Lec 02: The Basics

CSED415: Computer Security

Spring 2024

Seulbae Kim

POSTECH

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

New students

- Greetings! Please review lec 1 slides and post questions on PLMS

- Upcoming schedule reminders
  - Mon, Feb 26: Lab01 will be released
  - Mon, Mar 4 – Sun, Mar 10: Project team forming
    - Get to know each other
    - Form teams consisting of two members
      - Except for one team, which will have three members (2 + 2 + 2 + 2 + 3 = 11)
    - Choose a team name and designate a team leader
    - Team leaders should submit team information on PLMS
      - Refer to the assignment page for details

Recap

POSTECH

- What is “computer security”?
  - Protecting our computer-related assets

NIST* definition

- Computer security
  - The protection afforded to an automated information system in order to preserve the **confidentiality, integrity**, and **availability** of information system resources, which include hardware, software, firmware, information, data, and telecommunications.

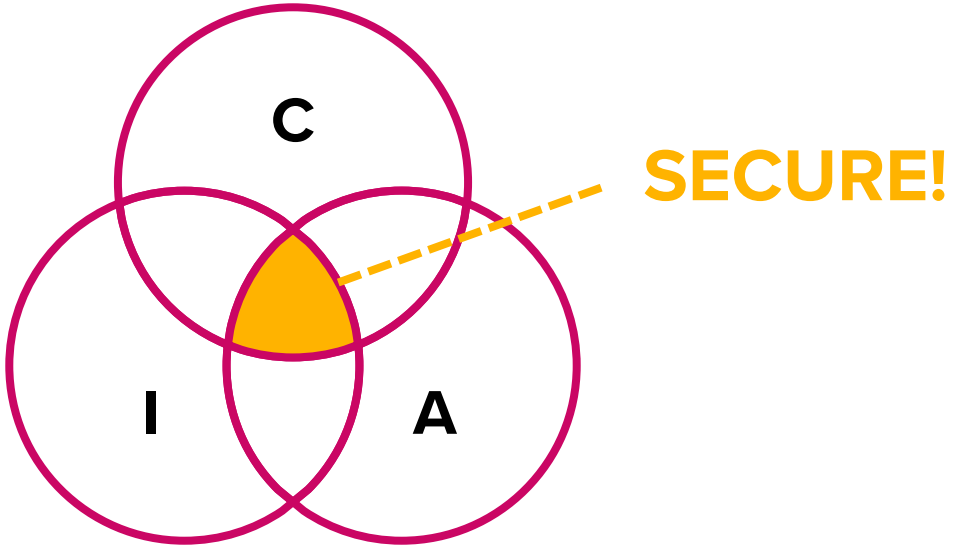
*NIST: National Institute of Standards and Technology

Key objective: preserving CIA

CIA overview

- Secure systems satisfy the “CIA triad”
  - Confidentiality: Information* is not available to unauthorized parties
  - Integrity: Information is not modified in an unauthorized manner
  - Availability: Information is readily available when it is needed

* Data, resource, …

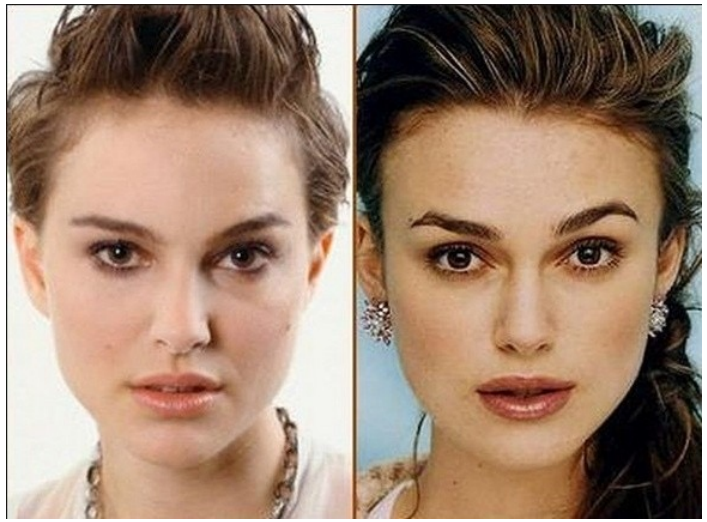


Confidentiality

- The ability to limit access of information to unauthorized entities
  - My emails, my bank accout balance, your grades, …
- Difficult to ensure confidentiality in practice due to..
  - Delegation, revocation, change of roles, having conflicting roles, …

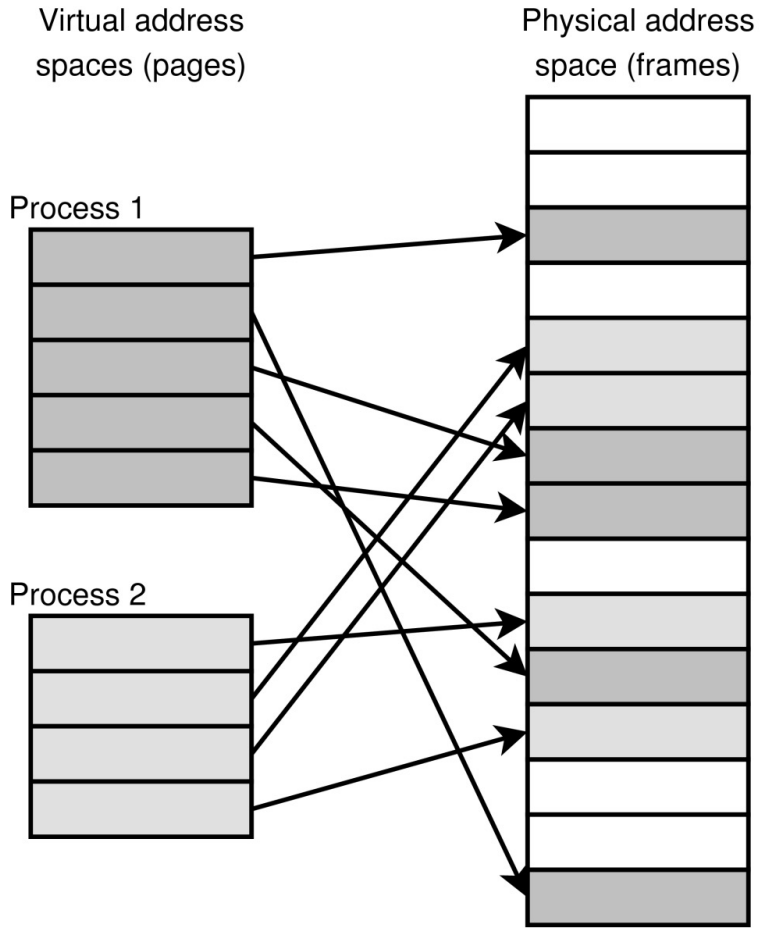
Ways to enforce confidentiality

- Identify sensitive information
- Access control: Specify “who” can access the sensitive info
- Authentication: Verify the identity
  - “Is the person claiming to be Natalie truly Natalie?”



