Please check your attendance using Blackboard!

Lecture 1 – Software Security Principles

[COSE451] Software Security

Instructor: Seunghoon Woo

Spring 2024

Overview

- **Definition of Security**
- **Basic terms**

Definition of Security

- **What is the definition of "computer security"?**
  - Finding vulnerabilities?
  - Preventing hacking?
  - Protecting sensitive data?

Definition of Security

- **Fundamental concepts of security**
  - Ensuring the CIA!
    - **C**onfidentiality
    - **I**ntegrity
    - **A**vailability

Confidentiality (기밀성)

- **Keeping data and resources hidden**
  - Granting permission to information solely to authorized users
  - This guarantees that an attacker cannot obtain protected data

Confidentiality

- **Keeping data and resources hidden**
  - Granting permission to information solely to authorized users
  - This guarantees that an attacker cannot obtain protected data

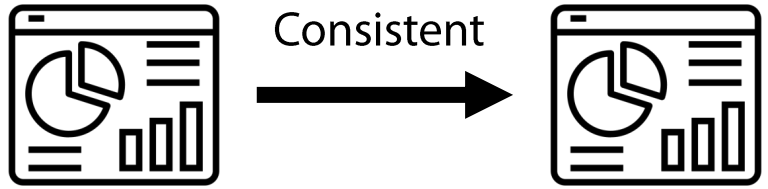
- **If confidentiality is broken?**
  - Personal information leakage, account and password leakage, etc.
  - E.g., The confidential information of a user is exposed in an unencrypted text

Confidentiality

- **Methods to ensure confidentiality**
  - Cryptography
    - Encrypting data with a cryptographic key will assure confidentiality
    - Only those with the decryption key can access the contents
  - Authentication
    - Proof of your identity and conduct (e.g., login process)
    - Only authorized users have access to the data

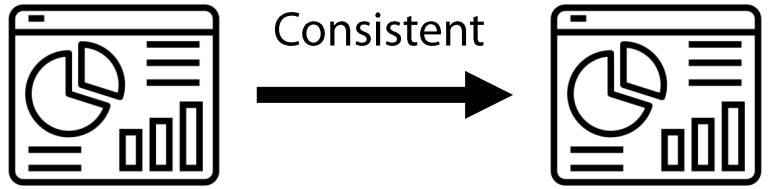
Integrity (무결성)

- **Ensuring data remains unchanged (as intended)**
  - Limiting the modification of information to privileged entities
  - This guarantees that an attacker cannot modify protected data



Integrity

- **Ensuring data remains unchanged (as intended)**
  - Limiting the modification of information to privileged entities
  - This guarantees that an attacker cannot modify protected data
- **If integrity is broken?**
  - The legitimate update file is replaced with a file containing malicious code, leading to subsequent malicious activities



Integrity

- **Methods to ensure integrity**
  - Authentication
    - Proof of your identity and conduct (e.g., login process)
    - Only authorized users can change the data
  - Hash function
    - Using hash functions for tamper detection

```
5416371cc0e990871746ddaac89f1a5e *ubuntu-16.04.6-desktop-amd64.iso
feefb18e7916c9a16bb09923ed98df64 *ubuntu-16.04.6-desktop-i386.iso
ac8a79a86a905ebdc3ef3f5dd16b7360 *ubuntu-16.04.6-server-amd64.iso
1817138b1a181507c5ebd5ec8a3f40ba *ubuntu-16.04.6-server-i386.iso
```

Availability (가용성)

- **Ensuring a service (or data/program) accessible**
  - Prohibiting an attacker from hindering computation
  - This guarantees that legitimate uses of the service remain possible

