integrity** – prevention of unauthorized modification of information
  (e.g. The adversary should not be able to change my bank balance)

- **Availability** – prevention of unauthorized denial of service
  (e.g. The adversary should not prevent me accessing my bank account)

5

When talking about security, these are the three properties that will appear immediately: **Confidentiality, Integrity, and Availability,** known as CIA.

These are indeed very important properties, but only focusing on them will not lead you to have strong security in modern systems.



What **Properties**?

NOT-SO-TRADITIONAL (BUT IMPORTANT) PROPERTIES

Authenticity

Anonymity

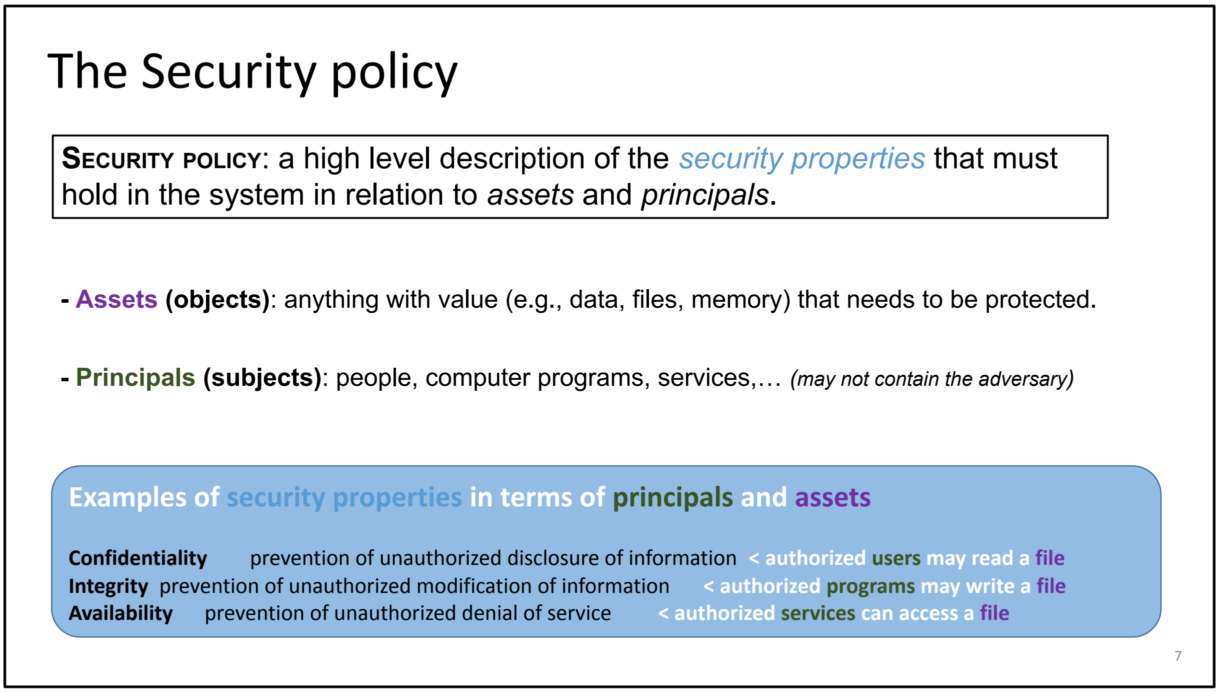
Non-repudiation

….

There are many more complex properties relevant in modern security

In modern systems, we typically need more properties, as will see during the course. We want entities (users, or machines):
- To be who they say they are (*authenticity*), for instance to grant access to a building
- To be able to use digital systems without revealing identity (*anonymity*), for instance to access political content without fear of consequences under oppressive regimes
- To not be able to deny that they have done an action (*non-repudiation*), for instance when paying for goods online

There are more advanced properties, which we will **not** see in this course, that express more complex notions of security. These are hard to express in words, and typically security experts define them in terms of *security games.*



The Security policy

Security policy: a high level description of the *security properties* that must hold in the system in relation to *assets* and *principals*.

- **Assets (objects)**: anything with value (e.g., data, files, memory) that needs to be protected.

- **Principals (subjects)**: people, computer programs, services,… *(may not contain the adversary)*

Examples of security properties in terms of principals and assets

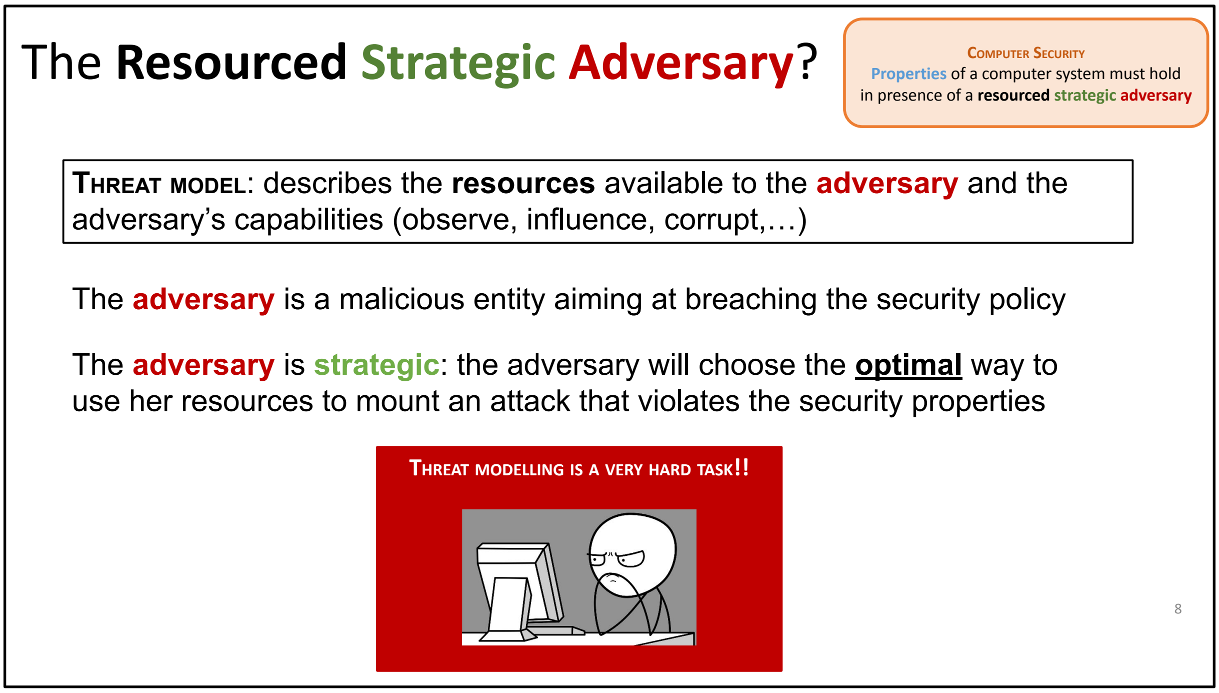
Confidentiality	prevention of unauthorized disclosure of information	< authorized users may read a file
Integrity	prevention of unauthorized modification of information	< authorized programs may write a file
Availability	prevention of unauthorized denial of service	< authorized services can access a file

7

A security policy is a statement that describes what it means for a system to be secure, i.e., what security properties must be met for the system to be considered secure.

This is in relation to
- **what** needs to be protected, which we call **assets**: pieces of data, files, memory parts, etc;
- **who** can perform actions on this assets, which we call **principals**: users, programs, services.

The list of principals may not contain the adversary. This is because the adversary may not be the user of a system, or may not be inside, or may have been overlooked when establishing the security policy.
Bad adversarial modeling is one of the main sources of problems, as we will see in the course.



The **Resourced Strategic Adversary**?