Examples of confidentiality

- POSTECH's physical access control (student id / professor id)
- OS: process isolation and
  virtual address
  - Q) What happens if a process tries to read what's beyond its address space?
- What else?



Integrity

- The ability to prevent unauthorized modification of information
  - Note: integrity is not about disallowing ANY change!
  - Change should happen in an authorized manner
    - e.g., Your account balance should remain unchanged unless you make a transaction. When a transaction occurs, the balance should be adjusted precisely by the amount spent.

Types of integrity

- Data integrity
  - Information is changed only in a specified and authorized manner
- System integrity
  - A system performs its intended function in an unimpaired manner
    - e.g., add(0.1, 0.2) should return 0.3

Ways to assure integrity

- Access control: Specify “who” can modify “which information”
- Authentication: Verify who (same as confidentiality)
- Redundancy: Creating multiple copies and cross-checking
- Data validity checks
  - e.g., checksum

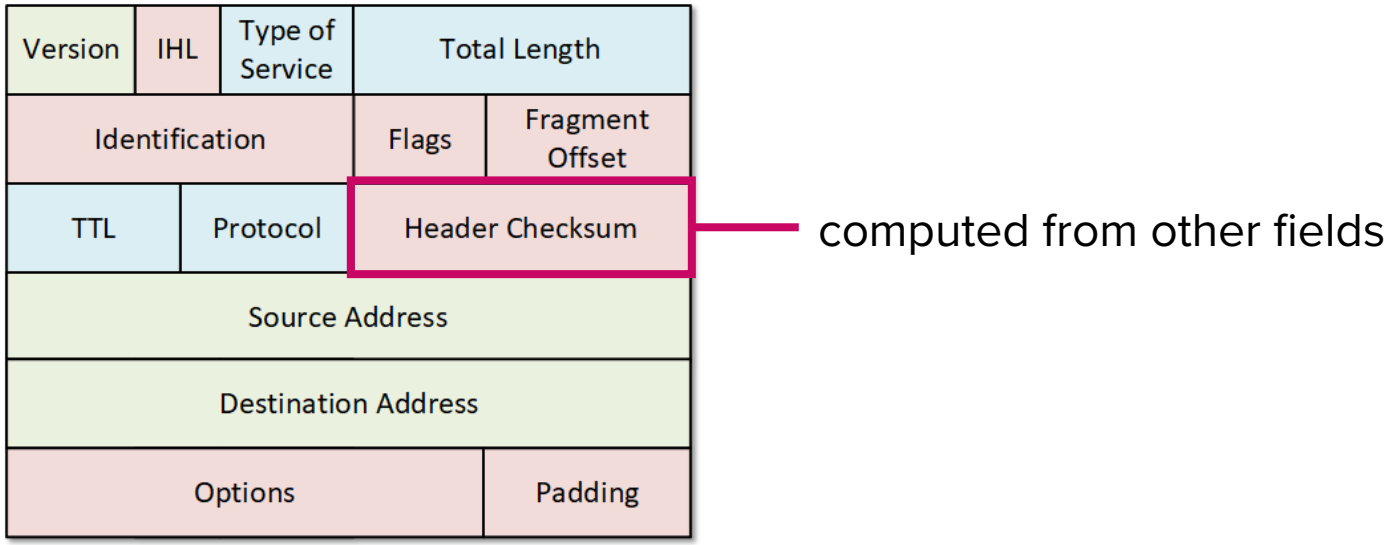
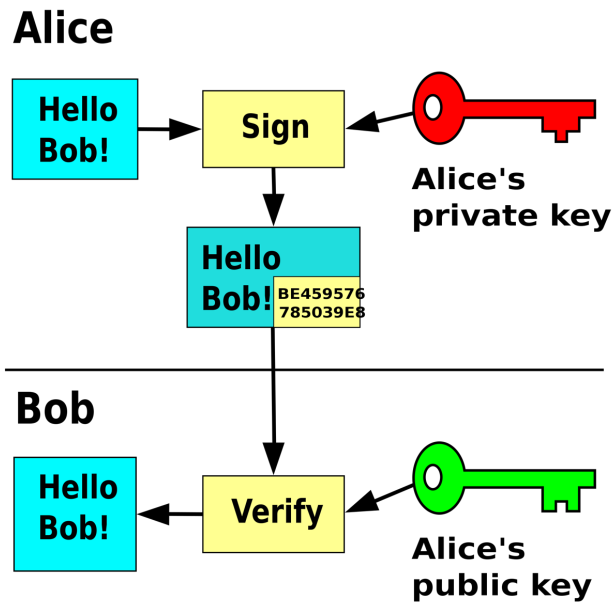