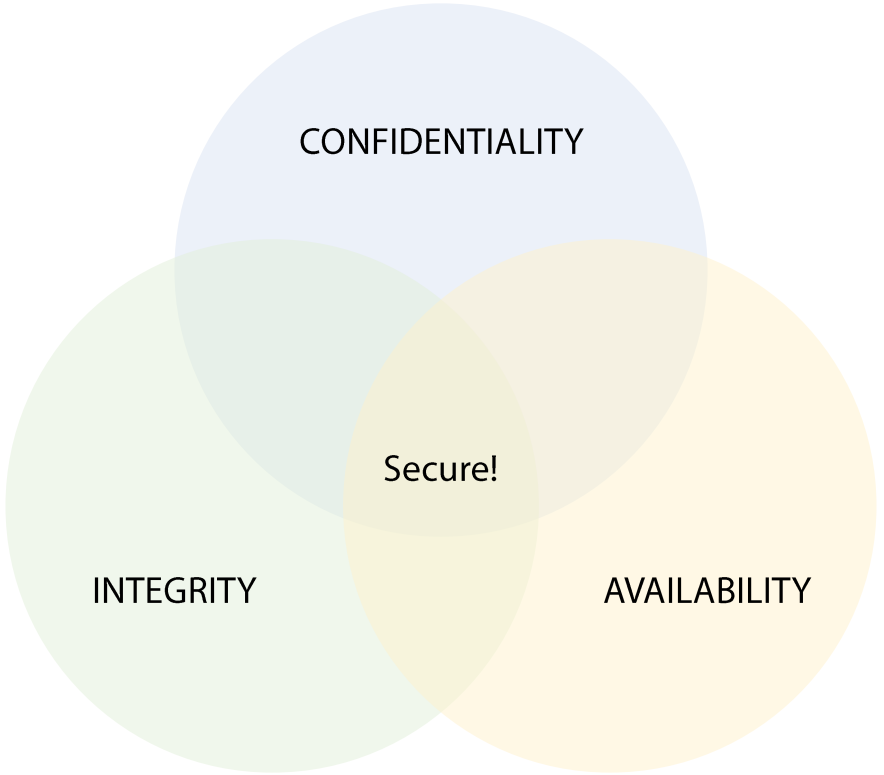
Availability

- **Ensuring a service (or data/program) accessible**
  - Prohibiting an attacker from hindering computation
  - This guarantees that legitimate uses of the service remain possible
- **If availability is broken?**
  - The legitimate user is unable to access the desired data
  - E.g., Denial of Service attacks (DoS)

Availability

- **Methods to ensure availability**
  - Data backup and replication
    - Replicating data across multiple locations to protect against issues at a single location
  - Error tolerance and recovery
    - Building services that are robust and capable of quickly recovering from errors
  - Monitoring
    - Monitoring resources and data

Relationship between CIA



Computer security vs Software security

- **Software security**
  - Focusing on the design, implementation, and operation of software systems, including the reliable enforcement of confidentiality, integrity and availability
  - Can be implemented with Security Development Lifecycles (SDL), Software Bill of Materials (SBOM), and Vulnerability Disclosure Programs (VDP)

Basic terms related to software security

- **Asset**
- **Vulnerability (vs Bug)**
- **Threat (vs Risk)**

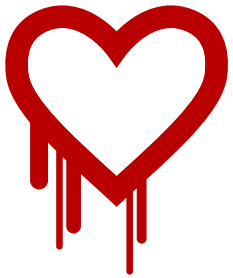
Asset

- **All valuable things that an organization possesses**
  - The things we want to protect
  - E.g., information, software, hardware, services

Vulnerability

- **Security weakness or flaw in a system, network, software, or any other entity that could be exploited by attackers**
  - This can compromise the CIA of the system or its data

