사이버보안개론

Introduction to Cybersecurity

유준수 (Joon Soo Yoo, Ph.D.)

Fall 2026

Defense & Advanced Security Lab (DAS Lab)
Department of Advanced Defense Technology and Industry
Jeonbuk National University

Chapter 1. Overview

1.1 Computer Security Concepts

What Do We Protect,
and What Does “Secure” Mean?

Learning Objectives

이번 Section에서는 Computer Security의 가장 기본적인 개념과 용어를 학습한다.

Learning Goals

- Computer Security의 기본 목적을 설명할 수 있다.
- Confidentiality, Integrity, Availability를 구분할 수 있다.
- Authenticity와 Accountability의 의미를 이해한다.
- Security Impact를 Low, Moderate, High로 구분할 수 있다.
- Computer Security가 어려운 이유를 설명할 수 있다.
- Asset, Vulnerability, Threat, Attack, Risk의 관계를 이해한다.

1. Three Fundamental Questions

Computer Security를 이해하기 위해 먼저 세 가지 질문에서 시작한다.

1. What assets do we need to protect?
2. How are those assets threatened?
3. What can we do to counter those threats?

예를 들어 대학의 정보시스템을 생각해보자.

Student Records

⇓

Unauthorized Access / Modification / Service Failure

⇓

Access Control / Encryption / Backup / Monitoring

핵심

보안은 무엇을 보호할 것인지, 무엇으로부터 보호할 것인지, 그리고 어떻게 보호할 것인지의 문제이다.

2. What is Computer Security?

Computer Security는 정보뿐 아니라 Computer System 전체의 자원을 보호하는 것을 의미한다.

보호 대상에는 다음이 포함될 수 있다.

- Hardware
- Software
- Firmware
- Information / Data
- Communication Systems

Computer Security의 핵심 목표는 크게 세 가지이다.

Confidentiality　　Integrity　　Availability

핵심

Computer Security의 기본 목표는 정보와 시스템 자원의 Confidentiality, Integrity, Availability를 보호하는 것이다.

3. The CIA Triad

Computer Security의 세 가지 핵심 목표를 CIA Triad라고 한다.

CIA Triad

Security Goal	Main Question
Confidentiality	허가되지 않은 사람이 정보를 볼 수 있는가?
Integrity	정보와 시스템을 믿을 수 있는가?
Availability	필요할 때 시스템과 정보를 사용할 수 있는가?

Confidentiality + Integrity + Availability

핵심

CIA Triad는 이후 다양한 Security Problem을 분류하고 분석하는 가장 기본적인 Framework이다.

4. Confidentiality

Confidentiality는 허가되지 않은 사용자에게 정보가 공개되지 않도록 하는 것이다.

Confidentiality

Concept	Meaning
Data Confidentiality	Private 또는 Confidential Information이 Unauthorized Individual에게 공개되지 않도록 함
Privacy	개인과 관련된 정보가 어떻게 수집, 저장, 사용 및 공개되는지 통제할 수 있도록 함

Example

학생의 성적 정보를 다른 학생이 허가 없이 열람하였다.

⇒ Loss of Confidentiality

5. Integrity

Integrity는 정보와 시스템이 허가되지 않은 방식으로 변경되지 않도록 하는 것이다.

Integrity에는 두 가지 측면이 있다.

Integrity

Concept	Meaning
Data Integrity	Information과 Program이 Authorized Manner로만 변경됨
System Integrity	System이 Unauthorized Manipulation 없이 의도된 기능을 정상적으로 수행함

Example

병원 Database에서 환자의 알러지 정보가 악의적으로 변경되었다.

$\mathit{Allergy = Yes} \rightarrow \mathit{Allergy = No}$

$\Rightarrow$ Loss of Integrity

6. Data Integrity vs. System Integrity

Integrity는 단순히 “Data가 변했는가?”만을 의미하지 않는다.

Data Integrity

데이터가 허가된 방식으로만 변경되어야 한다.

예:

- Database Record
- Configuration File
- Program Code
- Command Message

System Integrity

시스템 자체가 의도한 기능을 정상적으로 수행해야 한다.

예:

- Server Operation
- Control System
- Robot Behavior
- Autonomous System

핵심

데이터가 정상이어도 시스템 동작이 조작되었다면 System Integrity가 침해될 수 있다.

7. Availability

Availability는 Authorized User가 필요할 때 시스템과 서비스를 사용할 수 있도록 하는 것이다.

Authorized User

⇓

Request

⇓

System / Service

⇓

Timely and Reliable Access

Availability가 깨지는 대표적인 상황은 다음과 같다.

- Server Failure
- Network Failure
- Resource Exhaustion
- Denial-of-Service Attack

핵심

정보가 유출되거나 변경되지 않았더라도 정상 사용자가 서비스를 사용할 수 없다면 Security Failure이다.

8. Understanding CIA with One System

하나의 학생 성적 관리 시스템을 생각해보자.