image: networkacademy.io

Examples of integrity

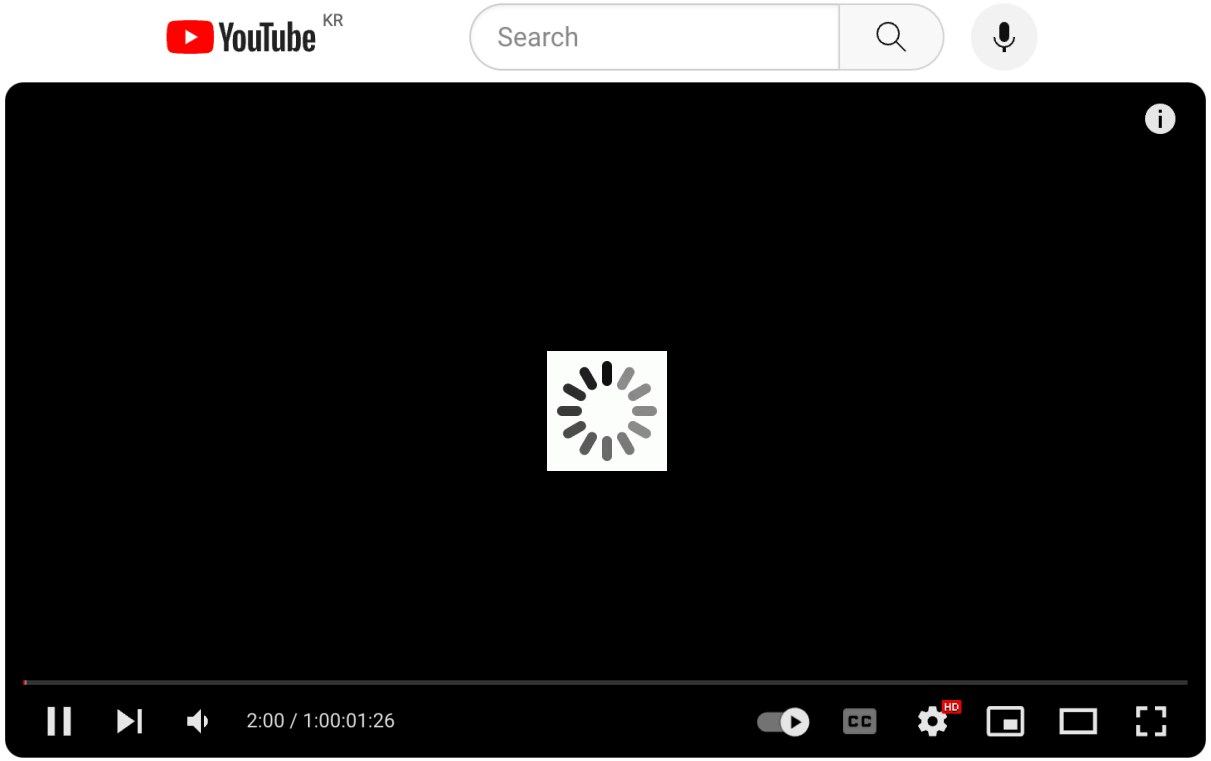
- Digital signatures
  - Will cover in week 6!



- Blockchain
  - Cryptographic hashing + consensus mechanism

Availability

- Ensuring that systems, services, and resources are accessible and usable when needed



Availability is a complex concept

- Context-dependent definition
  - Timely request-response
  - Fair allocation of resources (no starvation)
    - OS – scheduler!
  - Fault tolerant (no total breakdown)
    - e.g., emergency generator
  - Easy to use
    - Five doorlocks on the front gate for security?
  - …

Example of (broken) availability

- Colonial Pipeline attack (2021)
  - VPN (virtual private network) password got exposed
  - Attacker infects the pipeline control system with a ransomware
  - Colonial Pipeline shut down the pipeline to contain the attack
  - Gasoline shortage across the Southeast US

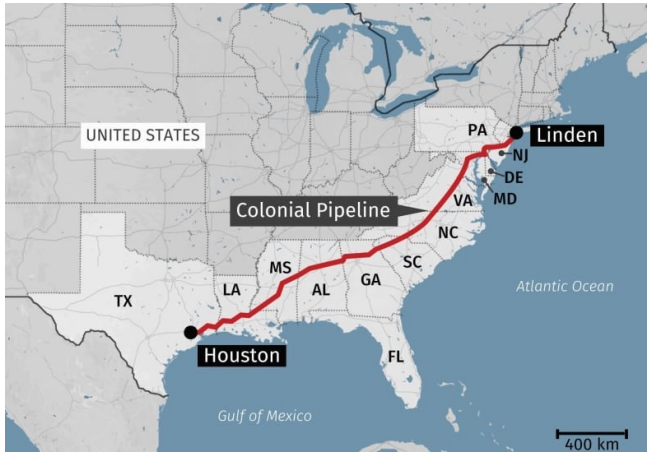


image: CBC


image: BNN Bloomberg

image: NPR

Balance matters!

- “To ensure confidentiality and integrity,
  I built an impenetrable vault, stored my data, and sealed it.”

Availability suffers :(

Balance matters!

- “For integrity, I wrote a new data management system. I create copies of my entire SSD and upload it to multiple cloud storage services. I cross-check all copies every minute so I can detect unauthorized modification of my data.”

Weaker confidentiality,
Almost no availability.

Additional concepts: AA

- Needed along with CIA for completeness

- Authenticity
  - Being genuine and being able to be verified and trusted
  - Certifying the integrity of the origin of information
- Accountability (== nonrepudiation)
  - Actions of an entity to be traced uniquely to the entity
  - A system must be able to trace security breach to the attacker
    - e.g., US government takes all 10 fingerprints when issuing a VISA