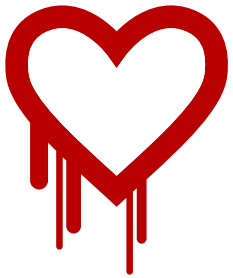
Vulnerability



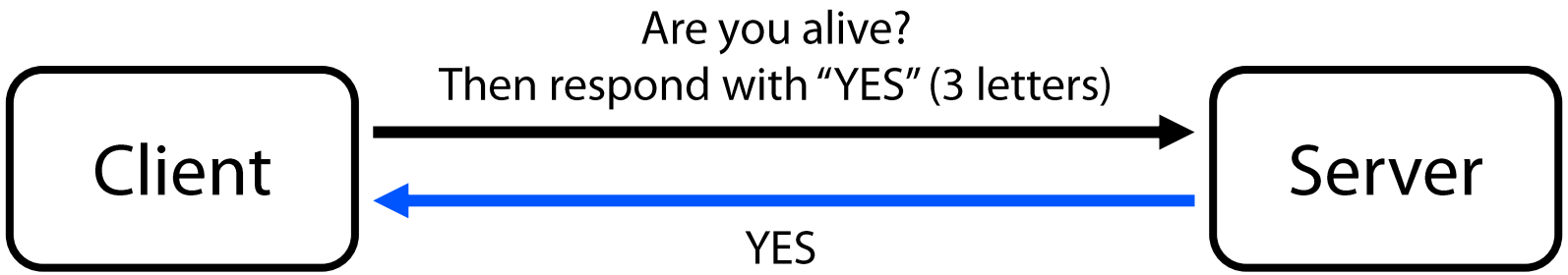
- **Example: HeartBleed vulnerability**
  - An extremely dangerous vulnerability discovered in OpenSSL*
  - This vulnerability embedded in the OpenSSL HeartBeat
    - HeartBeat used to exchange periodic signals between the server and the client to check for any issues or to maintain a stable connection

*OpenSSL: an encryption library that enables secure communication

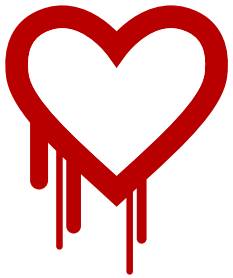
Vulnerability



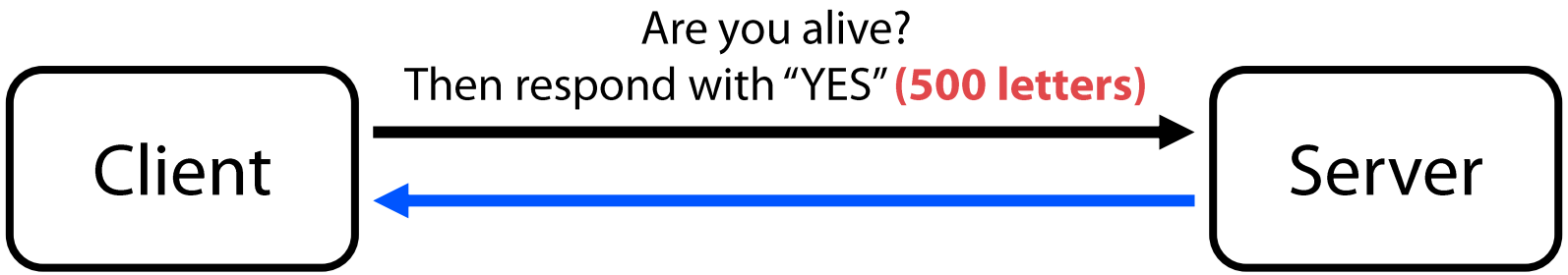
- **Example: HeartBleed vulnerability**
  - Normal case



