Lec 02: The Fundamentals

CSED415: Computer Security

Spring 2025

Seulbae Kim

POSTECH

POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Announcements

- Please vote on PLMS (“TA Office Hour Poll”)
  - The poll has been updated with more slots
- Lab 01
  - Released today and due Friday, Feb 28 at midnight
  - We will briefly discuss the setup on Feb 25
- Project team formation: Week 3
  - From teams of 5-6 members, choose a team name and team leader
  - Team leaders should submit the team information on PLMS
    - Only the team leader should make a submission
    - Refer to the assignment page for details (TBA)

Recap

- Goal of computer security
  - Protect our computer-related assets (hardware, software, data, and communications) from adversaries (malicious entities)

NIST* definition

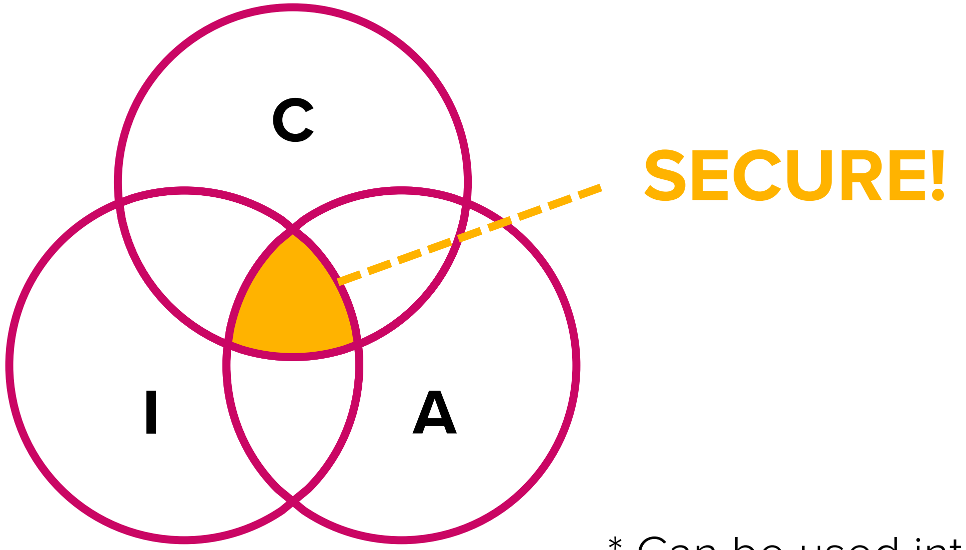
*National Institute of Standards and Technology

- 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.

Key objective:
Preserving CIA (+AA)

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



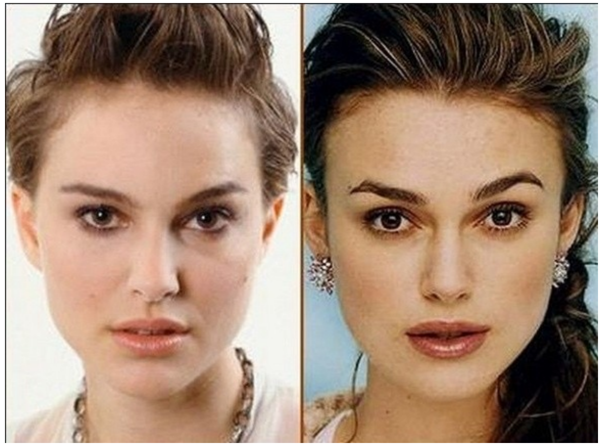
* Can be used interchangably with assets

Confidentiality

- Definition:
  - Ensuring that information is not disclosed to unauthorized parties
    - Private emails, bank accout balance, your grades, etc.
- Practical challenges and examples
  - Delegation: You need to remotely ask someone to log into your computer for a file
  - Revocation: Janitor, who knows all prof's office passwords, quits job
  - Conflicting roles: The current TA takes CSED415 next year

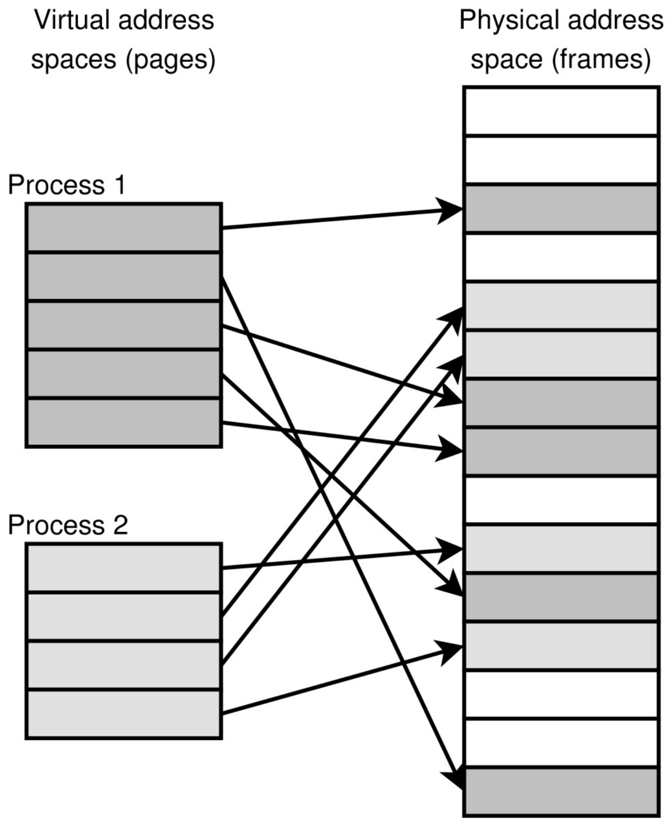
Ways to enforce confidentiality

- Enforcement
  - Identify sensitive information
  - Do authentication to verify user identities
    - “Is the person claiming to be Natalie truly Natalie?”
  - Do access control to specify “who” can access and modify that information
    - “Can Natalie read Bob’s files?”



Examples of confidentiality

- Physical access control
  - POSTECH requires an ID card for entry
- OS: Process isolation
  - Prevents a process from reading memory outside its allocated space
  - Q) What happens if a process tries to read beyond it?
- What else?