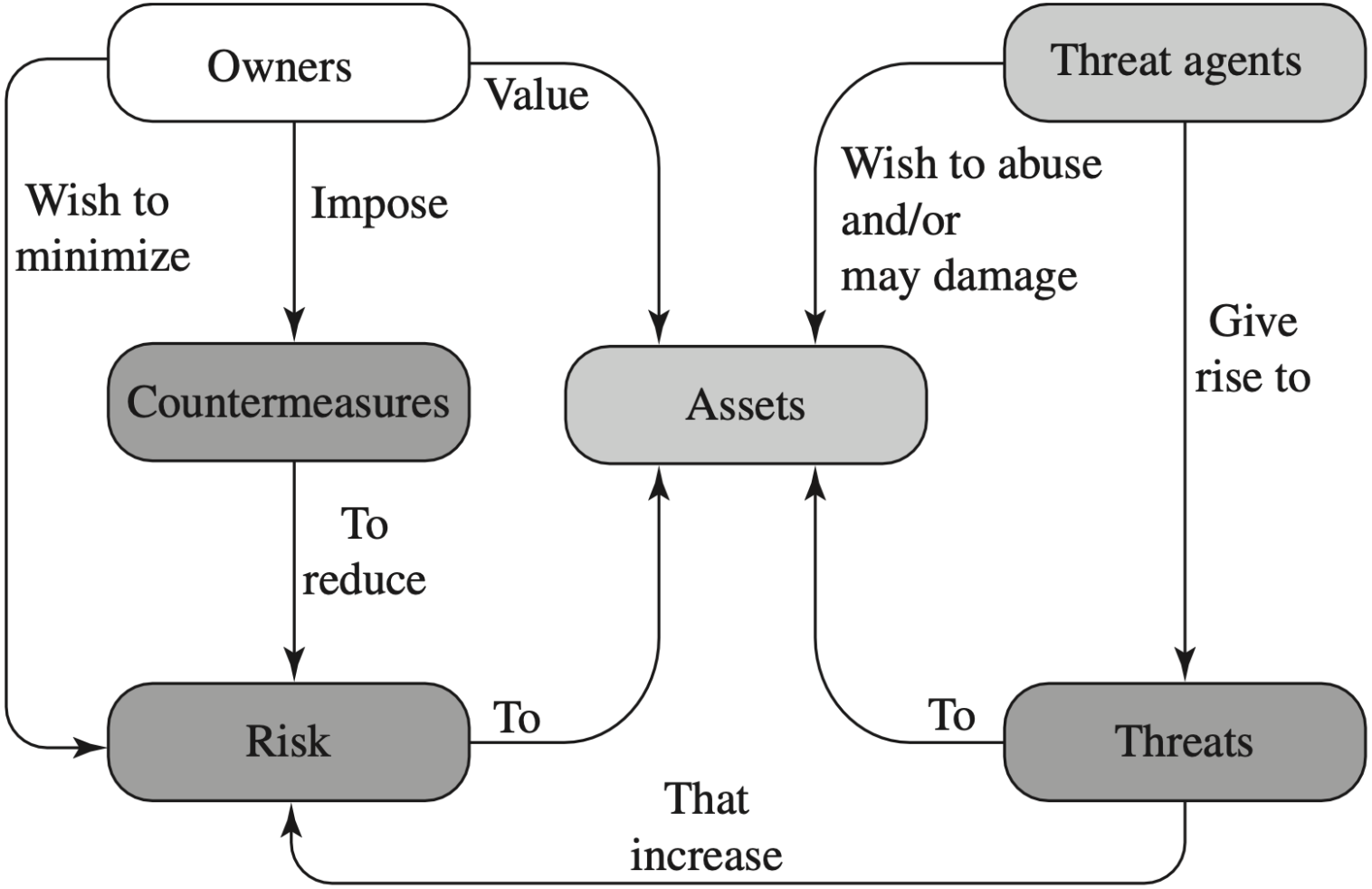
Question

- How can we determine if our system is secure?
  - In other words, how do we know if the principles of CIA+AA are being maintained?

We need a model!

Threat modeling

Security concepts and relationships



Terminology

- Asset
  - What we value
  - Hardware, software, data, communications, resources, …

Terminology

- Threat
  - A set of circumstances that can potentially impair security through unauthorized access (C), destruction (I, A), disclosure (C), modification (I), and/or denial of service (A)
    - ( ): affected property

- Attack
  - A threat that is carried out (i.e., an action)
  - “Cyber attack”, “physical attack”, “fabrication attack”, …

Terminology

- Vulnerability
  - Weakness in a system that can be exploited to cause broken CIA
    - Leakage (C), Corruption (I), unavailability (A)

- Bug vs Vulnerability
  - Bug: a flaw in a computer program or system that results in an unexpected outcome
  - Vulnerability: a bug that can be exploited by attackers to compromise security
    - A subset of bugs

Terminology

- Q) Is this program vulnerable?

```c
#include <stdio.h>
#include <string.h>

int main() {
    char buffer[5];
    strcpy(buffer, "Overflow!");
    printf("Buffer: %s\n", buffer);
    return 0;
}
```

Terminology

- Q) Is this program vulnerable? A) No

```c
#include <stdio.h>
#include <string.h>

int main() {
    char buffer[5];
    strcpy(buffer, "Overflow!");
    printf("Buffer: %s\n", buffer);
    return 0;
}
```

Vuln. Requirements

- System is buggy (T)
- Adversary has access to the bug (F)
- Adversary has capability to exploit the bug (T)

→ Not vulnerable

Terminology

- Countermeasure
  - Any device or techniques that deal with attacks
  - Includes
    - preventive methods (before attacks) and
    - recovery (after attacks)

Examples of threats and affected assets

POSTECH

	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.	An unencrypted USB drive is stolen.	
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines and Networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

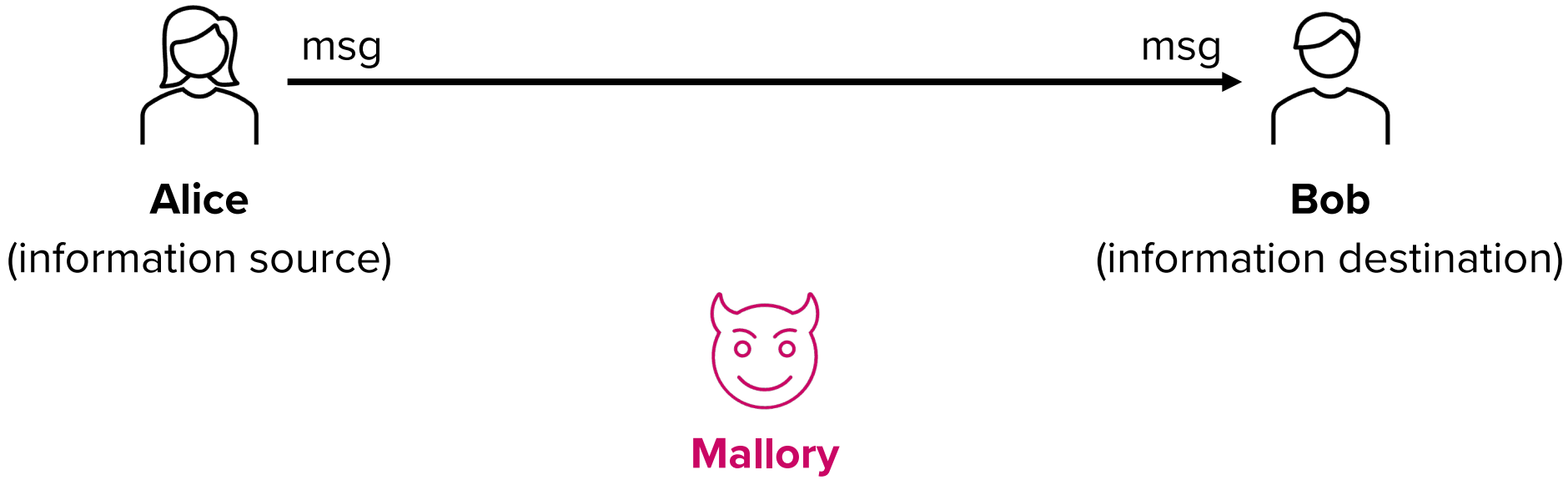
Threat modeling

- Process of systematically identifying threats to a system, such as vulnerabilities or lack of countermeasures
- In other words, threat modeling is evaluating a system from an attacker's perspective

What are attacker's capabilities?