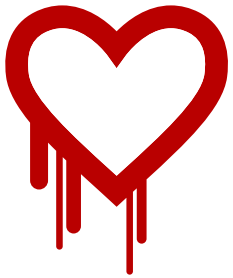
Vulnerability



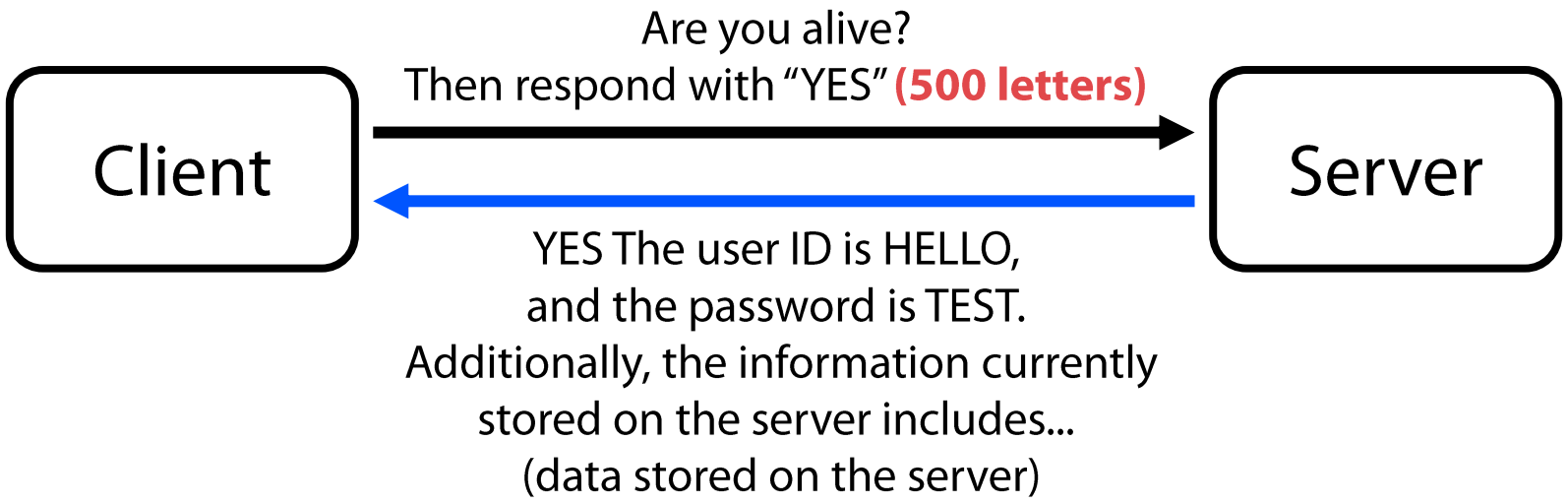
- **Example: HeartBleed vulnerability**
  - Vulnerable case



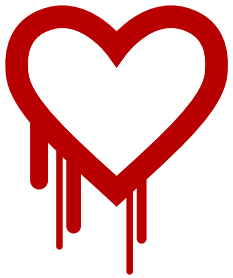
Vulnerability



- **Example: HeartBleed vulnerability**
  - Vulnerable case



Vulnerability



- **Example: HeartBleed vulnerability**
  - Vulnerable code / security patch

```
-       /* Read type and payload length first */
-       hbtype = *p++;
-       n2s(p, payload);
-       pl = p;
-
        if (s->msg_callback)
                s->msg_callback(0, s->version, TLS1_RT_HEARTBEAT,
                        &s->s3->rrec.data[0], s->s3->rrec.length,
                        s, s->msg_callback_arg);

+       /* Read type and payload length first */
+       if (1 + 2 + 16 > s->s3->rrec.length)
+               return 0; /* silently discard */
+       hbtype = *p++;
+       n2s(p, payload);
+       if (1 + 2 + payload + 16 > s->s3->rrec.length)
+               return 0; /* silently discard per RFC 6520 sec. 4 */
+       pl = p;
+
        if (hbtype == TLS1_HB_REQUEST)
                {
                unsigned char *buffer, *bp;
```

Vulnerability

- **Common Vulnerabilities and Exposures (CVEs)**
  - List of publicly known computer security vulnerabilities
  - More than 200,000 CVEs are managed
  - CVE MITRE (https://cve.mitre.org/), National Vulnerability Database (NVD)
  - CVE-YEAR-ID
    - E.g., CVE-2020-14147
      - Discovered in 2020