CIA Example

	Security Failure	Example
Confidentiality	Unauthorized Disclosure	다른 학생의 성적을 열람
Integrity	Unauthorized Modification	공격자가 성적을 변경
Availability	Service Disruption	학생이 성적 시스템에 접속할 수 없음

Same System

⇓

Different Security Objectives

⇓

Different Security Failures

핵심

하나의 Asset도 Confidentiality, Integrity, Availability 관점에서 각각 다르게 분석할 수 있다.

9. Beyond CIA

CIA Triad는 가장 기본적인 Security Objectives이지만, 그것만으로 모든 Security Requirement를 설명하기는 어렵다.

추가적으로 중요한 두 가지 개념이 있다.

Authenticity

정보나 사용자가 "진짜인가?"를 확인한다.

예:

- Is this really Alice?
- Did Alice send this message?
- Is this a trusted source?

Accountability

행동을 수행한 주체를 나중에 추적할 수 있어야 한다.

예:

- Logging
- Audit Trail
- Forensic Analysis
- Nonrepudiation

10. Authenticity

Authenticity는 사용자, 메시지 또는 데이터의 출처가 신뢰할 수 있는지 확인하는 것이다.

예를 들어 다음 메시지를 생각해보자.

From: Administrator

Delete all user accounts

중요한 질문은 다음과 같다.

- 정말 Administrator가 보낸 것인가?
- 메시지가 신뢰할 수 있는 Source에서 왔는가?
- 전송 과정에서 위조된 것은 아닌가?

핵심

Authenticity는 단순히 데이터의 내용이 아니라, 그 데이터와 주체의 "진위"를 확인하는 Security Objective이다.

11. Accountability

완벽하게 안전한 시스템을 만드는 것은 현실적으로 어렵다.

따라서 Security Incident가 발생했을 때 무슨 일이 있었는지 추적할 수 있어야 한다.

User Action

⇓

Logging / Audit

⇓

Security Incident

⇓

Trace the Responsible Entity

Accountability는 다음과 연결된다.

- Nonrepudiation
- Deterrence
- Fault Isolation
- Intrusion Detection
- Forensic Analysis

12. Security Impact Levels

모든 Security Breach가 같은 수준의 피해를 발생시키는 것은 아니다.

FIPS 199에서는 Security Loss의 영향을 세 단계로 분류한다.

Impact Levels

Level	Impact
Low	조직이나 개인에게 제한적인 피해가 발생
Moderate	조직의 기능이나 자산에 심각한 피해가 발생
High	조직의 핵심 기능, 자산 또는 사람에게 심각하거나 치명적인 피해가 발생

핵심

보호 수준은 모든 Asset에 동일하게 적용하는 것이 아니라, Security Failure가 가져올 Impact에 따라 결정해야 한다.

13. Example: Different Security Requirements

Asset의 종류와 목적에 따라 필요한 Security Level도 달라질 수 있다.

Examples

Asset	Main Requirement	Impact
Public Faculty Directory	Confidentiality	Low
Student Grade Information	Confidentiality	High
Patient Allergy Information	Integrity	High
University Public Website	Availability	Moderate
Critical Authentication Service	Availability	High

중요

"어떤 Security Goal이 가장 중요한가?"는 시스템과 Asset에 따라 달라진다.

14. Why is Computer Security Difficult?

Security Requirement 자체는 단순해 보인다.

Confidentiality

Integrity

Authentication

Availability

하지만 이를 실제 시스템에서 보장하는 것은 매우 어렵다.

그 이유는 다음과 같다.

- 공격자는 예상하지 못한 Weakness를 찾는다.
- Security Mechanism은 생각보다 복잡하다.
- Mechanism을 어디에 적용할지도 결정해야 한다.
- Key와 같은 Secret Information도 관리해야 한다.
- Security와 Performance / Usability가 충돌할 수 있다.

핵심

Security Goal은 간단해 보여도, 이를 만족시키는 Security Mechanism은 복잡하다.

15. Attacker vs. Defender

Computer Security는 공격자와 방어자 사이의 지속적인 경쟁이라고 볼 수 있다.

Attacker

공격자는 성공하기 위해 하나의 Weakness만 찾아도 된다.

- One Vulnerability
- One Misconfiguration
- One Weak Password
- One Programming Error

Defender

방어자는 가능한 모든 공격 경로를 고려해야 한다.

- Multiple Components
- Multiple Users
- Multiple Networks
- Multiple Attack Paths

Attacker: Find **one** weakness

Defender: Protect **all** important paths

핵심

공격자는 하나의 취약점만 찾아도 성공할 수 있지만, 방어자는 시스템 전체를 고려해야 한다.

16. Security Is More Than an Algorithm

좋은 Security Algorithm 하나를 사용하는 것만으로 안전한 시스템이 되는 것은 아니다.

예를 들어 강력한 Encryption Algorithm을 사용하더라도 다음 문제가 발생할 수 있다.