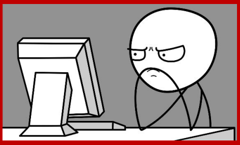
Computer Security
Properties of a computer system must hold in presence of a **resourced strategic adversary**

Threat model: describes the **resources** available to the **adversary** and the adversary's capabilities (observe, influence, corrupt,…)

The **adversary** is a malicious entity aiming at breaching the security policy

The **adversary** is **strategic**: the adversary will choose the **optimal** way to use her resources to mount an attack that violates the security properties

Threat modelling is a very hard task!!



The threat model is a key concept in computer security. It defines the adversary resources and capabilities, and as such describes the context in which the security policy will hold. In other words, security policies are written *within a threat model* and do not provide guarantees out of this context.

Examples of capabilities are observing a given part of the system, influencing parts of the system, or corrupting entities to extract secrets from them or make them act on behalf of the adversary.
Examples of resources are what are the computational capacity of the adversary (e.g., how powerful are her computers), the time the adversary needs to launch the attack (e.g., can be real time or requires a week), how much data the adversary can store (e.g., only some observations, or infinite capacity to store the history of all events in the system.)

When thinking about security, one has to think that the adversary is strategic. This means that the adversary will not try attacks at random, but will always choose the one that works best (most damage, cheaper, in the worse moment, etc.) against the security mechanisms in place

Key vocabulary to talk about the adversary

Threat model

Technical term to define the adversary’s capabilities.

The adversary can observe my connection

The adversary can corrupt my machine

The adversary controls a bank employee

Key vocabulary to talk about the adversary

Threat model

Technical term to define the adversary's capabilities.

The adversary can observe my connection

The adversary can corrupt my machine

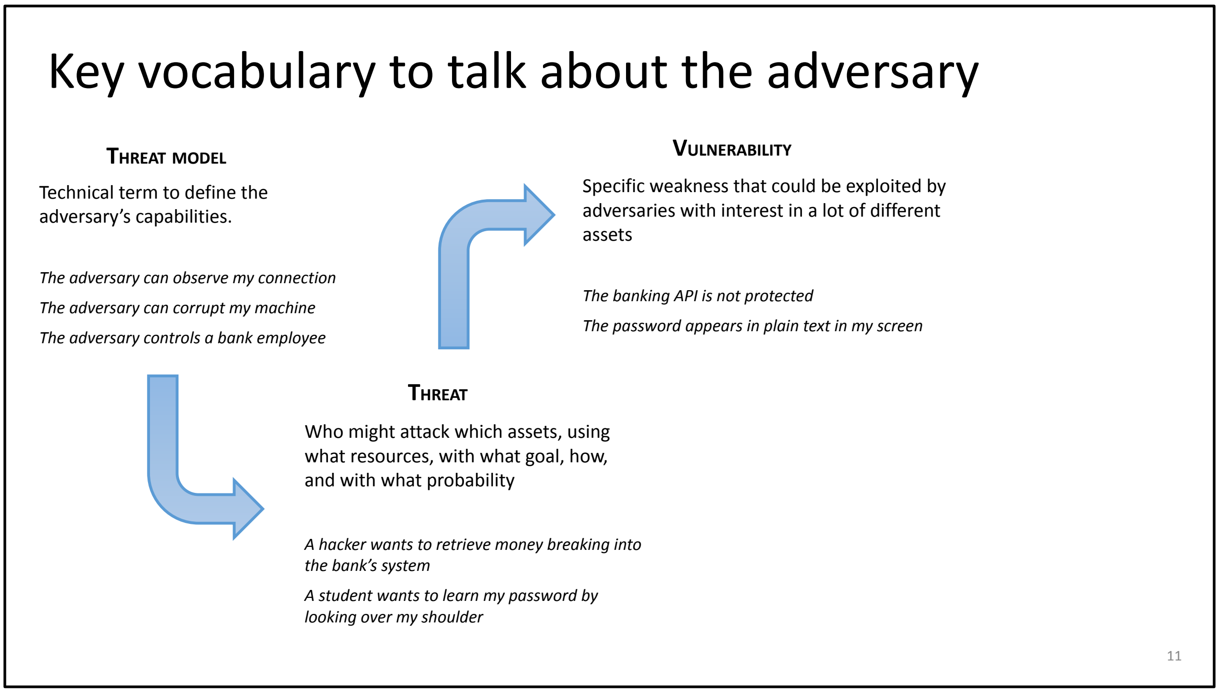
The adversary controls a bank employee

Threat

Who might attack which assets, using what resources, with what goal, how, and with what probability

A hacker wants to retrieve money breaking into the bank's system

A student wants to learn my password by looking over my shoulder



Key vocabulary to talk about the adversary

Threat model

Technical term to define the adversary's capabilities.

The adversary can observe my connection

The adversary can corrupt my machine

The adversary controls a bank employee

Vulnerability

Specific weakness that could be exploited by adversaries with interest in a lot of different assets

The banking API is not protected

The password appears in plain text in my screen

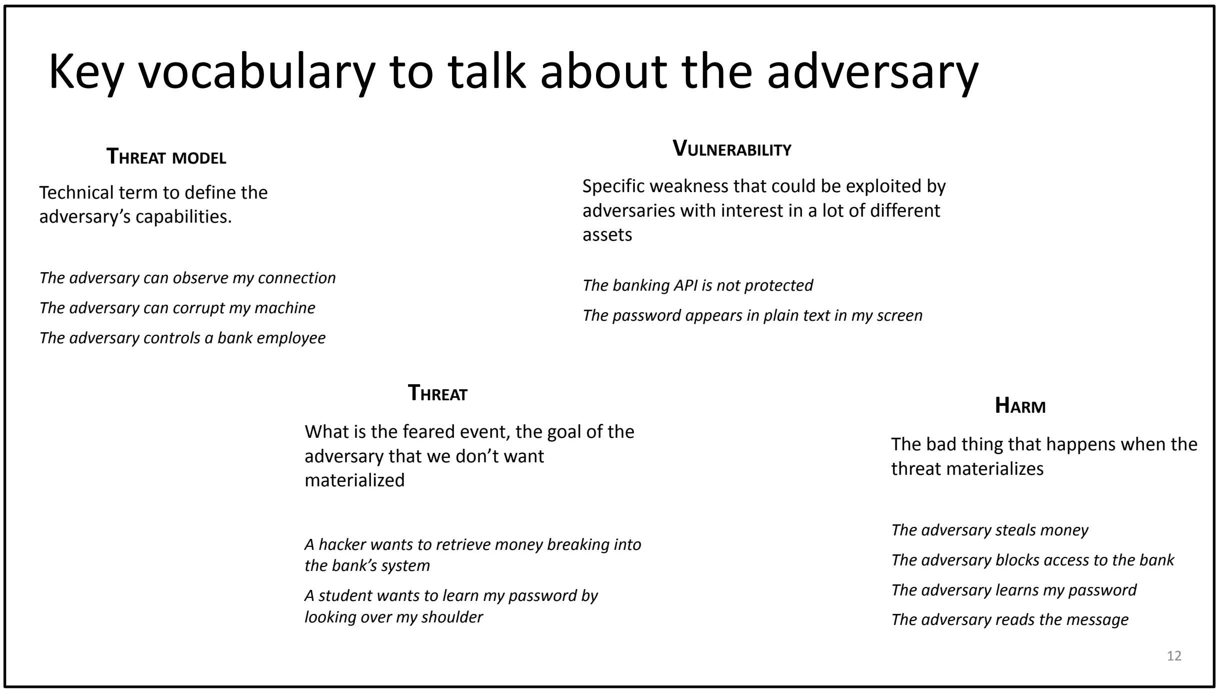
Threat

Who might attack which assets, using what resources, with what goal, how, and with what probability

A hacker wants to retrieve money breaking into the bank's system

A student wants to learn my password by looking over my shoulder

11



Key vocabulary to talk about the adversary

Threat model

Technical term to define the adversary's capabilities.