Vulnerability

CVE-2014-0160 Detail

Description

The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

Severity

CVSS Version 3.x | CVSS Version 2.0

CVSS 3.x Severity and Metrics:

NIST: NVD **Base Score:** 7.5 HIGH **Vector:** CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N

Weakness Enumeration

CWE-ID	CWE Name	Source
CWE-125	Out-of-bounds Read	NIST

Known Affected Software Configurations Switch to CPE 2.2

Configuration 1 (hide)

cpe:2.3:a:openssl:openssl:*:*:*:*:*:*:*:*	From (including)	Up to (excluding)
Show Matching CPE(s)	1.0.1	1.0.1g

Vulnerability vs Bug

- **Bug**
  - Errors, defects, or unexpected operation of software or programs
  - E.g., Type error

```
num1 = 10
num2 = ‘20’
result = num1 + num2 // TypeError
```

  - Bugs related to security issues are called vulnerabilities

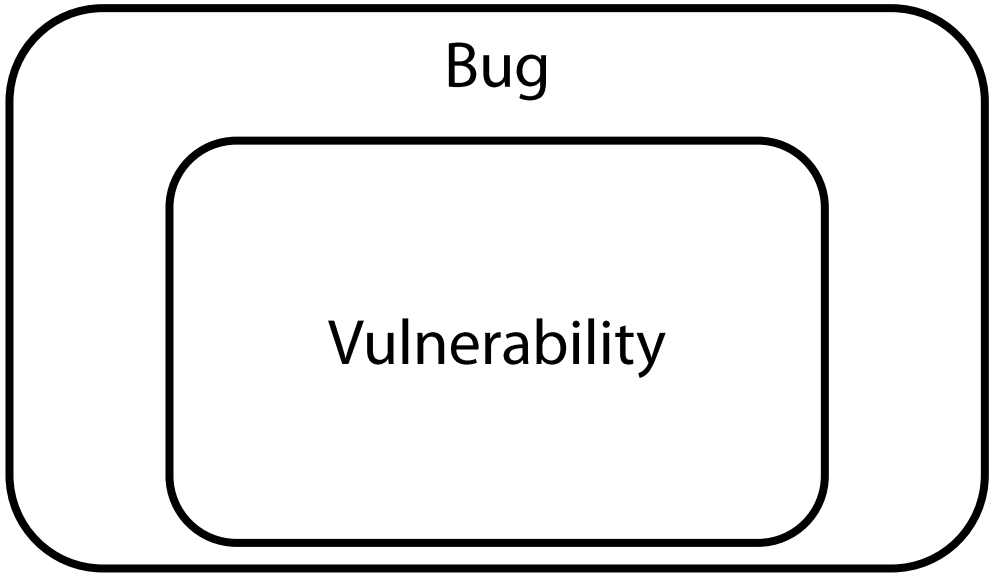
Vulnerability vs Bug

- **Bug**
  - Python code example that contains bugs but does not introduce vulnerabilities

```
def calculate_average(numbers):
    total = 0
    count = 0

    for number in numbers:
        total += number
        count += 1

    average = count / len(numbers)

    return average
```