- Encryption Key가 유출될 수 있다.
- Key Distribution이 잘못될 수 있다.
- Implementation에 취약점이 있을 수 있다.
- Authentication이 우회될 수 있다.
- Protocol Design에 문제가 있을 수 있다.
- 사용자가 Security Policy를 따르지 않을 수 있다.

$$\text{Strong Algorithm} \neq \text{Secure System}$$

핵심

Security는 Algorithm뿐 아니라 Implementation, Key Management, Protocol, User, System Design 전체의 문제이다.

17. Where Should Security Be Applied?

Security Mechanism을 설계한 뒤에는 어디에 적용할지도 결정해야 한다.

예를 들어 Network Security Mechanism은 다양한 위치에 적용될 수 있다.

Application

⇓

Transport Layer

⇓

Network Layer

⇓

Link Layer

⇓

Physical Network

중요한 질문은 다음과 같다.

- 어느 Layer에서 보호할 것인가?
- 어느 Network Point에 Security Device를 둘 것인가?
- End-to-End로 보호할 것인가?
- 특정 Component만 보호할 것인가?

18. Security Should Not Be an Afterthought

Security는 시스템을 모두 만든 뒤 마지막에 추가하는 기능이 되어서는 안 된다.

잘못된 접근은 다음과 같다.

System Design

⇓

Implementation

⇓

Deployment

⇓

“Now, add Security”

더 좋은 접근은 다음과 같다.

Requirements → Design → Implementation → Deployment

19. Security vs. Usability and Performance

강한 Security가 항상 좋은 User Experience를 의미하지는 않는다.

Security Mechanism은 다음과 같은 비용을 만들 수 있다.

- Additional Authentication
- Encryption Overhead
- Access Restrictions
- Logging Overhead
- Monitoring Cost
- Additional User Procedures

따라서 실제 시스템에서는 다음 균형이 필요하다.

$$\text{Security} \Longleftrightarrow \text{Performance} \Longleftrightarrow \text{Usability} \Longleftrightarrow \text{Cost}$$

20. A Model for Computer Security

Computer Security를 분석하기 위해 몇 가지 기본 용어를 정의한다.

Asset

⇓

Vulnerability

⇓

Threat

⇓

Attack

⇓

Security Consequence

그리고 이를 줄이기 위해

$$\text{Countermeasure} \Longrightarrow \text{Reduced Risk}$$

핵심

Security Problem을 분석할 때는 Asset, Vulnerability, Threat, Attack, Countermeasure, Risk를 구분해야 한다.

21. What is an Asset?

Asset은 Owner가 보호하고자 하는 System Resource이다.

Computer System의 Asset은 크게 다음과 같이 나눌 수 있다.

System Assets

Asset Type	Examples
Hardware	Computer, Storage Device, Network Equipment
Software	Operating System, Applications, Utilities
Data	Files, Databases, Password Files, Configuration
Communication	Network Links, Routers, Communication Services

핵심

보안에서 보호 대상은 Data만이 아니다. Hardware, Software, Network, Service도 중요한 Asset이다.

22. How Can an Asset Be Compromised?

Asset에는 공격자가 이용할 수 있는 Vulnerability가 존재할 수 있다.

공격자가 이러한 Vulnerability를 이용하면, Asset은 크게 다음과 같은 상태에 놓일 수 있다.

Possible Security Consequences

Problem	Meaning	Security Loss
Corrupted	정보가 잘못 변경되거나 시스템이 잘못 동작	Integrity
Leaky	Unauthorized User에게 정보가 노출	Confidentiality
Unavailable	System 또는 Service를 정상적으로 사용할 수 없음	Availability

핵심

Vulnerability는 Asset에 존재하는 약점이고, 그 약점이 공격에 이용되면 CIA의 손실로 이어질 수 있다.

23. What is a Vulnerability?

Vulnerability는 시스템의 Design, Implementation, Operation 또는 Management에 존재하는 Weakness이다.

예를 들어 Web Server를 생각해보자.

- SQL Injection Bug
- Weak Password
- Unpatched Software
- Incorrect Access Control
- Exposed Network Port
- Default Configuration

Vulnerability = A weakness that **can be exploited**

핵심

Vulnerability는 아직 공격 자체가 아니라, 공격자가 이용할 수 있는 시스템의 약점이다.

24. What is a Threat?

Threat는 Vulnerability를 이용하여 Security Harm을 발생시킬 수 있는 잠재적인 위험이다.

예를 들어 관리자 계정의 비밀번호가 너무 쉽게 설정되어 있다고 생각해보자.

Administrator Account

⇓

Weak Password (*1234*)

⇓

Attacker may guess the password

⇓

Unauthorized Login may be possible

아직 공격자가 실제로 로그인하지 않았더라도, 약한 비밀번호 때문에 Unauthorized Login이 가능하다는 Threat가 존재한다.

핵심

Vulnerability는 시스템에 존재하는 약점이고, Threat는 그 약점으로 인해 발생할 수 있는 잠재적인 위험이다. 공격자가 이를 실제 행동으로 수행하면 Attack이 된다.

25. Threat Agent and Adversary

Threat Agent 또는 Adversary는 공격을 수행하거나 시스템에 위협이 되는 Entity이다.

Threat Agent는 다양한 형태일 수 있다.