Integrity

- Definition:
  - Preventing unauthorized modification of information
    - Note: Integrity is not about disallowing ANY change!
  - Authorized changes should still happen
    - 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 you spent.

Types of integrity

- Data integrity
  - Information changes only according to specific, authorized rules
- System integrity
  - System performs its intended function as is
    - e.g., `add_float(0.1, 0.2)` should return `0.3`

Ways to assure integrity

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

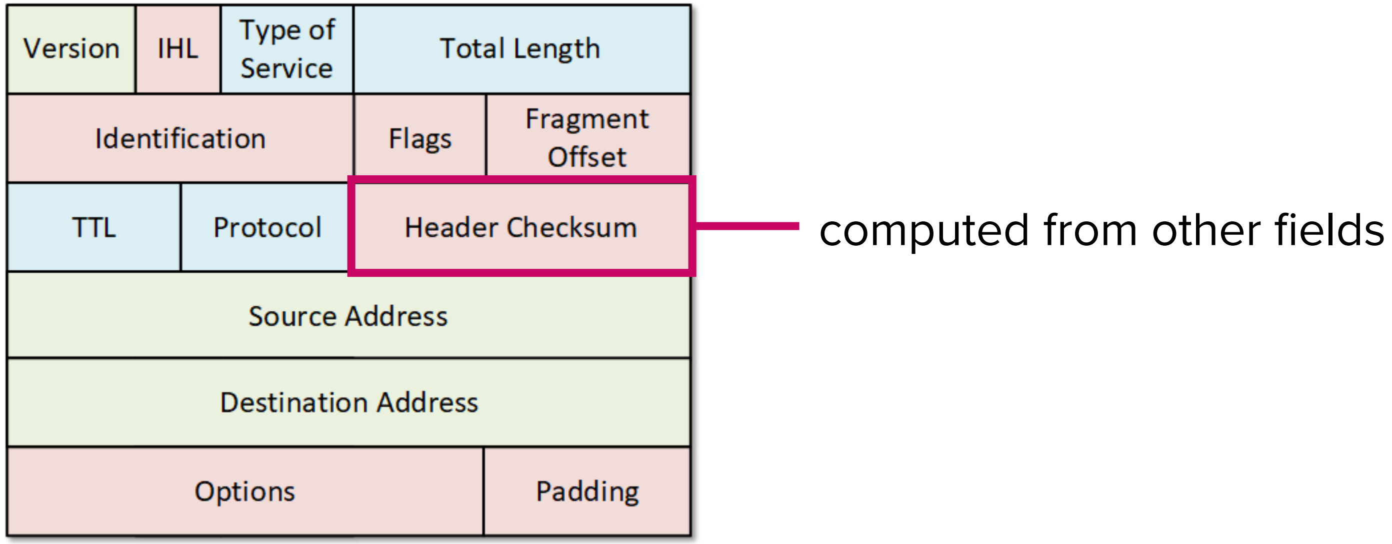
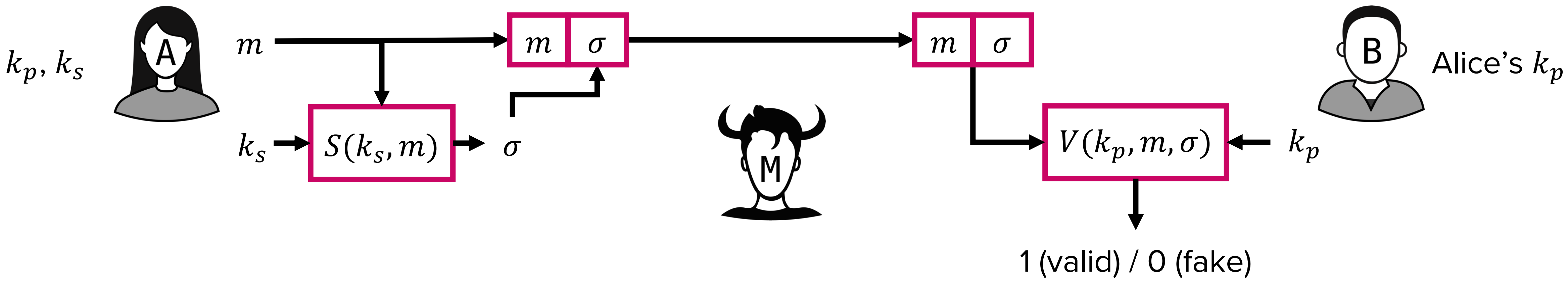


image: networkacademy.io

Examples of integrity

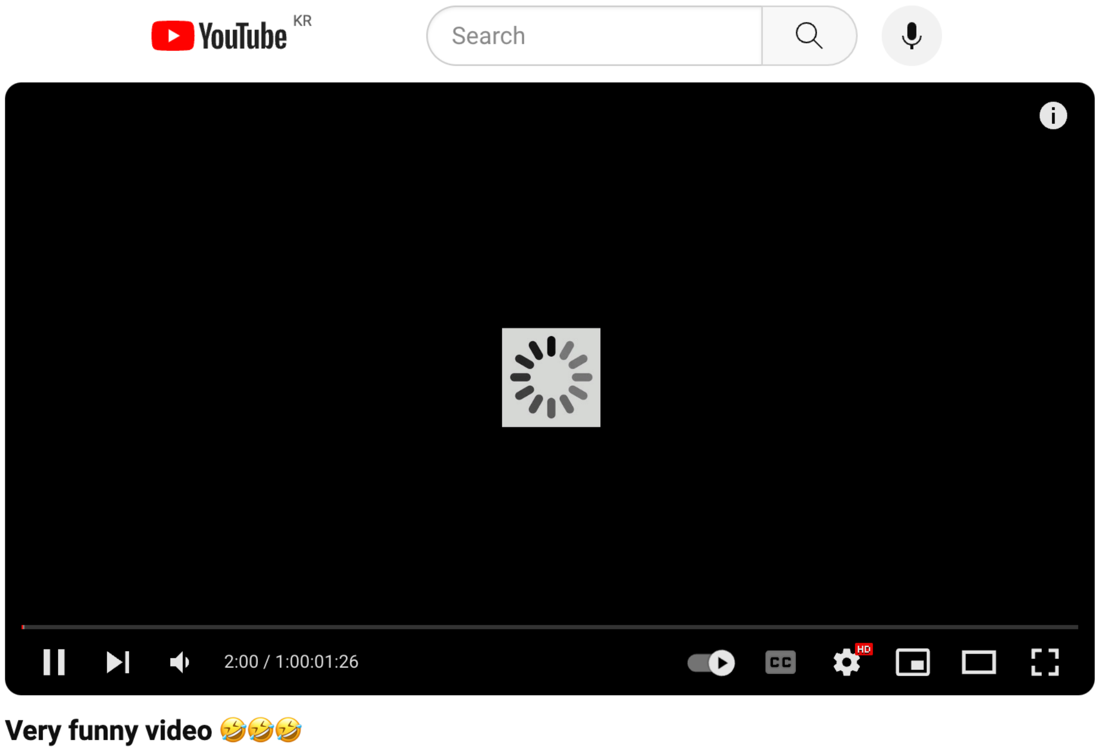
- Digital signatures
  - Will cover in Week 6!