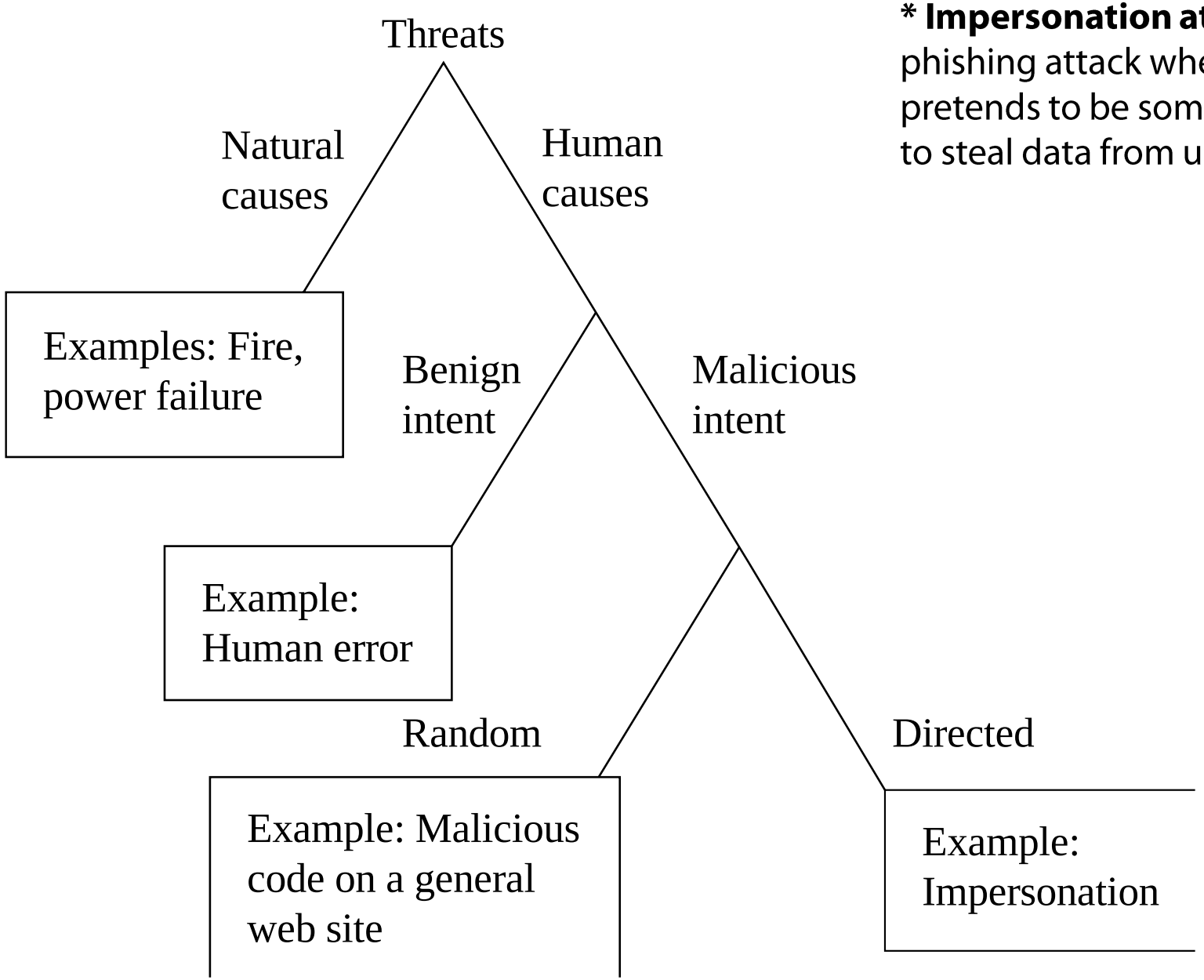
Vulnerability vs Bug

- **Bug**
  - Errors, defects, or unexpected operation of software or programs



Threat (위협)

- **The environment that presents the potential for loss or damage**
  - Potential danger or harmful events
  - E.g., The existence of vulnerabilities and hackers, lost hard drive