The adversary can observe my connection

The adversary can corrupt my machine

The adversary controls a bank employee

Vulnerability

Specific weakness that could be exploited by adversaries with interest in a lot of different assets

The banking API is not protected

The password appears in plain text in my screen

Threat

What is the feared event, the goal of the adversary that we don't want materialized

A hacker wants to retrieve money breaking into the bank's system

A student wants to learn my password by looking over my shoulder

Harm

The bad thing that happens when the threat materializes

The adversary steals money

The adversary blocks access to the bank

The adversary learns my password

The adversary reads the message

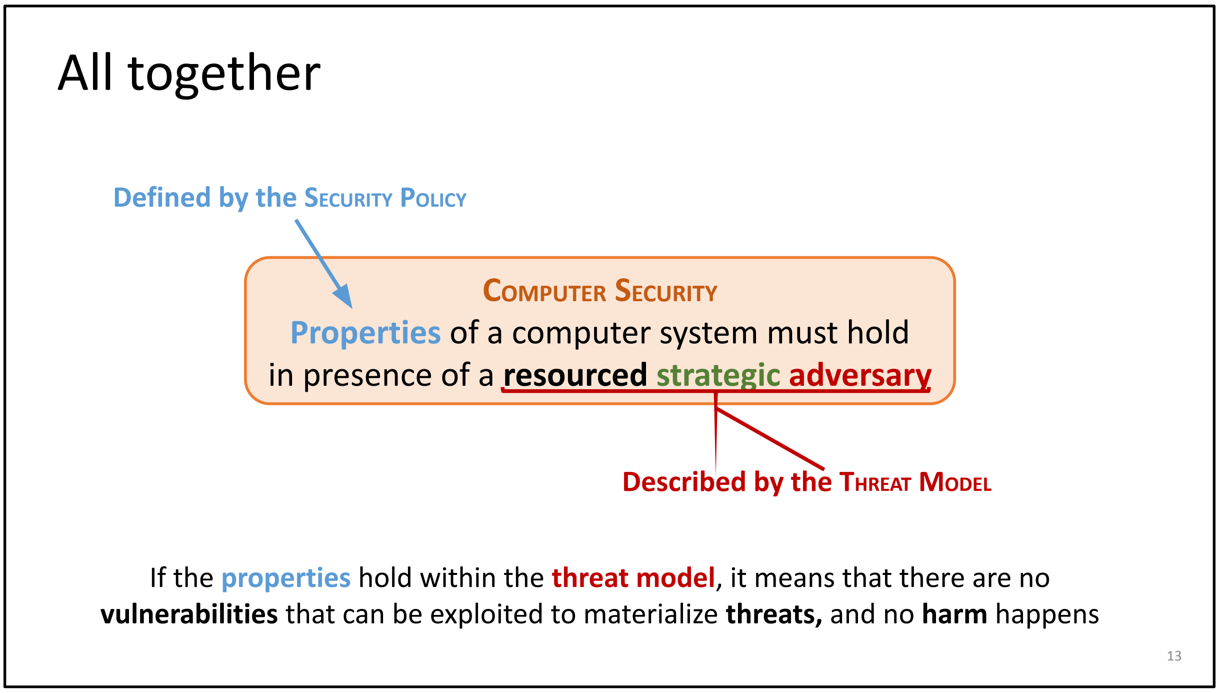
12

A **threat model** is a technical term to define the adversary's capabilities

It usually comes from the **threats** to the system: what are the feared events?

To materialize the threats, the adversary exploits **vulnerabilities**

And when a vulnerability is exploited, it creates a **harm**



All together

Defined by the Security Policy
Computer Security
Properties of a computer system must hold in presence of a resourced strategic adversary
Described by the Threat Model

If the **properties** hold within the **threat model**, it means that there are no **vulnerabilities** that can be exploited to materialize **threats,** and no **harm** happens

13

In summary:

Computer security deals with ensuring that certain **security properties** in a computer system are achieved even if a **strategic adversary** attacks the system.

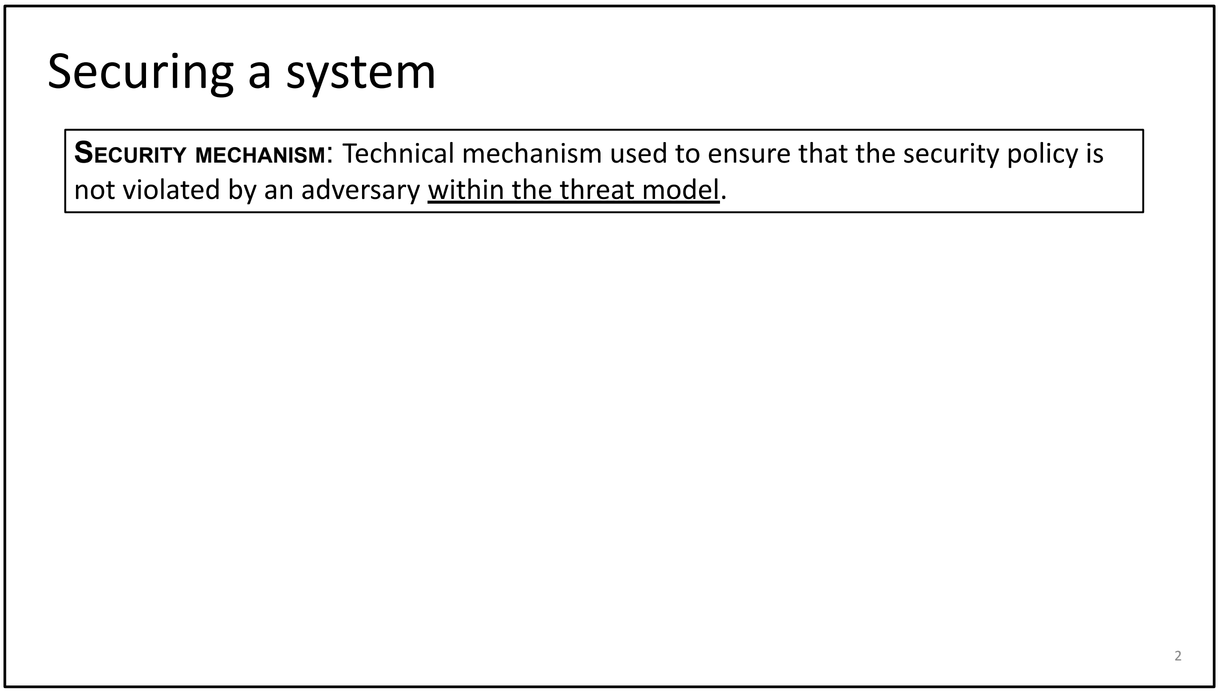
This adversary's capabilities are defined by the **threat model**. A good threat model assumes this adversary is **strategic,** *i.e.,* they know the system and they will always take the most damaging approach to breach security.

If the systems is secure, i.e., the properties hold, that means that none of the **vulnerabilities** of the system can be exploited by an adversary. In other words, no threat will become real and therefore there can be no **harms**.