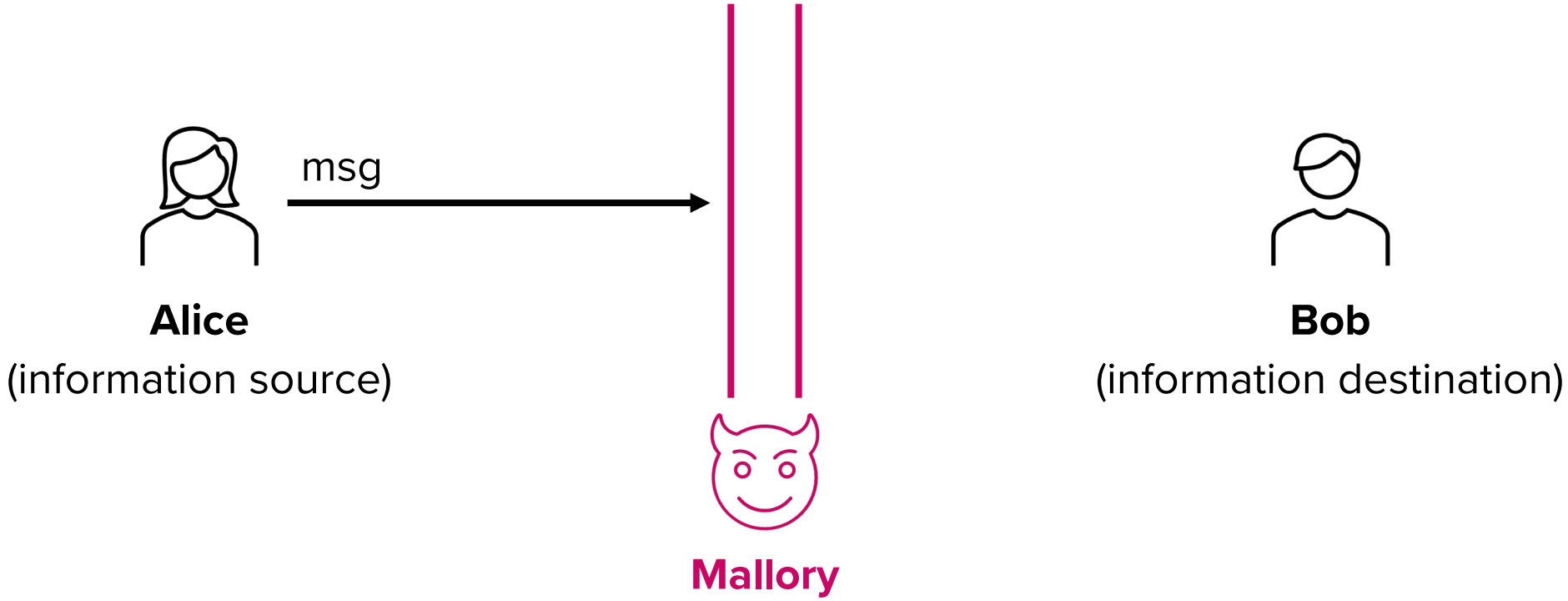
Classifying attacks – Information flow perspective

- Normal flow



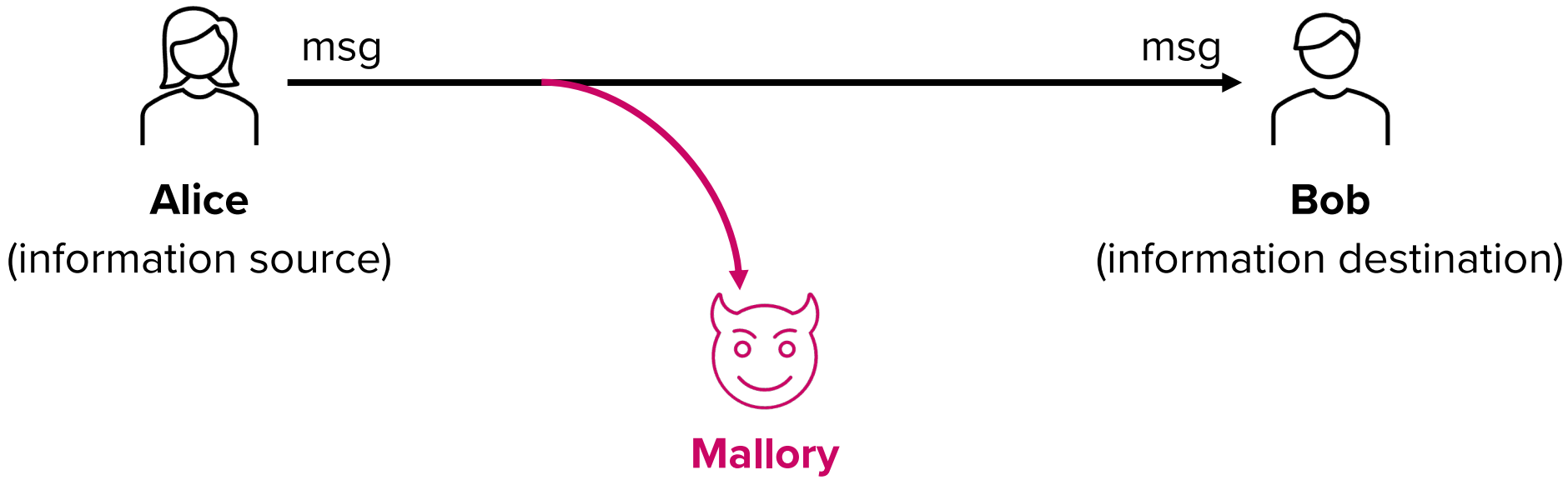
Classifying attacks – Information flow perspective