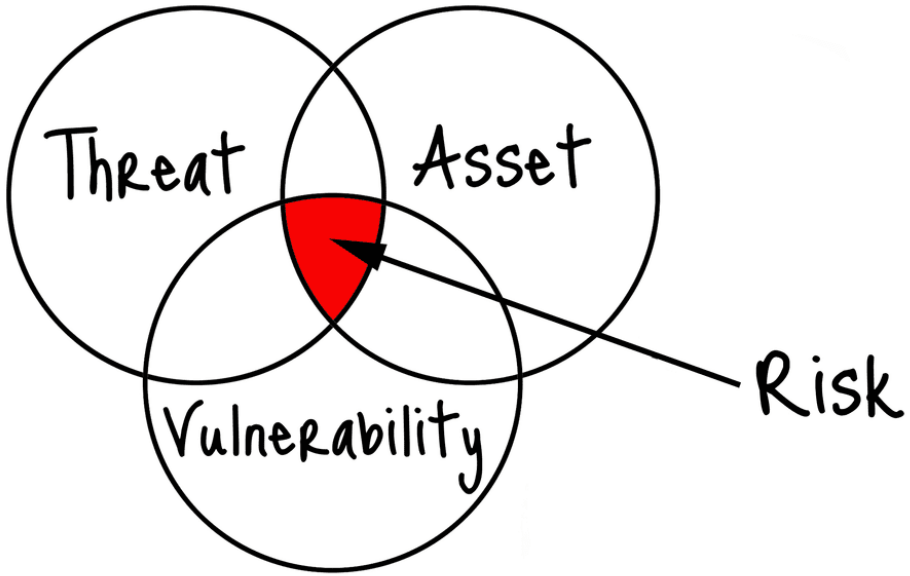
*** Impersonation attack:** a type of targeted phishing attack where a malicious actor pretends to be someone else or other entities to steal data from unsuspecting employees

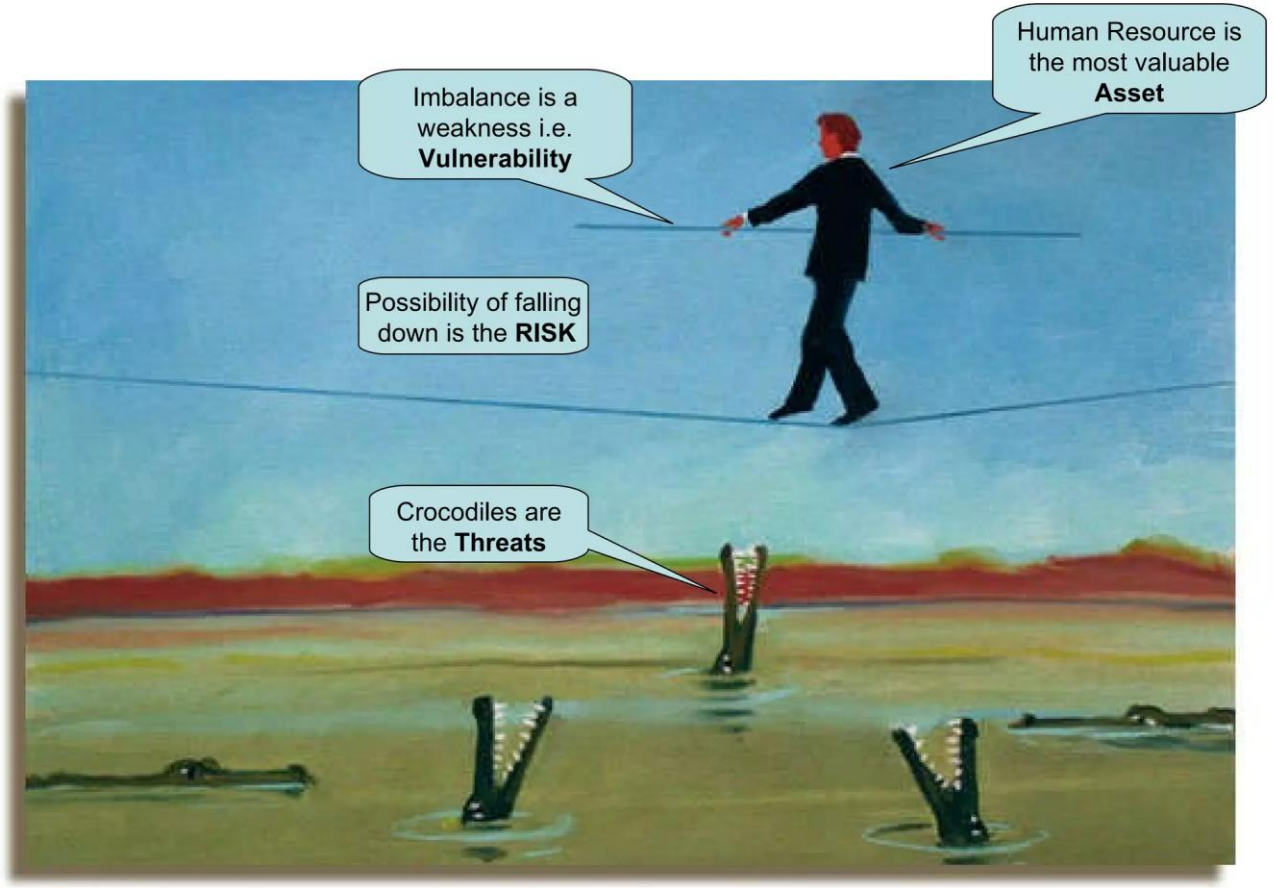
From Security in Computing, Fifth Edition, by Charles P. Pfleeger, et al. (ISBN: 9780134085043)