- Blockchain
  - Combines hashing and consensus mechanisms

Availability

- Definition:
  - Systems, services, and resources are accessible and usable when needed



Ways to assure availability

- Ensure timely request-response
  - Make system efficient (e.g., avoid slow algorithms)
- Ensure Fair allocation of resources to prevent starvation
  - Task of the schedulers of an OS
- Make system fault tolerant to prevent total breakdown
  - e.g., emergency generator
- Ensure ease of use
  - Five doorlocks on the front gate for security?

Example of availability failure

- Colonial Pipeline (CP) ransomware attack (2021)
  - VPN (virtual priv. network) password of pipeline sysadmin got leaked
  - Attacker infects the pipeline control system with a ransomware
  - CP shut down the pipeline system to stop the spread of ransomware
  - Gasoline shortage across the Southeast US

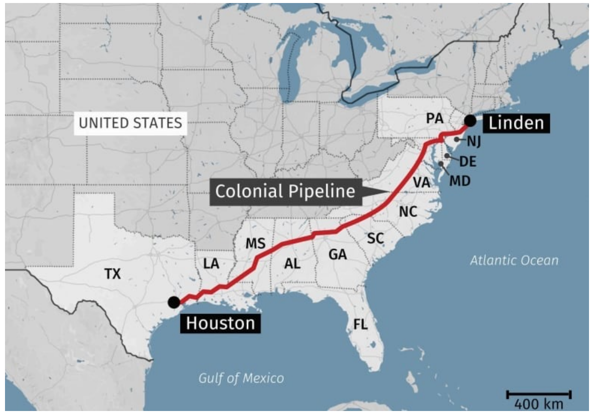


image: CBC

image: BNN Bloomberg

image: NPR

Authenticity & Accountability (AA)

POSTECH

- Additional concept that complements the CIA triad
- Authenticity
  - Ensures that the origin of information is legitimate
  - “Hey Bob, Alice says she hates you” means nothing w/o authenticity
- Accountability (Nonrepudiation)
  - Ability to uniquely trace actions to an entity
  - A system must be able to identify who performed an action, especially in security breaches
    - e.g., US government takes all 10 fingerprints when issuing a VISA

Importance of C-I-A balance

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

Availability suffers :(

Importance of C-I-A balance

- “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.

CIA+AA summary

- CIA+AA provide a conceptual background to evaluate the security of a system
  - Balanced enforcement is important
- Question:
  - In practice, how can we determine if CIA+AA are guaranteed?

We need a model!