- External Hacker
- Malicious Insider
- Organized Criminal Group
- Competitor
- Hostile Organization
- Compromised Computer or Malware

Threat Agent

⇓

Exploits Vulnerability

⇓

Performs Attack

핵심

Threat는 위험의 가능성을 의미하고, Threat Agent는 그 위험을 실제로 만들어낼 수 있는 주체이다.

26. What is an Attack?

Attack은 시스템의 Security Policy를 위반하기 위해 수행되는 의도적인 행위이다.

Threat

Potential danger

⇓

Threat Agent acts

⇓

Attack

Actual attempt

예를 들어,

- Vulnerability: SQL Injection Bug
- Threat: 공격자가 Database를 조작할 가능성
- Attack: 실제 SQL Injection Payload 전송

핵심

Threat는 가능성이고, Attack은 그 가능성이 실제 행동으로 실행된 것이다.

27. Active vs. Passive Attack

Attack은 시스템에 영향을 주는 방식에 따라 Active와 Passive로 나눌 수 있다.

Active Attack

System Resource 또는 System Operation을 변경한다.

예:

- Data Modification
- Command Injection
- Malware Execution
- Denial of Service

Passive Attack

System Resource를 변경하지 않고 정보를 관찰하거나 수집한다.

예:

- Eavesdropping
- Packet Sniffing
- Traffic Observation
- Information Collection

핵심

Active Attack은 시스템을 변화시키고, Passive Attack은 주로 정보를 관찰한다.

28. Inside vs. Outside Attack

Attack은 공격의 Origin에 따라 Inside와 Outside로도 구분할 수 있다.

Inside Attack

Security Perimeter 내부의 Authorized Entity가 권한을 악용한다.

예:

- Employee
- Administrator
- Authorized User
- Contractor

Outside Attack

Security Perimeter 밖의 Unauthorized Entity가 공격한다.

예:

- Internet Attacker
- Remote Hacker
- Botnet
- External Organization

핵심

Inside/Outside는 공격자의 위치와 권한을 기준으로 하는 분류이고, Active/Passive와는 별개의 분류이다.

29. Attack Classifications Can Be Combined

Active/Passive와 Inside/Outside는 서로 다른 기준의 분류이다.

따라서 하나의 공격이 두 분류에 동시에 해당할 수 있다.

Examples

Example	Origin	Behavior
Employee modifies database	Inside	Active
Employee reads confidential files	Inside	Passive
Remote attacker injects commands	Outside	Active
Remote attacker sniffs traffic	Outside	Passive

핵심

Attack은 하나의 기준으로만 분류하지 않는다. 공격의 Origin과 Behavior를 각각 분석할 수 있다.

30. What is a Countermeasure?

Countermeasure는 Threat, Vulnerability 또는 Attack을 줄이기 위해 적용하는 대응 수단이다.

Countermeasure의 목적은 크게 세 가지로 볼 수 있다.

Prevent

⇓

Detect

⇓

Recover

예를 들어,

- Firewall
- Access Control
- Encryption
- Authentication
- Intrusion Detection
- Backup and Recovery
- Security Monitoring

핵심

완벽한 Prevention이 불가능할 수 있으므로, Detection과 Recovery도 Security의 중요한 일부이다.

31. Countermeasures Are Not Perfect

Countermeasure를 적용한다고 해서 모든 Vulnerability가 사라지는 것은 아니다.

Original Vulnerability

⇓

Countermeasure

⇓

Reduced Vulnerability

⇓

Residual Vulnerability

또한 Countermeasure 자체가 새로운 Vulnerability를 만들 수도 있다.

예:

- Complex Authentication → User Workaround
- Security Software → New Software Bugs
- Centralized Security Server → Single Point of Failure

중요

Security Mechanism도 시스템의 일부이므로 그 자체가 새로운 Attack Surface가 될 수 있다.

32. What is Risk?

Risk는 특정 Threat가 Vulnerability를 이용하여 실제 피해를 발생시킬 가능성을 의미한다.

개념적으로는 다음과 같이 생각할 수 있다.

$$\text{Risk} \approx \text{Likelihood} \times \text{Impact}$$

즉 Risk를 판단할 때는 두 가지를 함께 본다.

- 공격이 실제로 발생할 가능성이 얼마나 높은가?
- 공격이 성공했을 때 피해가 얼마나 큰가?

Example

발생 가능성이 낮더라도 사람의 생명과 연결된 시스템이라면 Risk가 매우 중요할 수 있다.

33. Residual Risk

Countermeasure를 적용한 뒤에도 일정 수준의 Risk는 남을 수 있다.

Initial Risk

⇓

Apply Countermeasures

⇓

Risk Reduced

⇓

Residual Risk

현실적인 Security의 목표는

$$\text{Risk} = 0$$

을 만드는 것이 아니라,

$$\text{Risk} \rightarrow \text{Acceptable Level}$$

로 줄이는 것이다.

34. Putting the Concepts Together

지금까지의 개념을 하나의 예제로 연결해보자.