- Interruption



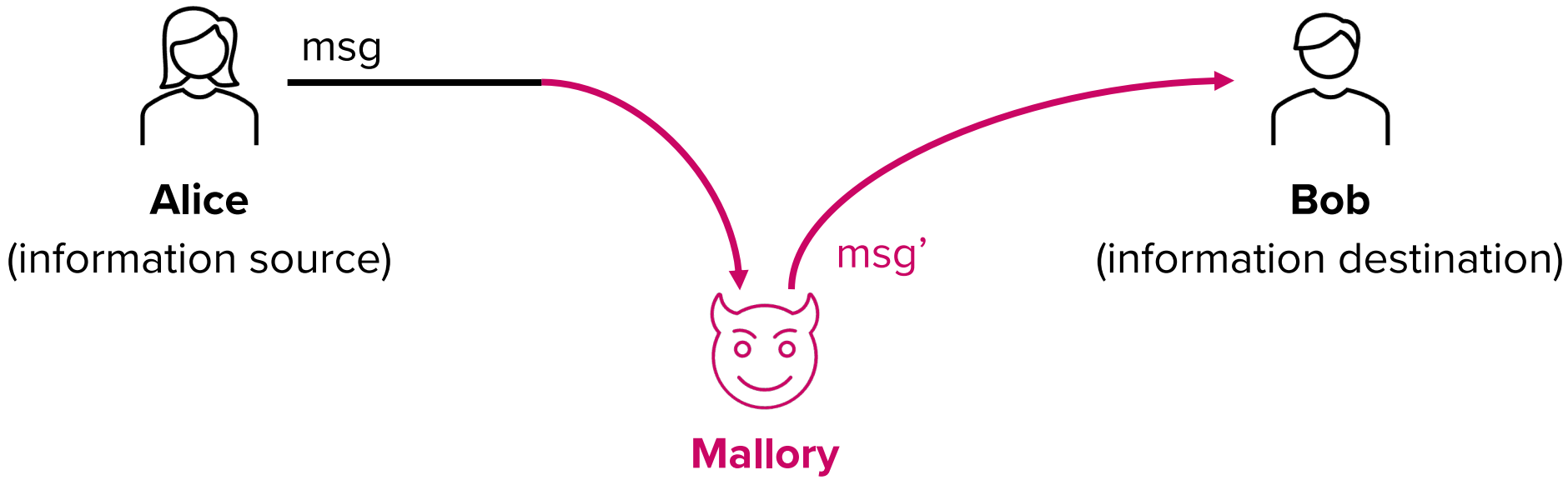
Classifying attacks – Information flow perspective

- Interception



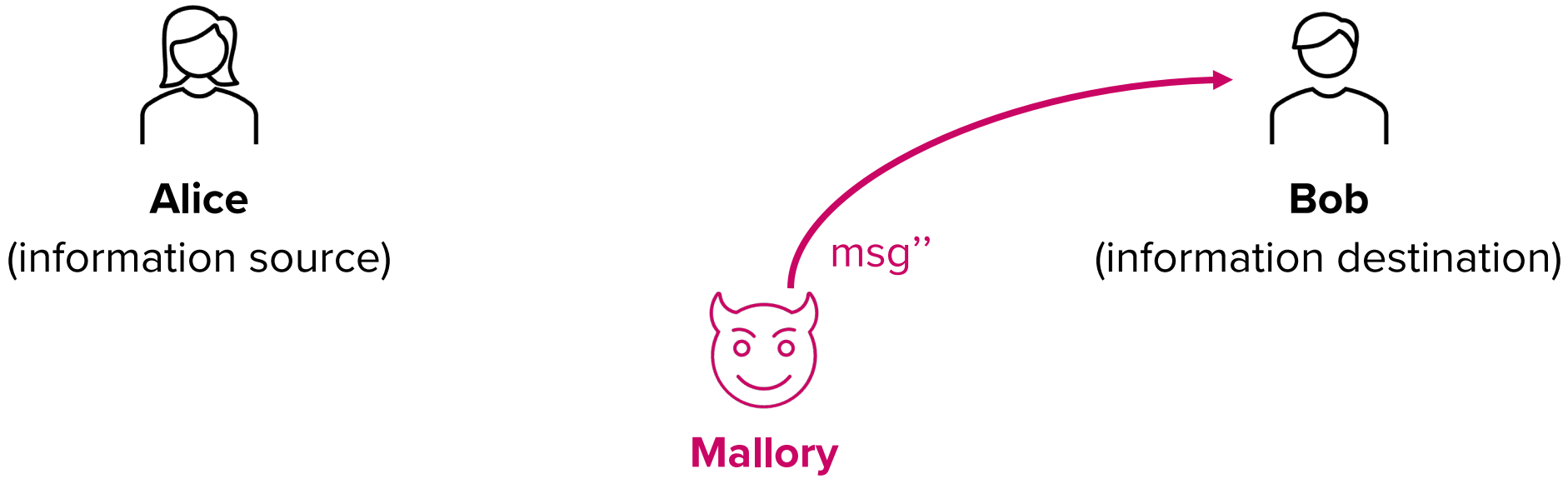
Classifying attacks – Information flow perspective

- Modification



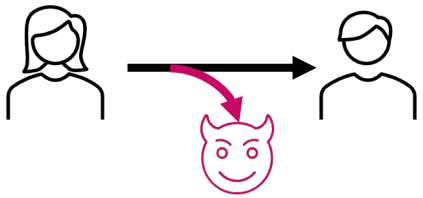
Classifying attacks – Information flow perspective

- Fabrication



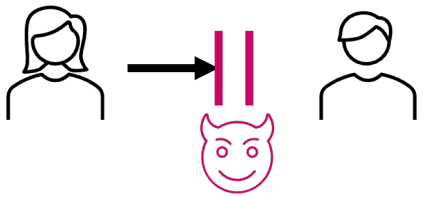
Classifying attacks – interaction perspective

- Passive attacks (original information flow is intact)

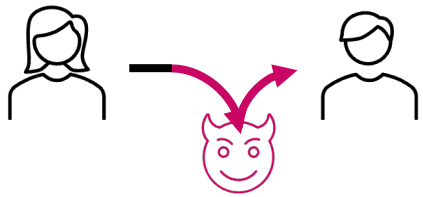


Interception
compromises confidentiality

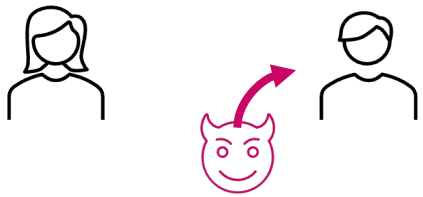
- Active attacks



Interruption
compromises availability



Modifiction
compromises integrity



Fabrication
compromises authenticity

Classifying attacks – origin perspective

- Inside attacks
  - Initiated by an entity inside the security perimeter (“an insider”)
  - Insiders are already authorized to access system resources but use them in a way not approved
    - e.g., TA colludes with a student to bump his/her grades