Computer Security and Privacy
Basic concepts of security engineering

Carmela Troncoso
SPRING Lab
carmela.troncoso@epfl.ch

Some slides/ideas adapted from: Philippe Oechslin, George Danezis, Emiliano de Cristofaro, Gianluca Stringhini

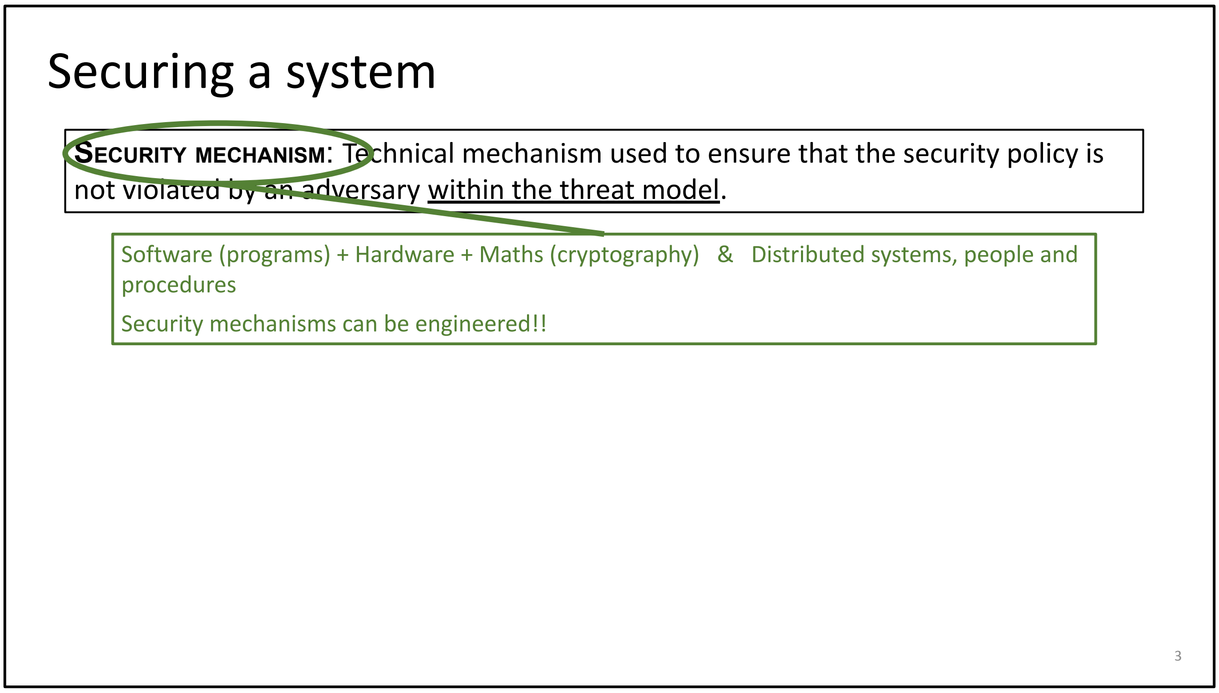


Securing a system

SECURITY MECHANISM: Technical mechanism used to ensure that the security policy is not violated by an adversary <u>within the threat model</u>.

2

We secure a systems using **security mechanisms**. These can be software / hardware / require maths (cryptography), or can be architectural (distribute your data across different systems), rely on people, or organizational procedures.



Securing a system

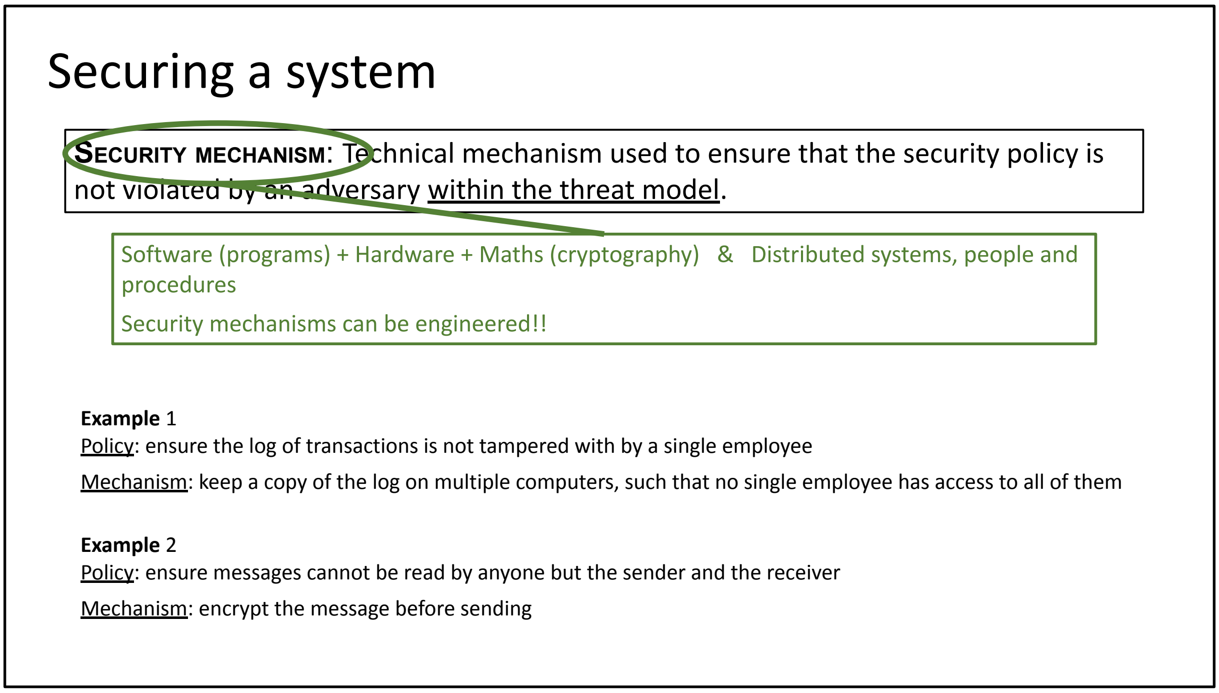
Security mechanism: Technical mechanism used to ensure that the security policy is not violated by an adversary within the threat model.

Software (programs) + Hardware + Maths (cryptography) & Distributed systems, people and procedures

Security mechanisms can be engineered!!

3

We secure a systems using **security mechanisms**. These can be software / hardware / require maths (cryptography), or can be architectural (distribute your data across different systems), rely on people, or organizational procedures.



Securing a system

Security mechanism: Technical mechanism used to ensure that the security policy is not violated by an adversary within the threat model.

Software (programs) + Hardware + Maths (cryptography) & Distributed systems, people and procedures

Security mechanisms can be engineered!!

Example 1
Policy: ensure the log of transactions is not tampered with by a single employee
Mechanism: keep a copy of the log on multiple computers, such that no single employee has access to all of them

Example 2
Policy: ensure messages cannot be read by anyone but the sender and the receiver
Mechanism: encrypt the message before sending

We secure a systems using **security mechanisms**. These mechanisms are tools that implement the security policy

Security can be software (e.g., an antivirus) / hardware (e.g., a smart card) / require maths (cryptography), or can be architectural (distribute your data across different systems), rely on people, or organizational procedures.

Systems are big! Need security mechanism**S**

Security **does not necessarily increase linearly** with the number of mechanisms!

Two ways of composing

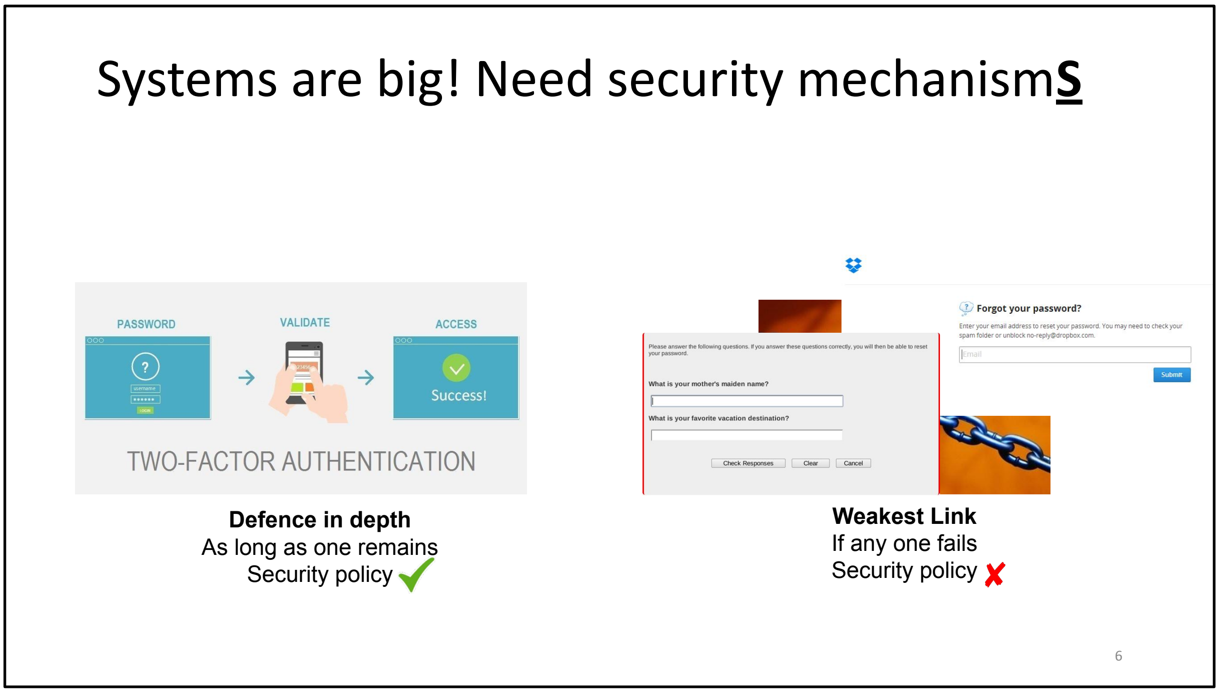
Defence in depth
As long as one remains
Security policy ✓