Example: Administrator Account

Concept	Example
Asset	Administrator Account and Protected Data
Vulnerability	Weak Password (*1234*)
Threat	Attacker may guess the password and gain unauthorized access
Adversary	Unauthorized External User
Attack	Attacker actually enters *1234* and logs in
Consequence	Unauthorized Access to Protected Data
Countermeasure	Strong Password Policy, MFA, Login Attempt Limiting
Residual Risk	Password theft, user mistakes, or unknown vulnerabilities may remain

핵심

Vulnerability가 존재하면 이를 이용한 Threat가 발생할 수 있고, Adversary가 이를 실제로 수행하면 Attack이 된다. Countermeasure는 이러한 Risk를 줄이지만 완전히 제거하지는 못할 수 있다.

35. Security Concept Flow

Computer Security의 기본 흐름을 하나의 구조로 정리하면 다음과 같다.

Owner

⇓

Values

⇓

Asset

⇓

Vulnerability

⇓

Threat

⇓

Attack

⇓

Possible Loss

⇓

Risk

⇓

Countermeasure

⇓

Reduced Risk

36. From CIA to Security Analysis

Section 1.1의 개념은 서로 독립적인 것이 아니다.

CIA

What security property should be protected?

⇓

Asset

What should be protected?

⇓

Vulnerability / Threat / Attack

How can security be violated?

⇓

Countermeasure

How can the risk be reduced?

핵심

CIA는 Security Goal을 정의하고, Asset–Threat–Countermeasure 구조는 Security Problem을 분석하는 방법을 제공한다.

37. Security Thinking

새로운 시스템을 볼 때 다음 질문을 반복하면 된다.

1. What are the important assets?
2. Which CIA properties matter for each asset?
3. What vulnerabilities exist?
4. Who are the possible adversaries?
5. What attacks are possible?
6. What would be the impact?
7. What countermeasures can reduce the risk?
8. What residual risk remains?

Security Insight

Security를 공부한다는 것은 공격 방법을 외우는 것이 아니라, 시스템을 이러한 질문으로 분석하는 사고방식을 배우는 것이다.

38. Example: Applying the Model to a Drone

같은 Security Model은 Cyber-Physical System에도 적용할 수 있다.

Drone Security Example

Concept	Example
Asset	GPS Position, Flight Command, Sensor Data
Security Goal	Integrity, Availability, Authenticity
Vulnerability	Unauthenticated Communication
Threat	Attacker may inject false command
Attack	Fake Command Injection
Impact	Wrong Flight Path or Mission Failure
Countermeasure	Authentication, Access Control, Anomaly Detection

1.2 Threats and Attacks

How Can Security Be Violated?

Threat Consequences and Attack Types

1. From Security Goals to Security Violations

앞에서는 Security Goal과 Threat, Attack의 기본 개념을 살펴보았다.

이제 실제로 어떤 종류의 Security Consequence가 발생할 수 있는지 살펴본다.

Confidentiality

Integrity

Availability

⇓

How can these security properties be violated?

주요 Threat Consequence를 네 가지로 분류한다.

- Unauthorized Disclosure
- Deception
- Disruption
- Usurpation

2. Four Major Threat Consequences

Threat Consequences

Threat Consequence	Meaning	Related Goal
Unauthorized Disclosure	Unauthorized Entity가 Sensitive Data에 접근	Confidentiality
Deception	Authorized Entity가 False Data를 True Data라고 믿게 됨	Integrity
Disruption	System Service와 Function의 정상 동작이 방해되거나 중단됨	Availability / Integrity
Usurpation	Unauthorized Entity가 System Resource나 Function의 Control을 획득	Integrity

핵심

CIA의 손실은 실제 시스템에서 여러 형태의 Threat Consequence로 나타날 수 있다.

3. Unauthorized Disclosure

Unauthorized Disclosure는 허가되지 않은 Entity가 Sensitive Data에 접근하는 것이다.

이는 Confidentiality에 대한 Threat이다.

Attack Types

Attack	Meaning
Exposure	Sensitive Data가 Unauthorized Entity에게 직접 노출됨
Interception	전송 중인 Sensitive Data를 Unauthorized Entity가 가로챔
Inference	직접 데이터를 보지 않고 주변 정보로 Sensitive Information을 추론
Intrusion	Security Protection을 우회하여 Sensitive Data에 접근

핵심

Unauthorized Disclosure의 핵심은 “보면 안 되는 정보가 Unauthorized Entity에게 알려지는 것”이다.

4. Example: Unauthorized Disclosure

각 Attack Type을 간단한 예제로 보면 다음과 같다.

- **Exposure**
  학생 개인정보 파일이 실수로 Public Website에 공개됨
- **Interception**
  공격자가 Network Traffic을 가로채 Password나 Message를 읽음
- **Inference**
  암호화된 Traffic의 양과 통신 패턴을 보고 중요한 활동을 추론함
- **Intrusion**
  공격자가 Access Control을 우회하여 Private File에 접근함

핵심

결과는 모두 Confidentiality Loss이지만, 그 정보가 노출되는 방식은 서로 다르다.