- Outside attacks
  - Initiated from outside the perimeter by an unauthorized or illegitimate user of the system (“an outsider”)
    - e.g., A student attacks PLMS to modify his/her grades

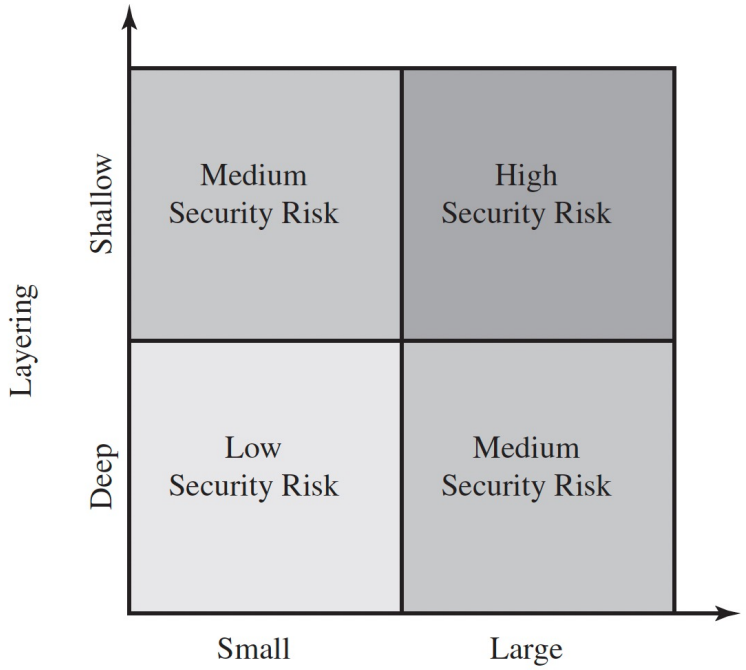
Attack surfaces

- Reachable and exploitable vulnerabilities in a system
- Categories
  - Software attack surface: vuln. in application, utility, or OS code
    - e.g., HeartBleed vulnerability (ref: Lec 1)
  - Human attack surface: vuln. created by personnel or outsiders
    - Social engineering, human errors, trusted insiders
  - Network attack surface: network protocol vulnerabilities
    - e.g., ARP spoofing

Attack surface analysis

- Enumerate possible attack points
- Assess potential scale and severity of threats to a system

→ Helps identify where security mechanisms are required



Attack tree

- A data model to represent attacks
  - Branching, hierarchical data structure that represents a set of potential techniques for exploiting vulnerabilities
  - Root: goal of the attack
  - Branches: different ways to reach the goal
  - Leaf: initiation

Attack tree example

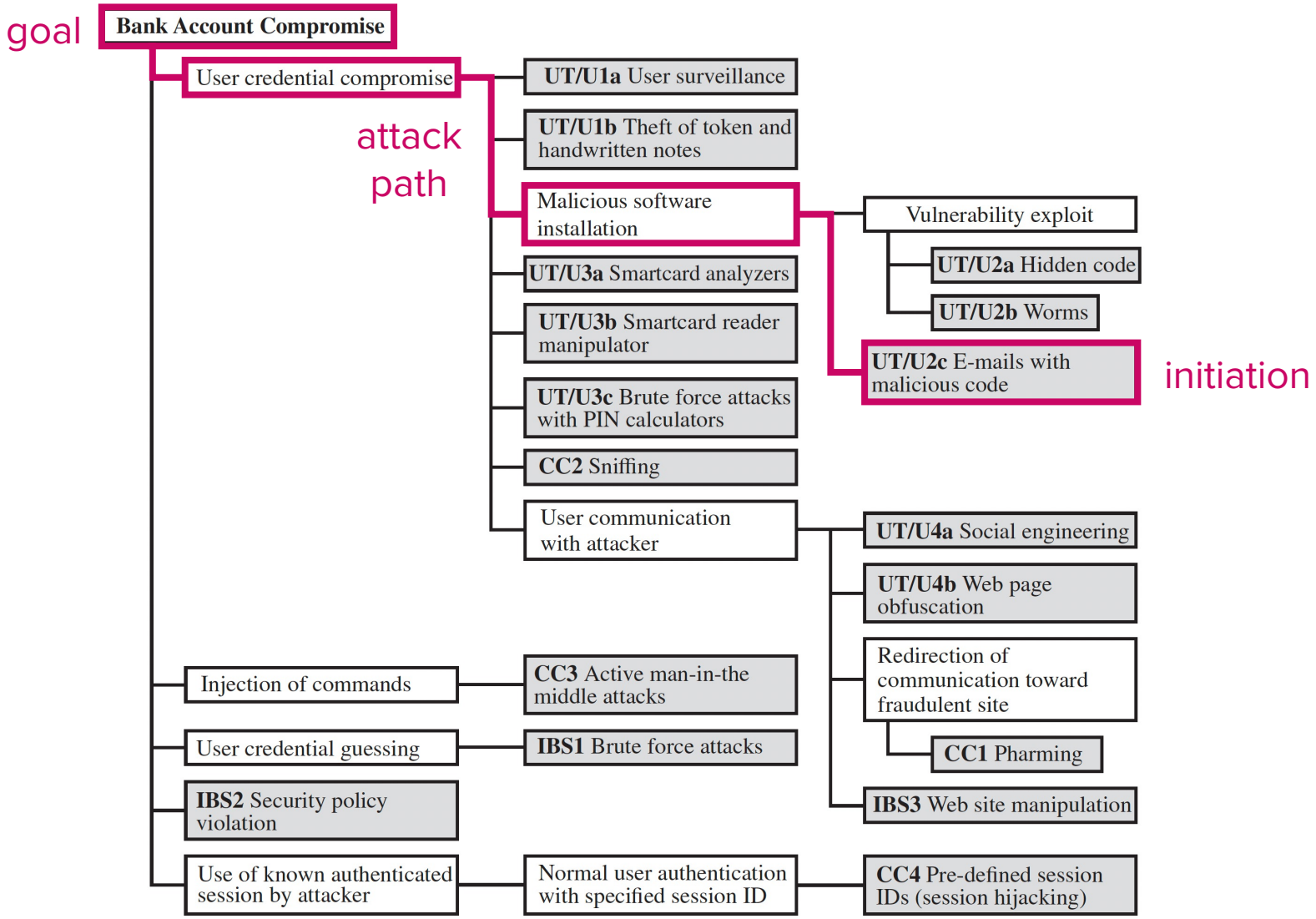


Figure 1.5 **An Attack Tree for Internet Banking Authentication**

Threat modeling in practice

- Student: “This course is so hard and there is no way I can get an A. However, in order to graduate, I desperately need an A.”