Weakest Link
If any one fails
Security policy ✗

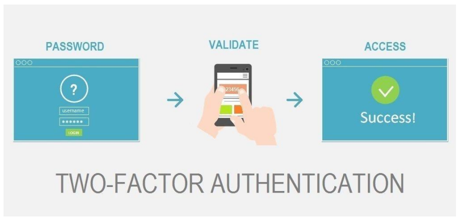
We have seen principles to design a security mechanism, but systems usually require more than one.

But composing mechanisms is far from simple, and not necessarily linear! Not because we have two security mechanisms now the system is more secure.

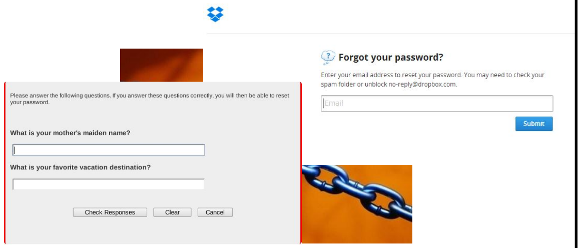
Two composition flavours:
- Defense in depth: As long as one mechanism is not broken, the security policy holds. The strongest mechanism defines the strength of the system.
- Weakest link: all mechanisms need to work properly for the security policy to holds. The weakest mechanism defines the strength of the system.



Systems are big! Need security mechanism**S**



Defence in depth
As long as one remains
Security policy ✓



Forgot your password?

Enter your email address to reset your password. You may need to check your spam folder or unblock no-reply@dropbox.com.

Please answer the following questions. If you answer these questions correctly, you will then be able to reset your password.

What is your mother's maiden name?

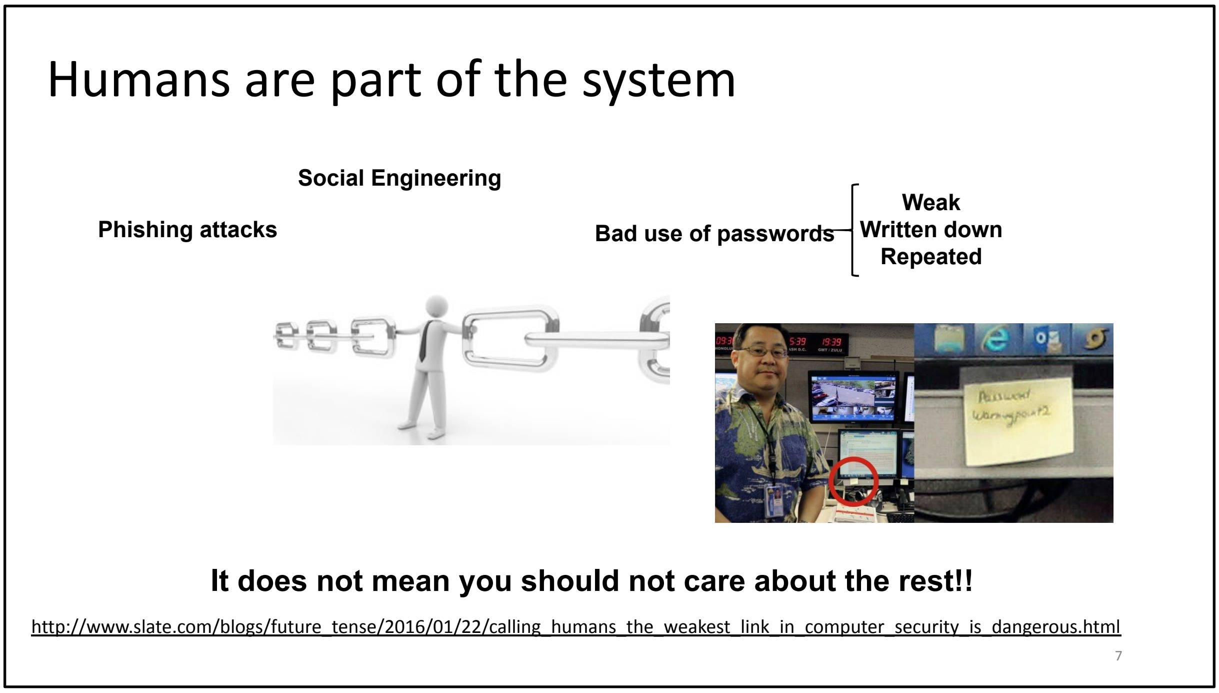
What is your favorite vacation destination?

Weakest Link
If any one fails
Security policy ✗

6

In two factor authentication, as long as one factor remains secure, the system resists.

When we have password recovery via security question, if either the password or the answer to the question is available to the adversary, the system is broken.



Humans are part of the system

Social Engineering

Phishing attacks

Bad use of passwords — Weak / Written down / Repeated

It does not mean you should not care about the rest!!

http://www.slate.com/blogs/future_tense/2016/01/22/calling_humans_the_weakest_link_in_computer_security_is_dangerous.html

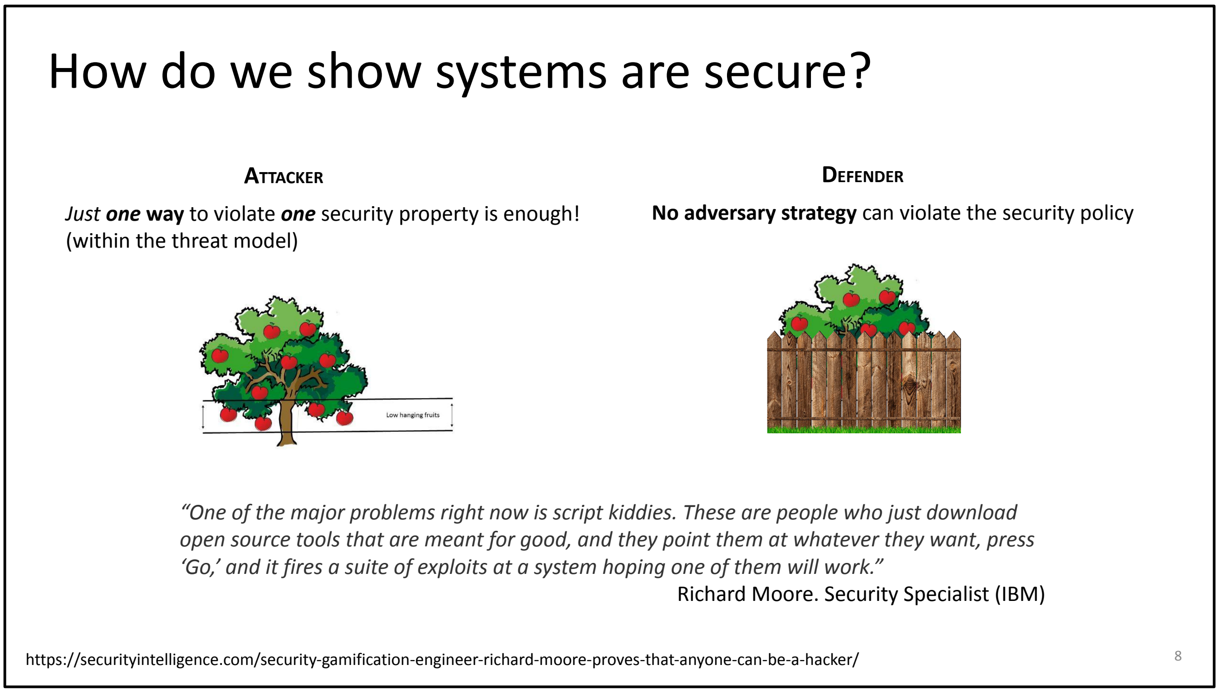
7

Security Policies may also be breached without breaking the security mechanisms.