Threat modeling

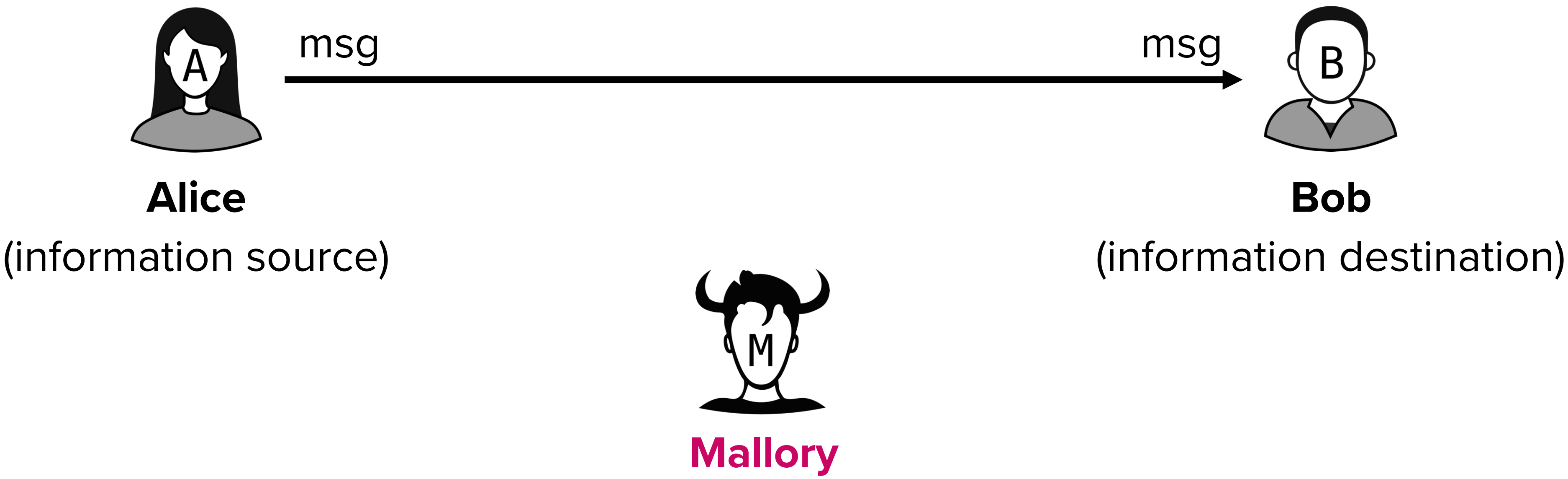
Threat modeling

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

First, we need to understand attacker's capabilities

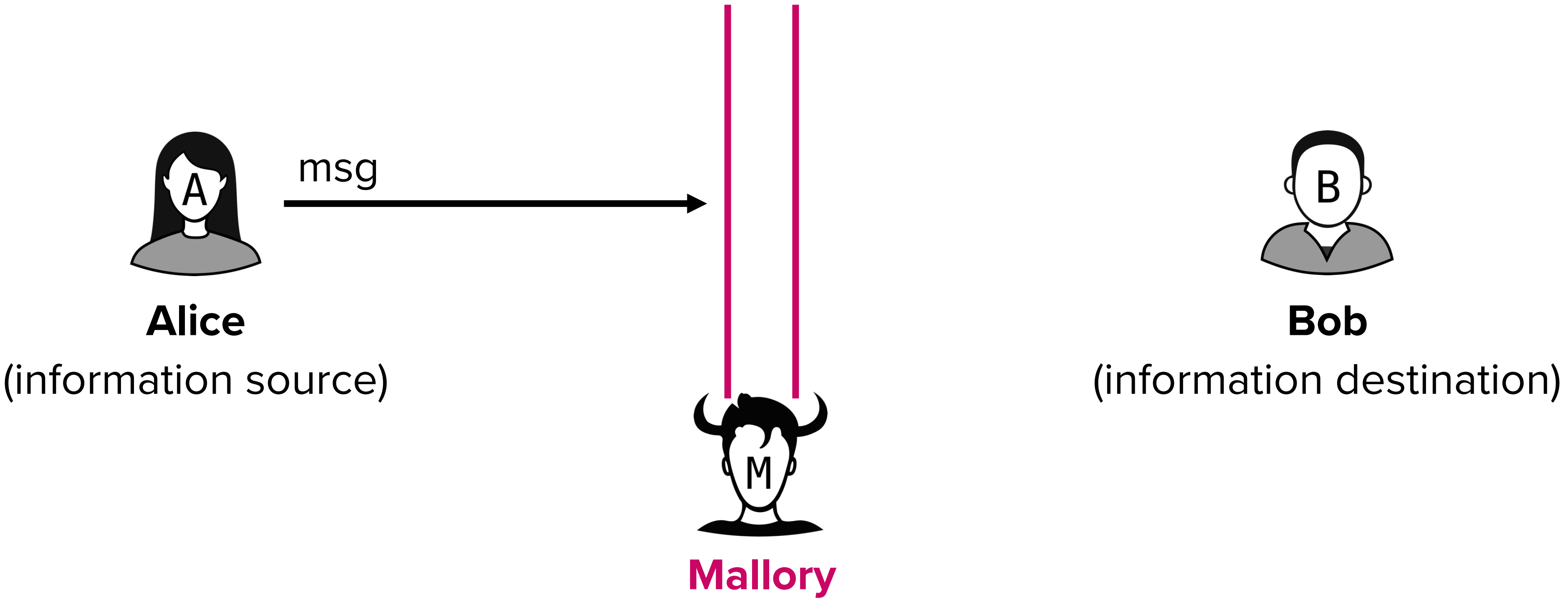
Classifying attacks – By information flow

- Normal flow



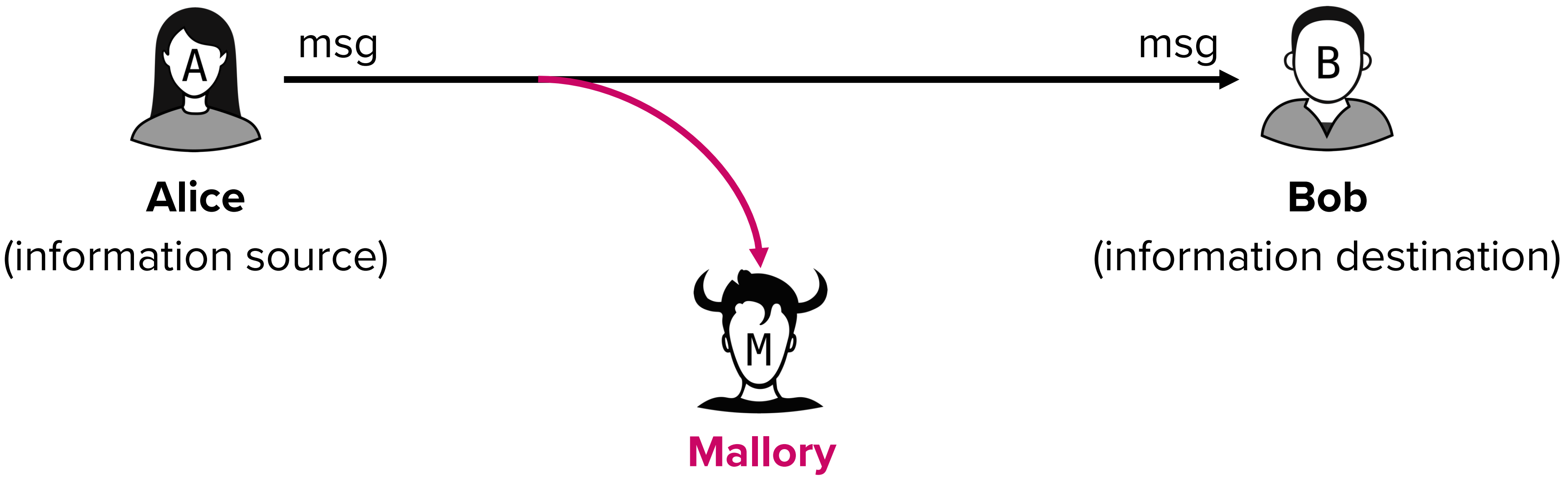
Classifying attacks – By information flow