Let’s try threat modeling

Threat modeling in practice

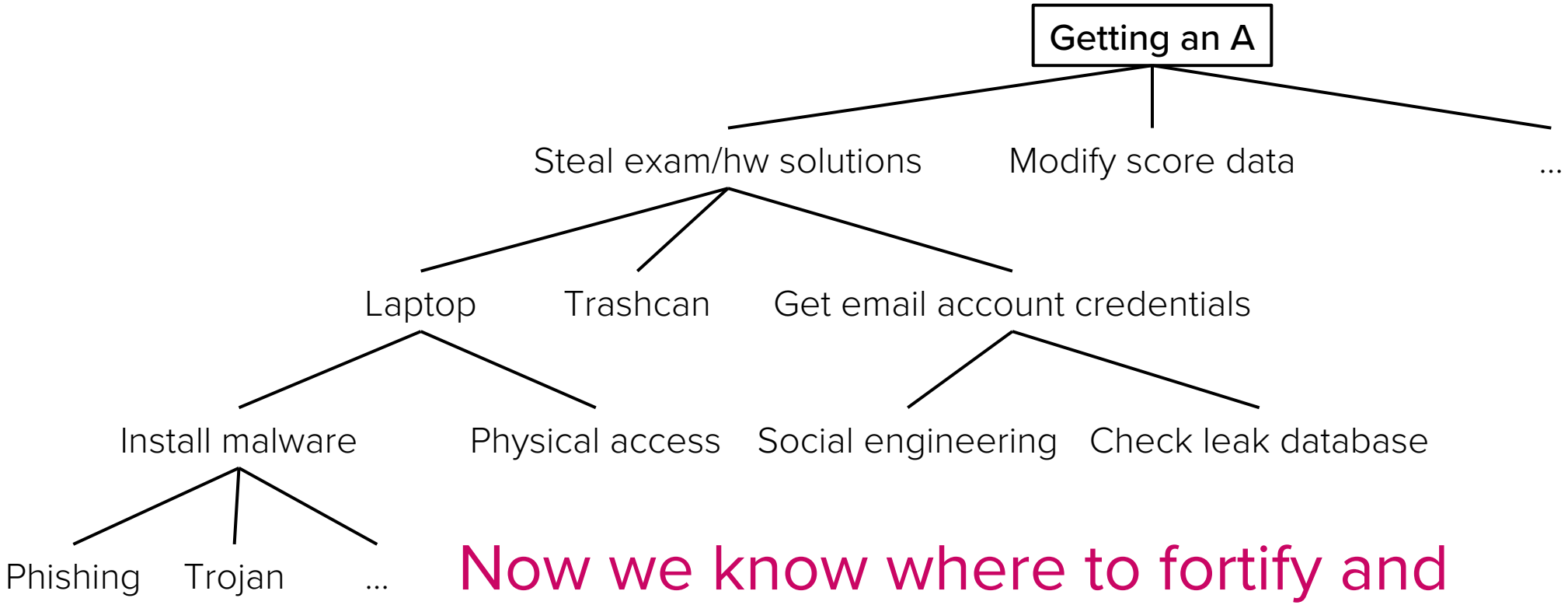
- Step 1: Enumerating potential threats
  - Hire security experts to take exams on behalf of you
  - Steal the solutions
    - Bribe the janitor (knows the password to professor's office)
    - Bribe the TA
  - Modify scores by hacking PLMS and getting administrative privilege
  - …

Threat modeling in practice

- Step 2: Enumerating attack surfaces
  - PLMS
  - TA's email account
  - Prof. Kim's laptop
  - Prof. Kim's USB drive
  - Trash bin in prof. Kim's office
  - …

Threat modeling in practice

- Step 3: Building an attack tree



Now we know where to fortify and which countermeasures are needed

Fundamental security design principles

Fundamental security design principles

- Widely agreed and tested design principles
  - These principles serve as a guiding framework for developing protection mechanisms
- Emphasis on proactive security
  - Implementing best-effort designs to preemptively mitigate threats
- Textbook reference
  - Strongly recommended to review the textbook
  - An excerpt is available on PLMS
    - [Book excerpt] Fundamental security design principles

13 principles (1)

1. Economy of mechanism
   - Keep it as simple and small as possible → less bugs
2. Fail-safe default
   - Default setting should be fail-safe (e.g., “default deny”, “read only”, …)
3. Complete mediation
   - Every access must be checked against the access control mechanism
   - Do not rely on cached content
4. Open design
   - Design of a security mechanism should be open and widely reviewed by many experts and users
   - Security by obscurity is discouraged

13 principles (2)

5. Separation of privilege
   - High privilege operations should be offloaded to separate process
   - e.g., userland API calls vs kernel mode system calls
6. Least privilege
   - Every entity should operate using the least set of privileges necessary to perform a task
   - e.g., students are granted only read permission for syllabus (PLMS)
7. Least common mechanism
   - Mechanisms used to access resources should not be shared
   - e.g., password reuse problem

13 principles (3)

8. Psychological acceptability
   - The security mechanisms should not hinder the usability or accessibility of resources
   - If not, users may opt to turn off these mechanisms
9. Modularity
   - Provide common security functions (e.g., cryptographic functions) as common modules
   - Security functions should be modular (easier to patch and upgrade when needed)

13 principles (4)

10. Isolation
    - Access isolation (public-facing resources and critical resources)
        - Server machine hosing our website and server for internal services (e.g., your lab server) are physically separated
    - Process and file isolation
        - Each user on Linux cannot access each others' files and processes
    - Security mechanism isolation
        - Prevent access to security-specific mechanisms (e.g., Intel SGX)
11. Encapsulation
    - Specific form of isolation based on object-oriented functionality

13 principles (5)

12. Layering
    - Use multiple, overlapping protection approaches
    - Also called “defense in depth”
13. Least astonishment
    - A program or user interface should always respond in the way that is least likely to astonish the user
    - Should be transparent and intutive

Week 1 summary

- We covered the basics of computer security
  - What is computer security?
  - Why is it important and why is it challenging?
  - What can go wrong?
  - What are the key principles for designing secure systems?

Coming up next: from concepts to technique

POSTECH

- Secure coding
  - Code-level mitigation for potential attacks
  - If we (as developers) are cautious enough when writing code, we MAY be able to prevent attacks!

Questions?