5. Deception

Deception은 Authorized Entity가 False Data를 True Data라고 믿게 만드는 것이다.

이는 주로 Data Integrity와 System Integrity에 대한 Threat이다.

Attack Types

Attack	Meaning
Masquerade	Unauthorized Entity가 Authorized Entity인 것처럼 행동
Falsification	정상 데이터를 변경하거나 False Data를 생성
Repudiation	자신이 수행한 Action에 대한 책임을 거짓으로 부인

핵심

Deception의 핵심은 “가짜를 진짜라고 믿게 만드는 것”이다.

6. Example: Deception

- **Masquerade**
  공격자가 탈취한 ID와 Password로 정상 사용자처럼 로그인
- **Falsification**
  학생이 자신의 Grade를 *B*에서 *A+*로 변경
- **Repudiation**
  사용자가 실제로 명령을 보낸 뒤
  “나는 그런 명령을 보낸 적이 없다”
  라고 부인

핵심

Deception은 Identity, Data, Action의 진위를 속이는 공격과 연결된다.

7. Disruption

Disruption은 System Service나 Function의 정상적인 동작을 방해하거나 중단시키는 것이다.

주로 Availability와 System Integrity에 영향을 준다.

Attack Types

Attack	Meaning
Incapacitation	System Component를 비활성화하여 정상 동작을 불가능하게 함
Corruption	System Function이나 Data를 변경하여 의도하지 않은 동작을 유발
Obstruction	System Operation을 방해하여 Service Delivery를 어렵게 함

핵심

Disruption의 핵심은 “시스템이 정상적으로 동작하지 못하게 만드는 것”이다.

8. Example: Disruption

- **Incapacitation**
  Malware가 Server를 Crash시켜 서비스를 완전히 중단
- **Corruption**
  공격자가 System Configuration을 변경하여 System이 잘못 동작하도록 만듦
- **Obstruction**
  과도한 Traffic을 발생시켜 정상 사용자가 Service에 접근하지 못하게 함

Excessive Traffic

⇓

System Overload

⇓

Normal Service Unavailable

9. Usurpation

Usurpation은 Unauthorized Entity가 System Resource나 Function의 Control을 획득하는 것이다.

이는 주로 System Integrity에 대한 Threat이다.

Attack Types

Attack	Meaning
Misappropriation	System Resource를 Unauthorized하게 자신의 목적으로 사용
Misuse	System Component나 Function을 Security에 해로운 방식으로 사용

Example

공격자가 다른 사용자의 Computer를 감염시켜 DDoS Bot으로 사용한다.

⇒ CPU, Network, OS Resource를 Unauthorized하게 사용하는 Misappropriation

10. Putting the Threat Consequences Together

Summary

Consequence	Simple Meaning	Representative Attacks
Unauthorized Disclosure	정보를 보면 안 되는 사람이 봄	Exposure, Interception, Inference, Intrusion
Deception	가짜를 진짜라고 믿게 만듦	Masquerade, Falsification, Repudiation
Disruption	System의 정상 동작을 방해	Incapacitation, Corruption, Obstruction
Usurpation	System의 Resource나 Control을 빼앗음	Misappropriation, Misuse

핵심

Attack의 구체적인 방법은 다양하지만, 그 결과는 Confidentiality, Integrity, Availability의 손실로 연결된다.

11. One More View: Passive vs. Active

Network Attack은 크게 Passive와 Active로도 나눌 수 있다.

Passive Attack

System Resource를 변경하지 않고 정보를 관찰한다.

- Message Eavesdropping
- Traffic Analysis

Detection이 어렵기 때문에 주로 Prevention이 중요하다.

Active Attack

Data나 System Operation을 실제로 변경하거나 방해한다.

- Replay
- Masquerade
- Message Modification
- Denial of Service

Detection과 Recovery가 중요하다.

1.4 Security Design Principles

How Should We Design a Secure System?

Fundamental Principles for Security

1. Why Do We Need Security Design Principles?

완벽하게 모든 Security Flaw를 제거하는 설계 방법은 존재하지 않는다.

따라서 Secure System을 설계할 때 따라야 할 기본적인 Design Principle이 필요하다.

Security Requirements

⇓

Security Mechanisms

⇓

How should we design them?

⇓

Security Design Principles

핵심

Security Design Principle은 특정 Security Product가 아니라, 안전한 시스템을 설계하기 위한 기본적인 사고 원칙이다.

2. Fundamental Security Design Principles

대표적인 Security Design Principle은 다음과 같다.

- Economy of Mechanism
- Fail-Safe Defaults
- Complete Mediation
- Open Design
- Separation of Privilege
- Least Privilege
- Least Common Mechanism
- Psychological Acceptability
- Isolation
- Encapsulation
- Modularity
- Layering
- Least Astonishment

핵심

각 원칙은 서로 독립적인 규칙이라기보다, Security Flaw와 Attack Surface를 줄이기 위한 상호 보완적인 설계 원칙이다.