- Interruption



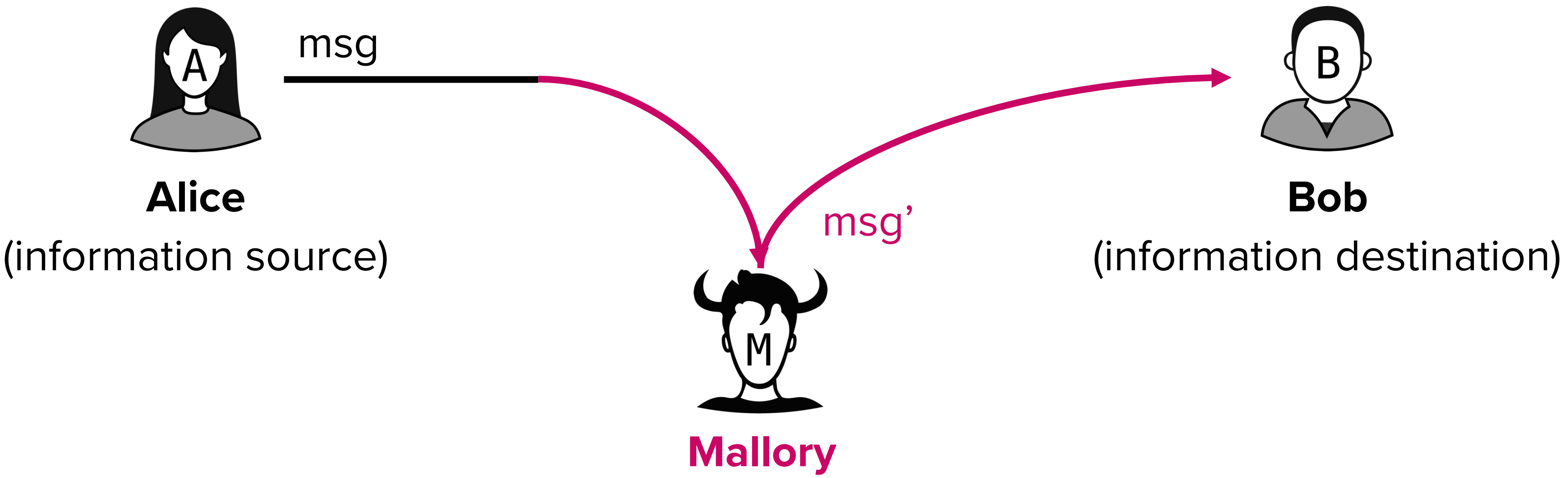
Classifying attacks – By information flow

- Interception



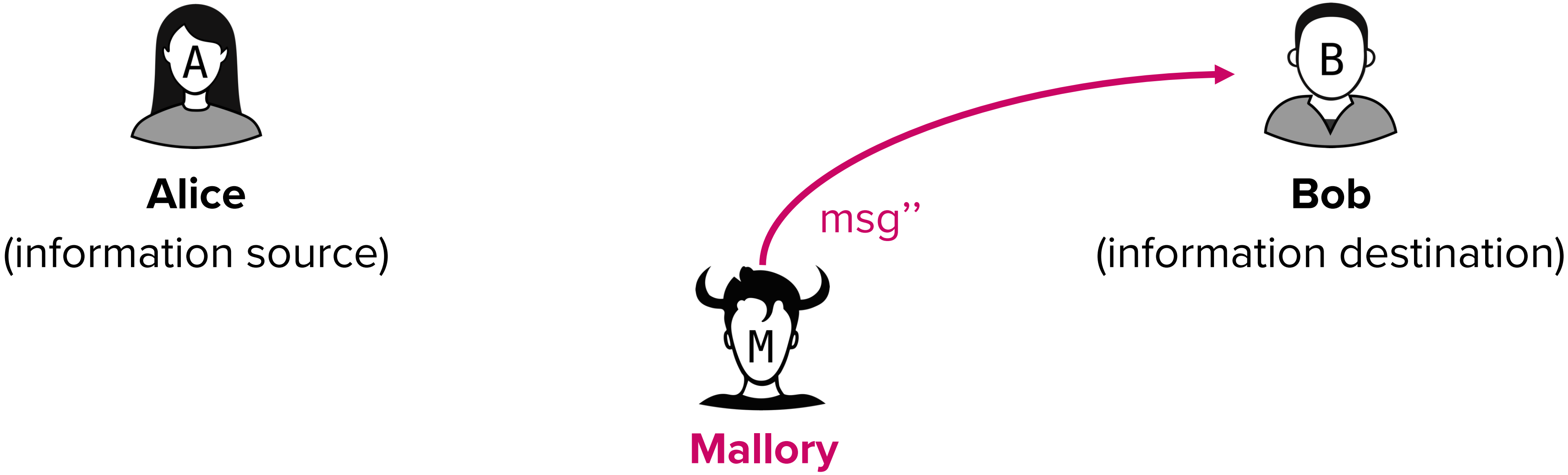
Classifying attacks – By information flow

- Modification



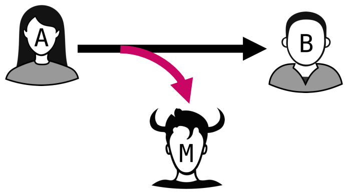
Classifying attacks – By information flow

- Fabrication



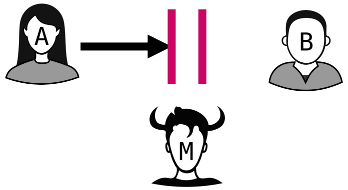
Classifying attacks – By interaction

- Passive attacks (original information flow is intact)

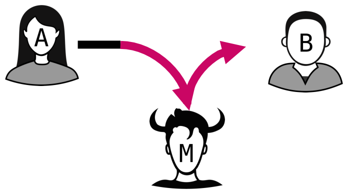


Interception
(compromises confidentiality)

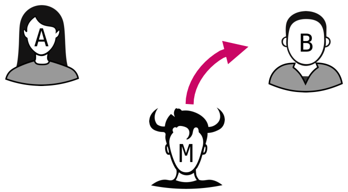
- Active attacks (information flow is altered)



Interruption
(compromises availability)



Modifiction
(compromises integrity)



Fabrication
(compromises authenticity)

Classifying attacks – By origin

- 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 up
- 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

- Definition:
  - Reachable and exploitable vulnerabilities in a system
- Categories
  - Software attack surface: Vulnerabilities in application or OS code
    - e.g., HeartBleed vulnerability (recall *Lecture 1*)
  - Human attack surface: Vulnerabilities created by insiders or outsiders
    - e.g., Victims of social engineering, trusted insiders
  - Network attack surface: Insecure protocols
    - e.g., ARP spoofing