Threat (위협) vs Risk (위험)

- **Risk**
  - The likelihood of potential harm or loss resulting from the exploitation of vulnerabilities by threats
  - E.g., Information leakage due to vulnerability, attacks by hackers
- **Risk = Threat * Vulnerability * Assets**
  - If vulnerability does not exist, a threat cannot become a risk
  - Reducing risk (risk management) is one of the important issues in security
    - E.g., Identifying vulnerabilities and remediating them

Asset, Vulnerability, Threat, and Risk





https://www.slideshare.net/mfnaqvi/asset-vulnerability-threat-risk-control
https://sikkertgaute.medium.com/cybersecurity-risks-in-higher-education-part-1-background-assets-threat-events-and-threat-236365e9c3ea

Design principles for securing software

- **Eight principles enumerated by Saltzer and Schroeder**

Economy of Mechanism	Fail-safe Defaults	Complete Mediation	Open Design
Separation of Privilege	Least Privilege	Least Common Mechanism	Psychological Acceptability

J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, 1975. https://doi:org/10:1109/PROC:1975:9939

Design principles for securing software

1. **Economy of Mechanism**
   - Keep the design as simple and small as possible
   - Simple implementations are easier to manually audit
2. **Fail-safe Defaults**
   - Base access decisions on permission rather than exclusion
   - Malicious behaviors are difficult to enumerate and identify exhaustively

Design principles for securing software

3. Complete Mediation

- Every access to every object must be checked for authority

4. Open Design

- The design should not be secret
- This allows researchers and auditors to examine how security controls operate to ensure their correctness
- E.g., Open-source software itself is known to be highly secure (Google, etc.)

Design principles for securing software

5. Separation of Privilege

- Maintaining separation between roles or accounts with different permissions
- E.g., to not allow users with administrator privileges to also have regular user privileges
- This increases the security of the system by allowing multiple accounts or roles to have some privileges rather than one account having all privileges

6. Least Privilege

- Strengthen system security by granting only the minimum necessary privileges to each user or process
- E.g., if an operation needs to only read some information, it should not also be granted the privileges to write or delete this information

Design principles for securing software

7. Least Common Mechanism

- Minimize the amount of mechanism common to more than one user and depended on by all users
- Shared objects provide potentially dangerous channels for information flow and unintended interactions

8. Psychological Acceptability

- The human interface should be designed for ease of use
- The security control should be naturally usable so that users 'routinely and automatically' apply the protection

Next Lecture

- **Authentication**