3. Economy of Mechanism

Security Mechanism은 가능한 한 작고 단순하게 설계해야 한다.

복잡한 시스템에서는

- 더 많은 Bug가 발생할 수 있고,
- 예상하지 못한 Interaction이 생길 수 있으며,
- Security Verification이 어려워지고,
- 유지보수와 Configuration도 복잡해진다.

More Complexity

⇓

More Possible Mistakes

⇓

More Potential Vulnerabilities

핵심

불필요한 Complexity를 줄이면 Security Mechanism을 이해하고 검증하기 쉬워진다.

4. Example: Economy of Mechanism

간단한 Authentication System을 생각해보자.

User Credential

⇓

Authentication

⇓

Allow / Deny

여기에 계속 예외적인 기능이 추가된다고 생각해보자.

- Legacy Login
- Temporary Login
- Administrator Bypass
- Compatibility Mode
- Multiple Recovery Paths

기능이 많아질수록 예상하지 못한 Security Bypass가 발생할 가능성도 증가한다.

핵심

Security에서 불필요한 Complexity는 그 자체로 Risk를 증가시킬 수 있다.

5. Fail-Safe Defaults

Access Control의 기본 상태는 Permission이 아니라 Denial이어야 한다.

Default Allow

기본적으로 모두 허용하고 금지할 대상을 지정한다.

Default = Allow
Block Alice
Block Bob
Block Eve

Default Deny

기본적으로 모두 거부하고 허용할 대상을 지정한다.

Default = Deny
Allow Professor
Allow Administrator

핵심

명시적으로 허용되지 않은 Access는 기본적으로 거부되어야 한다.

6. Why is Default Deny Safer?

설정이나 구현에서 실수가 발생한다고 생각해보자.

Failure Behavior

Design	Possible Failure
Default Allow	차단해야 할 사용자를 빠뜨리면 Unauthorized Access가 허용될 수 있음
Default Deny	허용해야 할 사용자를 빠뜨리면 Authorized Access가 거부됨

두 경우 모두 오류이지만,

Unauthorized Access

vs.

Temporary Access Failure

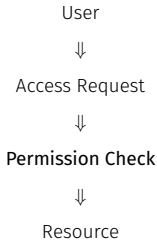
후자가 일반적으로 Security 관점에서는 더 안전한 Failure이다.

핵심

Fail-Safe Defaults는 시스템이 실패하더라도 가능한 한 안전한 상태로 실패하도록 설계하는 원칙이다.

7. Complete Mediation

모든 Resource Access는 Access Control Mechanism에 의해 확인되어야 한다.



중요한 것은 처음 한 번만 확인하는 것이 아니라 Access가 발생할 때 권한이 유효한지 확인하는 것이다.

핵심

"Previously allowed"라는 이유만으로 현재의 Access도 자동으로 허용해서는 안 된다.

8. Example: Complete Mediation

사용자가 File을 읽고 있다고 생각해보자.

10:00 Permission Check → Allow

10:01 File Access

10:02 File Access

10:03 Administrator revokes permission

10:04 File Access → ?

Complete Mediation 원칙에서는 변경된 Permission이 이후 Access에 반영되어야 한다.

하지만 실제 시스템에서는 Performance를 위해 Cache나 Session을 사용하기 때문에 완전한 적용이 어려울 수 있다.

Trade-off

Complete Mediation은 Security를 강화하지만, 모든 Access를 반복적으로 검사하면 Performance Overhead가 발생할 수 있다.

9. Open Design

Security Mechanism의 안전성은 Design 자체를 숨기는 것에 의존해서는 안 된다.

대표적인 예가 Cryptography이다.

Encryption Algorithm

Public

Encryption Key

Secret

Algorithm이 공개되어 있어도 Key 없이는 Security가 깨지지 않아야 한다.

핵심

Security by Obscurity에 의존하기보다, 공개적인 검토에도 안전한 Design을 사용해야 한다.

10. Separation of Privilege

중요한 Resource에 접근하기 위해 하나 이상의 조건을 요구할 수 있다.

대표적인 예가 Multi-Factor Authentication이다.

Password

+

Security Token / Smartphone

⇓

Access Granted

하나의 Credential이 유출되어도 즉시 중요한 Resource에 접근하지 못하게 한다.

핵심

중요한 권한을 하나의 조건이나 하나의 Privilege에만 의존하지 않는다.

11. Least Privilege

User와 Process는 자신의 작업에 필요한 최소한의 Privilege만 가져야 한다.

예를 들어 학생에게 필요한 권한이

Read Own Grade

뿐이라면,

다음 권한까지 줄 필요는 없다.

- Read All Students' Grades
- Modify Grades
- Delete Grades
- Manage User Accounts

핵심

필요하지 않은 Privilege를 제거하면, Account나 Process가 공격당했을 때 발생할 수 있는 피해도 줄어든다.

12. Least Privilege Over Time

Least Privilege는 권한의 종류뿐 아니라 권한을 가지는 시간에도 적용된다.