Attack surface analysis

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

→ Helps identify where security mechanisms are required

- Representation
  - Identified attack surface can be contextualized as an “attack tree”

Attack tree

- Definition:
  - An hierarchical data structure to represent the different ways an attacker could achieve a specific malicious goal
    - Each level of the tree represents sub-goals and more granular techniques
  - Legend
    - Root: Goal of the attack
    - Branches: Different ways to reach the goal
    - Leaf: Initiation of an attack

Attack tree example

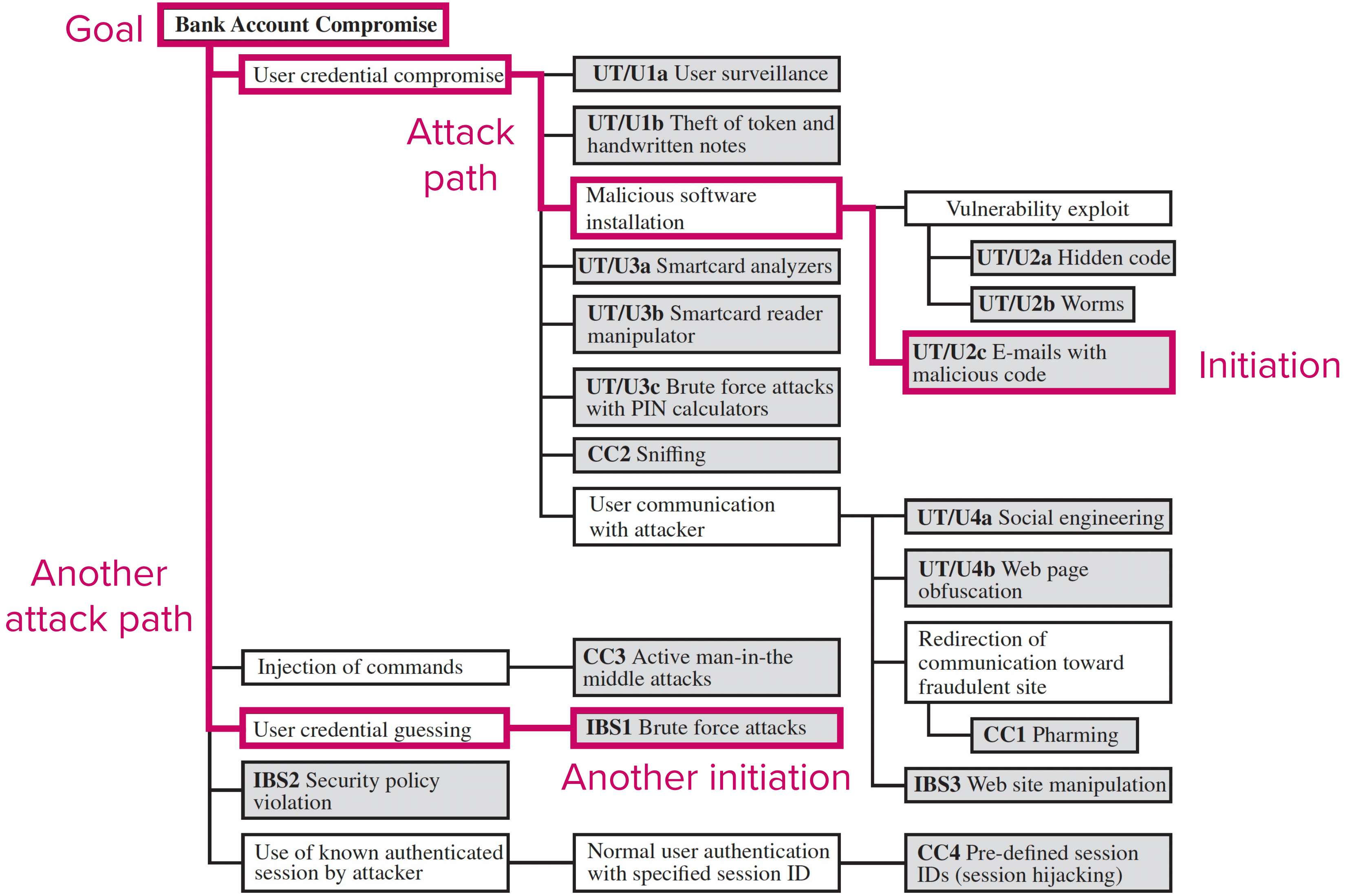
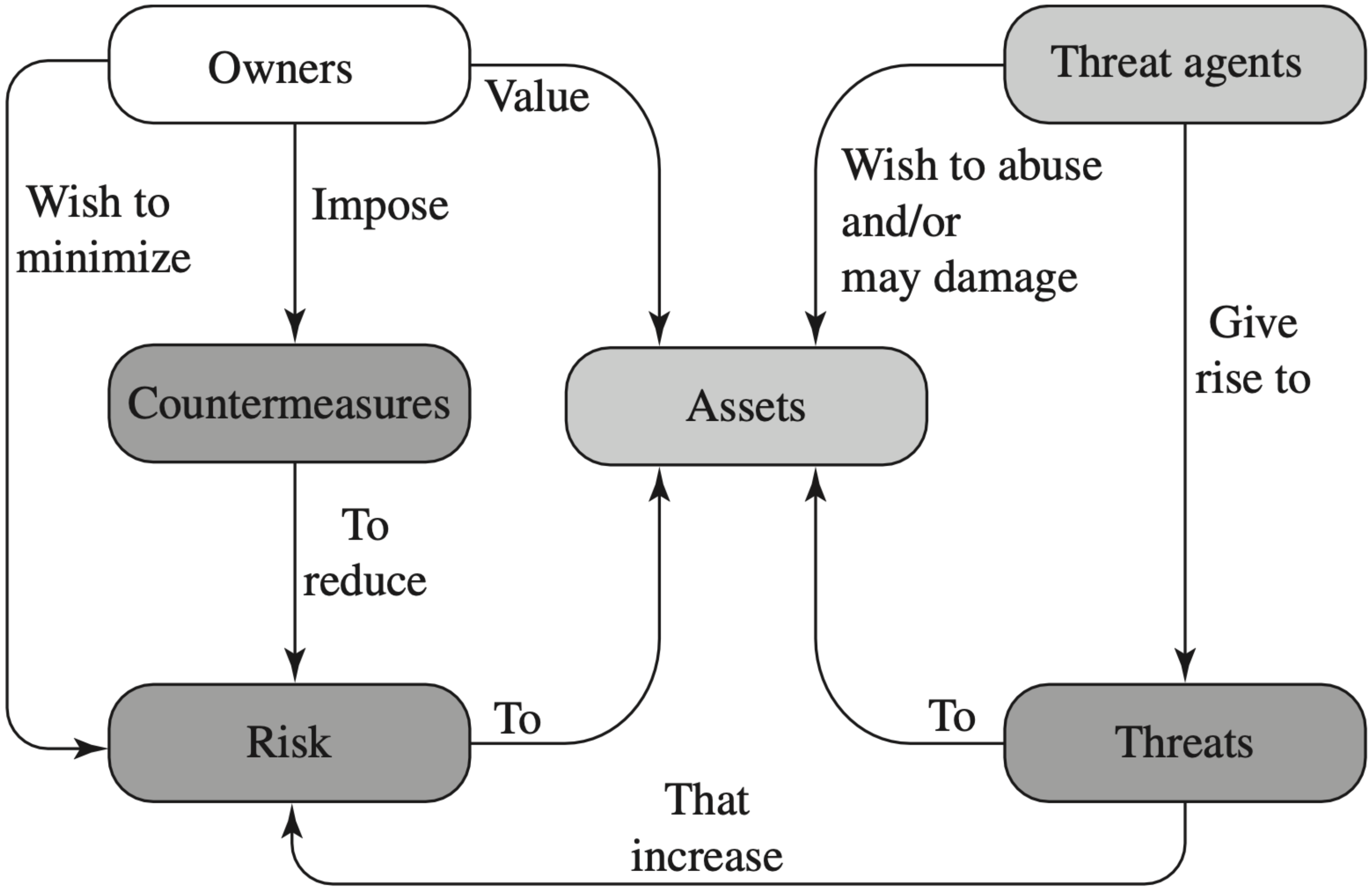