Users are principals in the system, and as such they are potential targets for attacks in order to get information to break into the system.

Typical attacks can be:
- Phishing, in which users get tricked into inputting their data on a fake system that looks real
- Social engineering, in which a human tricks the user into telling their secrets
- Abuse bad use of passwords (we will see more in the Authentication lecture)

More about the photo:
https://www.businessinsider.com/hawaii-emergency-agency-password-discovered-in-photo-sparks-security-criticism-2018-1?r=US&IR=T
When you write your passwords down, they can be read!


Low hanging fruits

How do we show systems are secure?

ATTACKER

*Just **one*** **way** to violate ***one*** security property is enough!
(within the threat model)

DEFENDER

No adversary strategy can violate the security policy

“One of the major problems right now is script kiddies. These are people who just download open source tools that are meant for good, and they point them at whatever they want, press ‘Go,’ and it fires a suite of exploits at a system hoping one of them will work.”
Richard Moore. Security Specialist (IBM)

https://securityintelligence.com/security-gamification-engineer-richard-moore-proves-that-anyone-can-be-a-hacker/

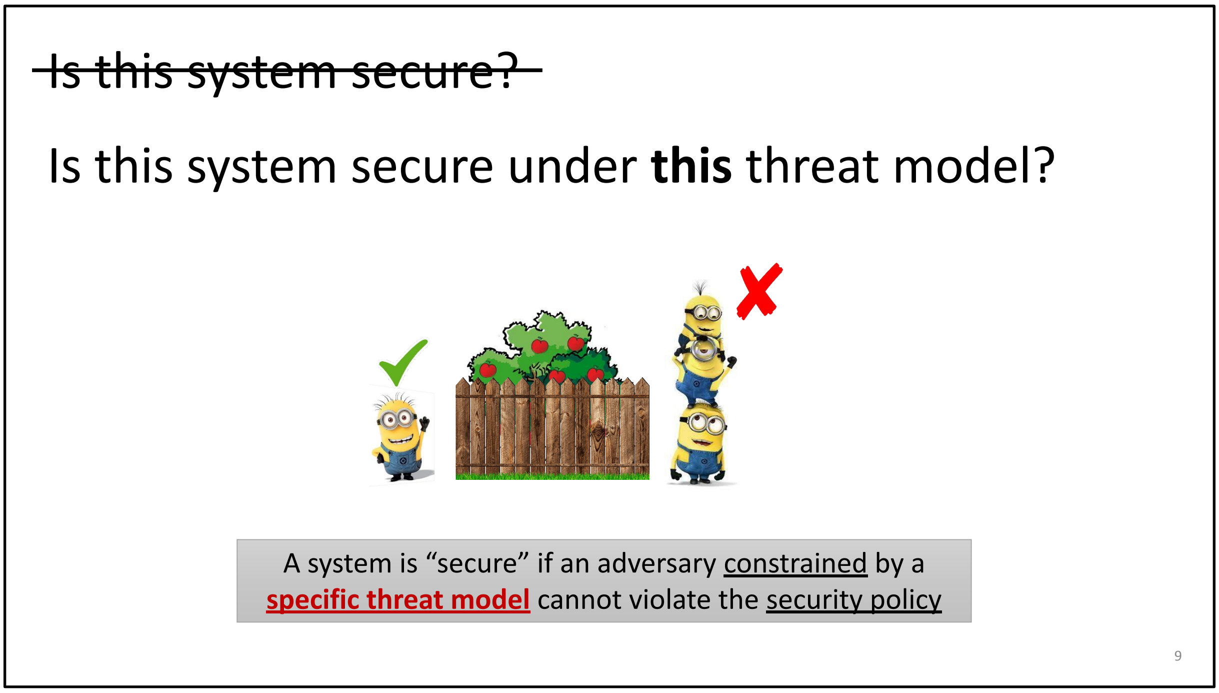
8

The defender always has a much harder task than the adversary.

Adversaries just need to find one way to break the security policy. And as they are **strategic**, they will find the simplest way to do that with their resources.

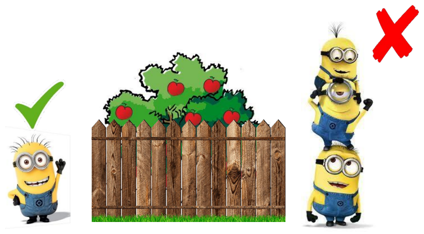
The defender, on the other hand, needs to find a strategy to defend **against any adversary strategy**. This obviously is much harder.

Even worse, as the UNIL example highlights, there is no need for very savvy adversaries. The only requirement is to find the tool that allows you to attack the system.



~~Is this system secure?~~

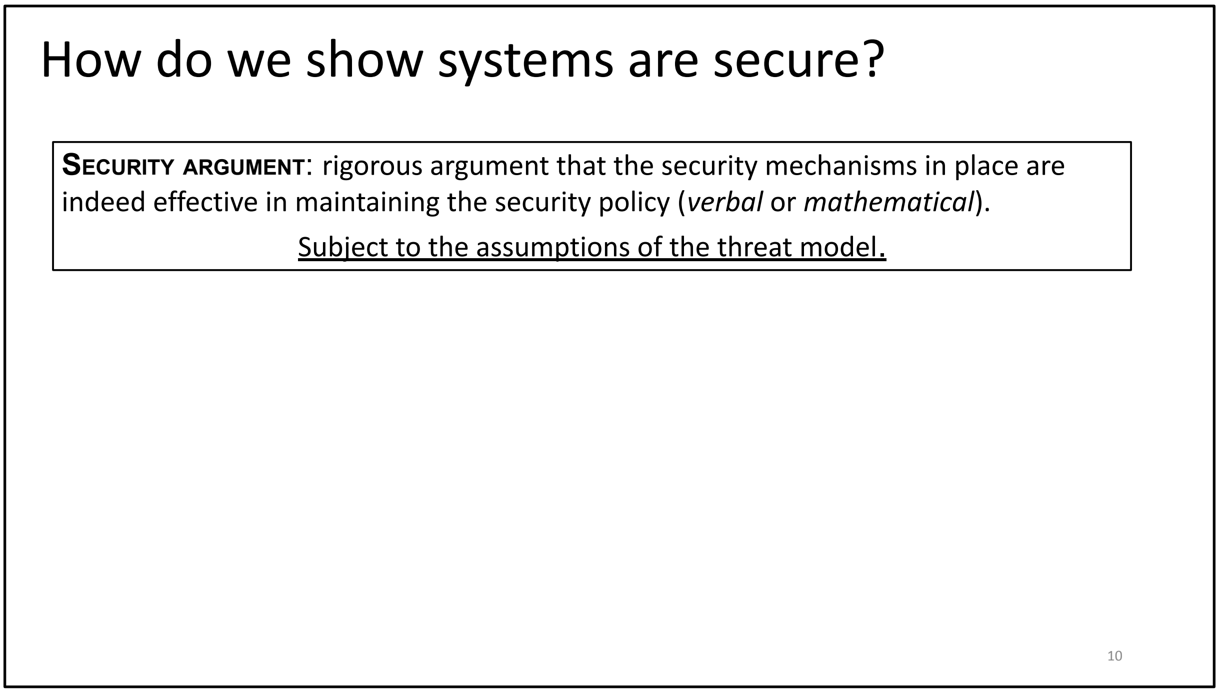
Is this system secure under **this** threat model?