Figure 1.5 **An Attack Tree for Internet Banking Authentication**

Security concepts and relationships



Terminology

- Two parties involved
  - Owner
  - Threat agents (adversaries)
- Asset
  - What the owner values
  - Hardware, software, data, communications, resources, …

Terminology

- Threat
  - A set of circumstances that can potentially impair security through unauthorized access (C), destruction (I, A), information 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”, …
- Risk
  - Likelihood of threat from occurring

Terminology

- Vulnerability
  - Weakness in a system that can be exploited to break CIA
    - Leakage (C), Corruption (I), Service disruption (A)
- Bug vs Vulnerability
  - Bug: A flaw leading to unexpected behavior
  - Vulnerability: A bug exploitable by attackers to compromise security
    - Vulnerability is 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;
}
```

Vulnerability Requirements

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

→ Not vulnerable

Terminology

- Q) Is this program vulnerable? A) YES!

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

int main(int argc, char* argv[]) {
  char buffer[5];
  strcpy(buffer, argv[1]);
  printf("Buffer: %s\n", buffer);

  return 0;
}
```

Vulnerability Requirements

- System is buggy (T)
- Adversary has access to the bug (T)
  - Via `argv[1]`
- Adversary has capability to exploit the bug (T)
  - By providing a long string

→ Vulnerable!

Terminology

- Countermeasures
  - Any technique or device that prevents, detects, or recovers from attacks
  - Other terms
    - Mitigation
    - Fortification

Threat modeling in practice

- Example situation:
  - A student wants to cheat: “This course is too difficult and there seems to be no way I can get an A. However, I need an A to graduate.”

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

POSTECH

- 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

- Getting an A
  - Steal exam/hw solutions
    - Laptop
      - Install malware
        - Phishing
        - Trojan
        - …
      - Physical access
    - Trashcan
    - Get email account credentials
      - Social engineering
      - Check leak database
  - Modify score data
  - …

Now we know where to fortify and which countermeasures are needed

Threat modeling summary

- Summary
  - Purpose: Identify potential threats to a system
  - Steps:
    - Enumerate threats → Identify attack surfaces → Build an attack tree
- A caveat
  - Threat modeling is for existing systems
  - Can we design secure systems in the first place?

Yes, by referring to the security design principles!

Fundamental security design principles

Fundamental security design principles

POSTECH

- Widely agreed and tested design 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 fundamental security design principles

1. Economy of mechanism
   - Keep the design of security mechanism as simple and small as possible, since complexity often leads to more bugs
   - Example: Minimalistic firewall rules (small, well-defined set of rules)
2. Fail-safe default
   - When system fails or encounters an error, it should default to a secure state
   - Example: “Default deny” policy in access control – A digital door lock should be locked upon power loss

13 fundamental security design principles

3. Complete mediation

- Every access to a resource must be checked for authorization
- Access permissions should not be “cached”
- Example:
  - Web requests must include up-to-date session token. Web applications must validate the token instead of granting indefinite access after login

13 fundamental security design principles

4. Open design
   - The security architecture should not rely on the secrecy of design or implementation
   - Designs of security mechanisms should be open and widely reviewed by many experts and users
   - “Security by obscurity” is discouraged
   - Example:
     - Open-source cryptographic libraries (e.g., OpenSSL) are openly reviewed by the security community

13 fundamental security design principles

POSTECH

5. Separation of privilege
    - Multiple privilege attributes are required to achieve access to a restricted resource
    - Examples:
        - Two-Factor Authentication: Access to sensitive accounts requires something you know (password) and something you have (smartphone)
        - Dual control in banking: Withdrawing large funds require signatures from at least two different managers

13 fundamental security design principles

6. Least privilege
    - Every entity (user or process) should have the minimal privileges necessary to perform its task and nothing more
    - Examples:
        - Students are granted only read permission for syllabus (PLMS)
        - A web application's database user is granted only SELECT and INSERT privileges, not DROP TABLE or CREATE TABLE privileges
        - Sandbox: Mobile apps run in sandboxes, preventing them from reading or writing outside their own designated data

13 fundamental security design principles

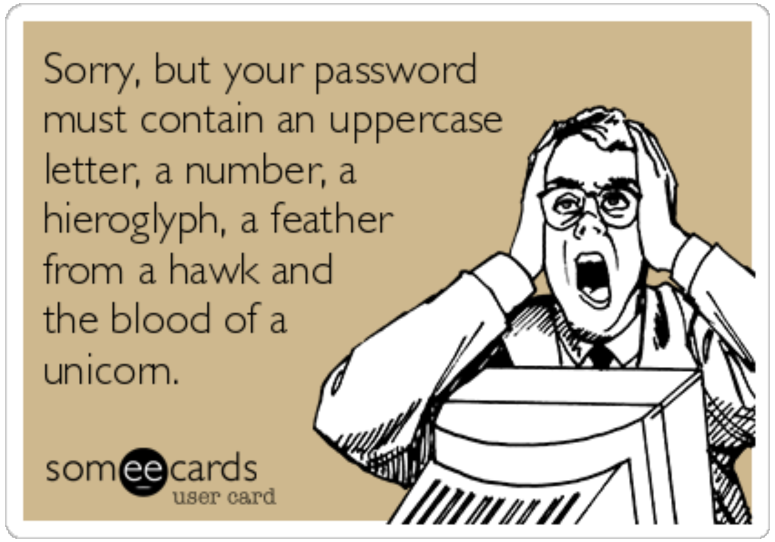
POSTECH

7. Least common mechanism
   - Systems should not unnecessarily share mechanisms or resources
   - Minimizing shared resources reduces the chance that a flaw or misuse in one shared component compromises multiple users
   - Examples:
     - Admin and user logins are served on different subdomains, rather than single shared login page
     - Compartmentalization: Each microservice (e.g., Netflix - authentication, payment, video streaming, analytics, etc.) runs on its own virtual machine or container with dedicated resources

13 fundamental security design principles

8. Psychological acceptability

- The security mechanisms should not be so burdensome or unintuitive that users refuse to use them
- Example: Requiring login password of length 48



13 fundamental security design principles

9. Modularity
   - Security functions should be designed as independent modules, making them easier to develop, maintain, and update
   - Example:
     - Rather than implementing cryptographic schemes, applications should utilize a cryptographic library, which can be upgraded (e.g., from SHA-1 to SHA-256) without rewriting the entire application

13 fundamental security design principles

10. Isolation

- Isolate critical assets and processes so that compromises in one area do not immediately spread to others
- Examples
  - Access isolation (public-facing resources and critical resources)
    - Public web servers and internal computation nodes are physically separated
  - Process and file isolation
    - Each user on Linux cannot access each others' files and processes
  - Security mechanism isolation
    - Hardware security features like Intel SGX isolate sensitive code/data from the main OS

11. Encapsulation

- Specific form of isolation based on object-oriented functionality

13 fundamental security design principles

POSTECH

12. Defense in depth (Layering)

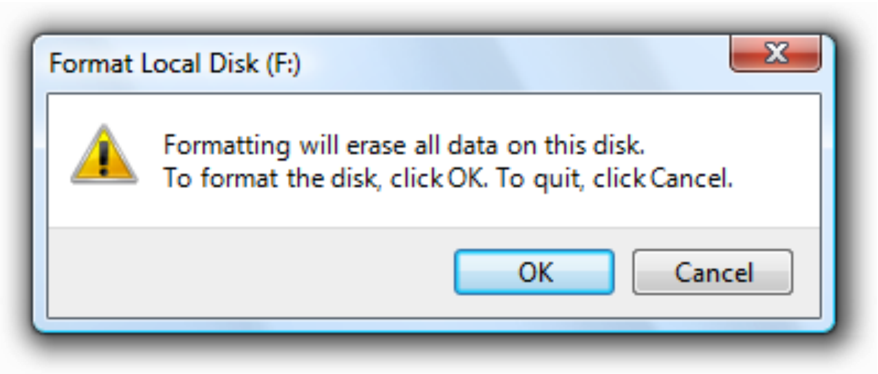
- Use multiple, overlapping protection strategies so that if one layer is breached, subsequent layers still protect the system
- Example:
  - Firewall + Intrusion detection system (IDS) + OS security: An attacker who penetrates the firewall must still evade the IDS and local OS security features to succeed

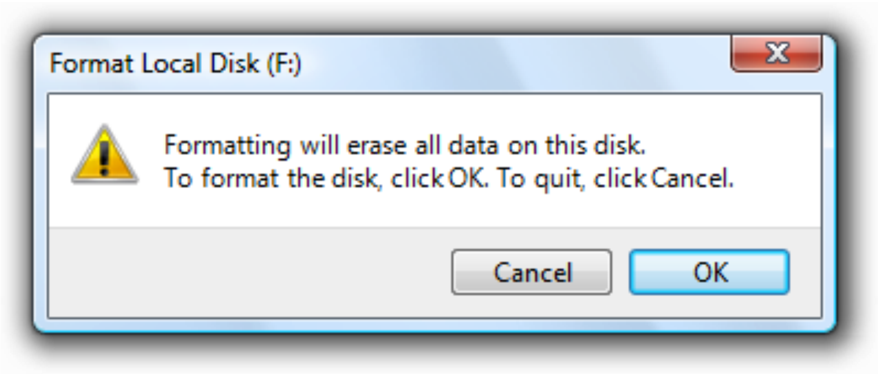
13 fundamental security design principles

POSTECH

13. Least astonishment

- A program or user interface should behave in a way that does not surprise its users
- Surprises often cause users to misunderstand or bypass security
- Example:





Week 1 summary

- We covered:
  - What computer security is
  - Why it is important, yet challenging
  - CIA triad and threat modeling
  - Key security design principles

Coming up next: From concepts to technique

- Secure coding
  - We will explore code-level mitigation for preventing attacks
  - Idea: If we (software developers) are cautious enough when writing code, we MAY be able to avoid or mitigate vulnerabilities

Questions?