A system is "secure" if an adversary constrained by a specific threat model cannot violate the security policy

One cannot say "a system is secure". This sentence is **void** of content.

We can only say "a system is secure under a specific threat model" (that is, an adversary **within** the threat model cannot violate the security policy)



How do we show systems are secure?

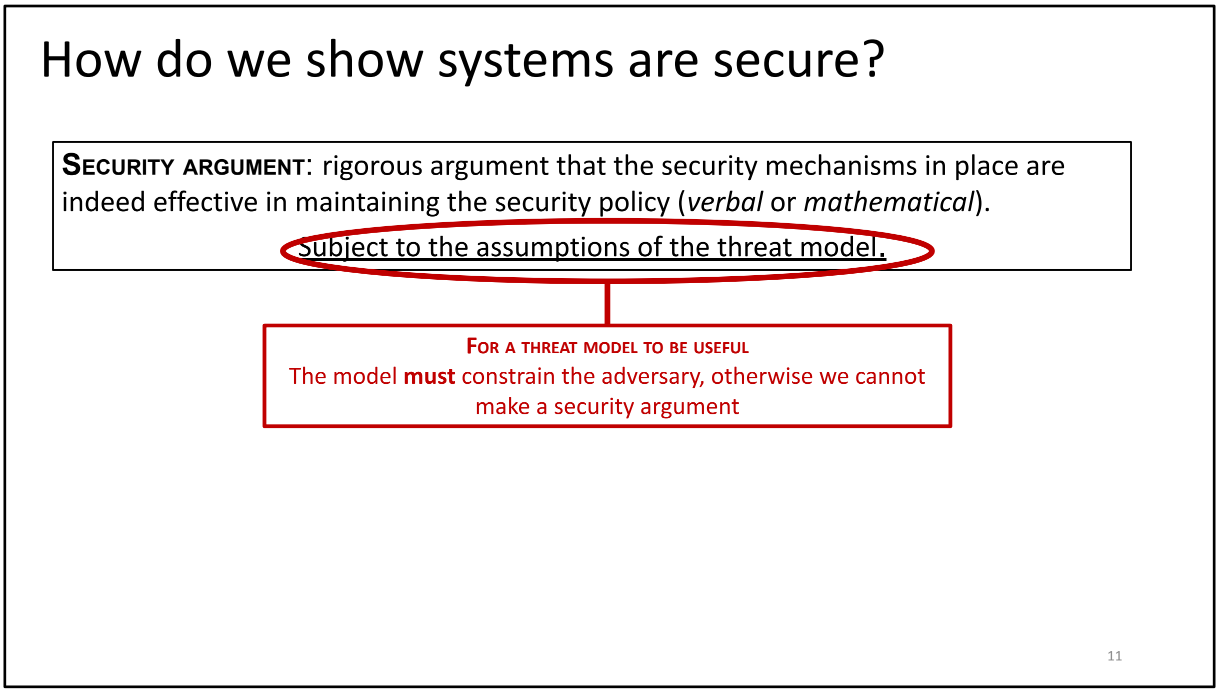
SECURITY ARGUMENT: rigorous argument that the security mechanisms in place are indeed effective in maintaining the security policy (*verbal* or *mathematical*).

<u>Subject to the assumptions of the threat model.</u>

Once mechanisms are in place, to say the system is secure if we can make a **security argument** that says that, under the considered threat model, no adversary can break the security policy. This argument can be verbal, but modern systems rely on mathematical arguments and proofs that are very complex to construct and understand (continue your studies if you want to know more about this).

To show that a composition is secure the security engineer must provide a **security argument**

In either case, the only way to show that a composition of mechanisms is secure is to provide a security argument that the composition is fulfils the security policy under a given threat model.



How do we show systems are secure?

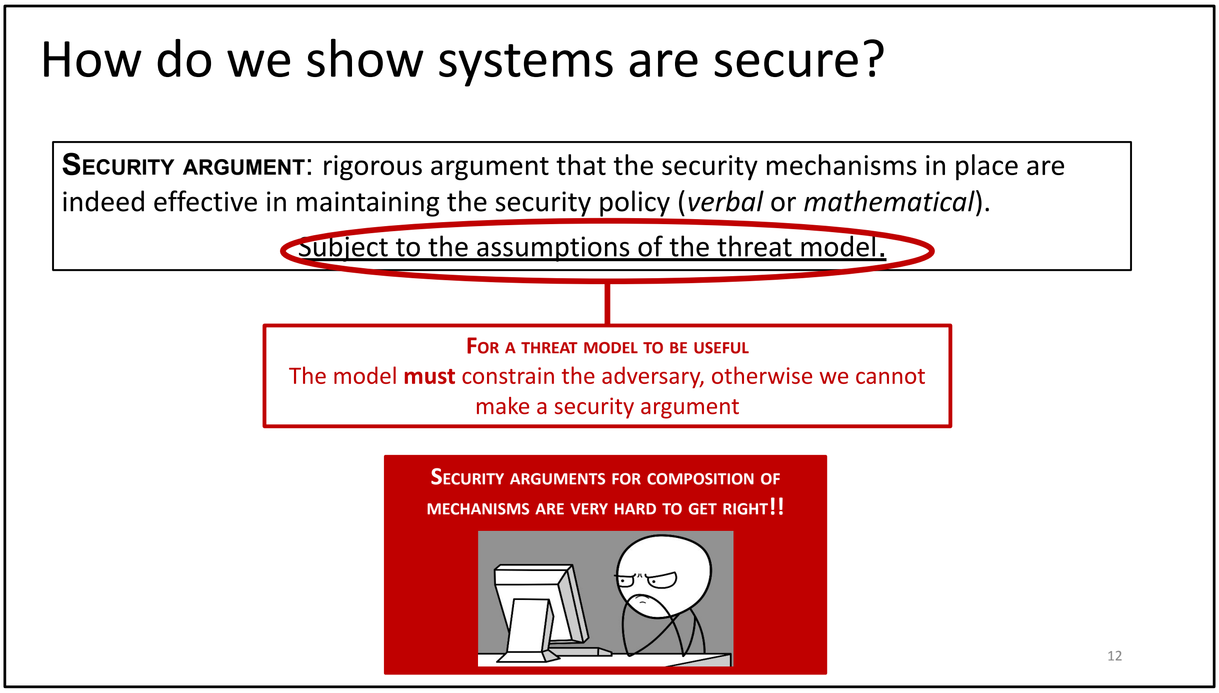
Security argument: rigorous argument that the security mechanisms in place are indeed effective in maintaining the security policy (*verbal* or *mathematical*).
Subject to the assumptions of the threat model.

For a threat model to be useful
The model **must** constrain the adversary, otherwise we cannot make a security argument

Once mechanisms are in place, to say the system is secure if we can make a **security argument** that says that, under the considered threat model, no adversary can break the security policy. This argument can be verbal, but modern systems rely on mathematical arguments and proofs that are very complex to construct and understand (continue your studies if you want to know more about this).

To show that a composition is secure the security engineer must provide a **security argument**

In either case, the only way to show that a composition of mechanisms is secure is to provide a security argument that the composition is fulfils the security policy under a given threat model.



How do we show systems are secure?

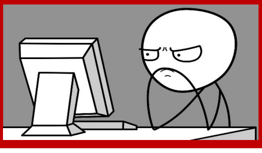
Security argument: rigorous argument that the security mechanisms in place are indeed effective in maintaining the security policy (*verbal* or *mathematical*).

Subject to the assumptions of the threat model.

For a threat model to be useful

The model **must** constrain the adversary, otherwise we cannot make a security argument

Security arguments for composition of mechanisms are very hard to get right!!

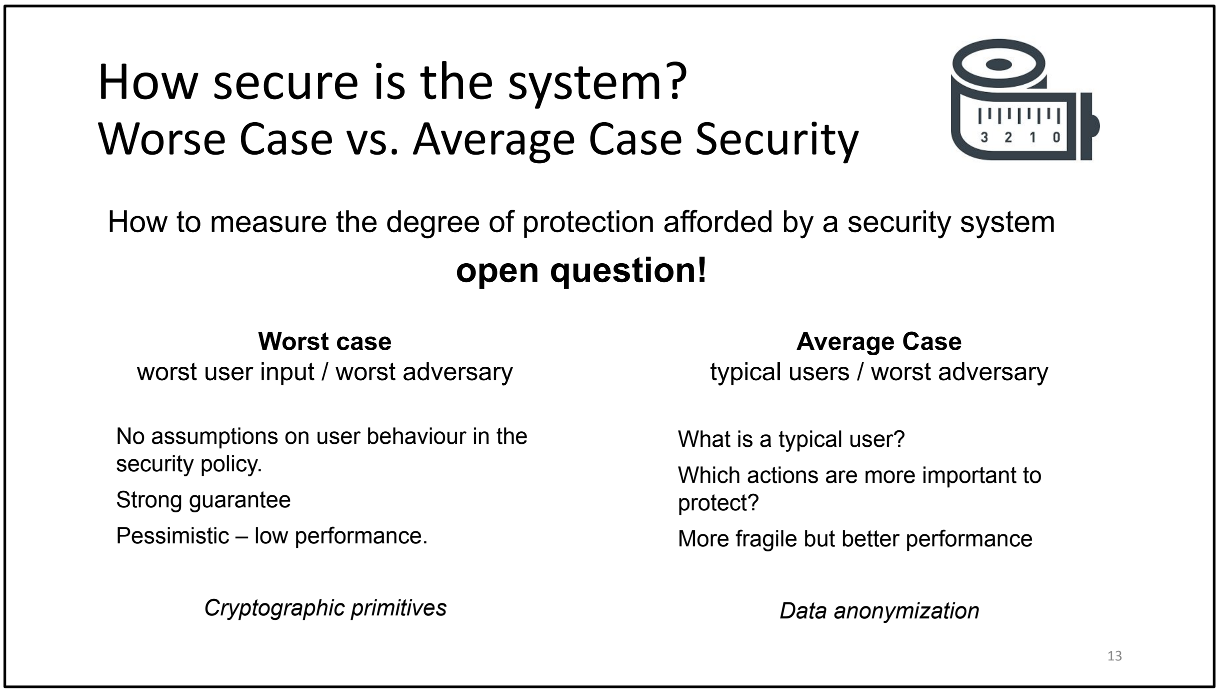


12

Once mechanisms are in place, to say the system is secure if we can make a **security argument** that says that, under the considered threat model, no adversary can break the security policy. This argument can be verbal, but modern systems rely on mathematical arguments and proofs that are very complex to construct and understand (continue your studies if you want to know more about this).

To show that a composition is secure the security engineer must provide a **security argument**

In either case, the only way to show that a composition of mechanisms is secure is to provide a security argument that the composition is fulfils the security policy under a given threat model.



How secure is the system?
Worse Case vs. Average Case Security

How to measure the degree of protection afforded by a security system

open question!

Worst case
worst user input / worst adversary

No assumptions on user behaviour in the security policy.

Strong guarantee

Pessimistic – low performance.

Cryptographic primitives

Average Case
typical users / worst adversary

What is a typical user?

Which actions are more important to protect?

More fragile but better performance

Data anonymization

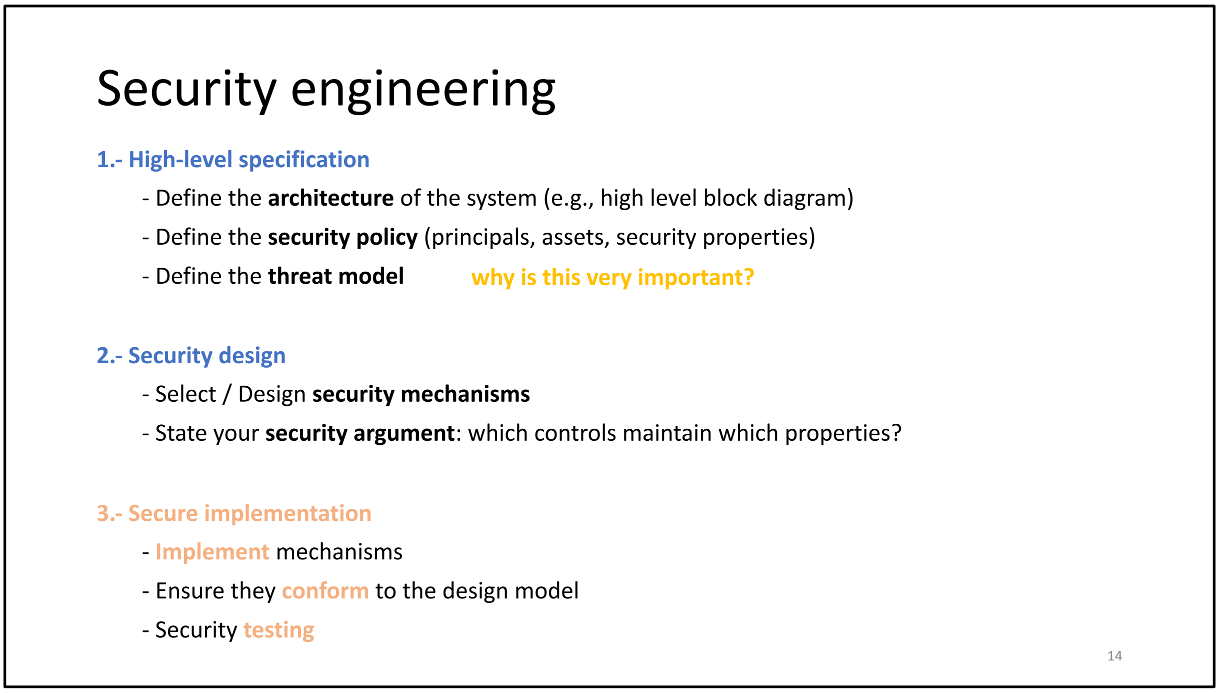
13

In this security argument, you must argue how secure the system is. There is no standard metric for this.

There are two ways of reasoning:

- Your system is secure in the worst case: even if the most powerful adversary in your threat model has access to the worst case input, regardless of how users behave, the security properties hold. Getting such level of protection is great, but can have significant impact on the performance of the system. Examples of significant impact can be: requiring expensive computations (cryptography) or requiring heavy modification of data to the point that it is not useful .

- Your system is secure in the average case: in this type of security arguments one needs to define the user behaviour and demonstrate that for that particular case the security properties hold most of the time. In other words, this is a risk-based approach in which one gets less protection but much better utility.



Security engineering

1.- High-level specification

- Define the **architecture** of the system (e.g., high level block diagram)
- Define the **security policy** (principals, assets, security properties)
- Define the **threat model** **why is this very important?**

2.- Security design

- Select / Design **security mechanisms**
- State your **security argument**: which controls maintain which properties?

3.- Secure implementation

- Implement mechanisms
- Ensure they conform to the design model
- Security testing

Defining the threat model is very important, because otherwise one cannot really make a security argument!

Summary

Security problems always involve an **adversary**

- The adversary is **strategic**, will take the most damaging approach
- The adversary's capabilities define a **threat model**
- **Security mechanisms** aim at fulfilling a **security policy <u>within</u> a threat model**
- Showing security implies providing a **security argument**