Normal Work

⇓

Normal Privilege

Administrative Task

⇓

Temporary Elevated Privilege

Task Finished

⇓

Privilege Removed

Administrator라고 해서 항상 높은 Privilege로 작업할 필요는 없다.

핵심

Privilege는 필요한 사용자에게, 필요한 작업을 위해, 필요한 시간 동안만 제공하는 것이 바람직하다.

13. Least Common Mechanism

여러 User나 Process가 공통으로 사용하는 Mechanism과 Resource를 가능한 줄여야 한다.

Shared

User A

$\Downarrow$

Shared Resource

$\Uparrow$

User B

한 사용자의 행동이 다른 사용자에게 영향을 줄 가능성이 있다.

Separated

User A $\rightarrow$ Resource A

User B $\rightarrow$ Resource B

불필요한 Shared Path를 줄일 수 있다.

핵심

불필요한 Sharing을 줄이면 사용자 사이의 예상하지 못한 Security Interaction도 줄어든다.

14. Psychological Acceptability

Security Mechanism은 사용자가 실제로 사용할 수 있을 정도로 편리해야 한다.

보안 정책이 지나치게 복잡하거나 불편하면

Complex Security Policy

⇓

User Frustration

⇓

Security Workaround

⇓

Weaker Security

예를 들어 지나치게 복잡한 Password Policy 때문에 사용자가 Password를 종이에 적어둘 수도 있다.

핵심

사용자가 Security Mechanism을 우회하지 않도록 Security와 Usability를 함께 고려해야 한다.

15. Isolation

중요한 Resource와 Process는 다른 시스템이나 사용자로부터 분리해야 한다.

Isolation은 여러 수준에서 적용될 수 있다.

- Public System과 Critical System 분리
- User 간 Process / Memory / File 분리
- Security Mechanism과 Key를 일반 Application에서 분리

Public System

Isolation Boundary

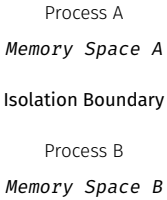
Critical System

핵심

한 영역이 공격당하더라도 다른 중요한 영역으로 피해가 쉽게 확산되지 않도록 분리한다.

16. Example: Isolation

Operating System의 Process Isolation을 생각해보자.



정상적인 상황에서는 Process A가 Process B의 Memory를 마음대로 읽거나 수정할 수 없다.

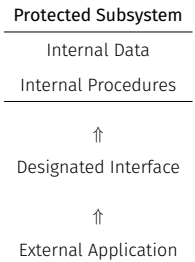
같은 원리는 Network Segmentation이나 Virtual Machine Isolation에도 적용될 수 있다.

핵심

Isolation은 Security Boundary를 만들어 Attack의 확산 범위를 제한한다.

17. Encapsulation

Encapsulation은 Data와 Procedure를 하나의 보호된 영역 안에 두는 방식이다.



External Application은 내부 Data에 직접 접근하지 않고 정해진 Interface를 통해서만 기능을 사용한다.

핵심

내부 구조를 직접 노출하지 않고 제한된 Entry Point를 통해서만 접근하도록 한다.

18. Modularity

Security Function은 가능한 한 독립적인 Module로 설계하는 것이 좋다.

예를 들어 여러 Application이 각각 Cryptographic Function을 구현한다고 생각해보자.

Application A $\rightarrow$ Crypto A

Application B $\rightarrow$ Crypto B

Application C $\rightarrow$ Crypto C

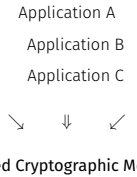
이 경우 각각의 Crypto Implementation을 모두 검증하고 유지보수해야 한다.

문제

같은 Security Function이 여러 곳에 중복 구현되면 Bug와 Configuration Error가 증가할 수 있다.

19. Example: Modular Security Design

Security Function을 하나의 검증된 Module로 만들 수 있다.



이러한 구조의 장점은 다음과 같다.

- 하나의 Security Module에 검증을 집중할 수 있다.
- 중복된 Security Code를 줄일 수 있다.
- Security Update를 쉽게 적용할 수 있다.
- 새로운 Algorithm으로 교체하기 쉽다.

핵심

Modularity는 Security Function을 독립적으로 설계하여 검증, 유지보수, 업그레이드를 쉽게 만든다.

20. Layering

하나의 Security Mechanism만으로 전체 시스템을 보호하려고 해서는 안 된다.

여러 Security Mechanism을 겹쳐서 사용하는 것이 바람직하다.

Internet

⇓

Firewall

⇓

Intrusion Detection

⇓

Authentication

⇓

Access Control

⇓

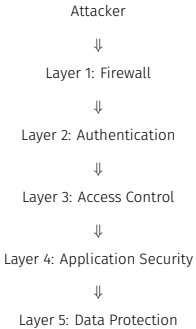
Protected Resource

핵심

하나의 Protection Layer가 실패해도 다른 Layer가 추가적인 방어를 제공하도록 설계한다.

21. Defense in Depth

Layering을 Security에서는 **Defense in Depth**라고도 부른다.



공격자가 하나의 Layer를 우회하더라도 다른 Security Control을 다시 통과해야 한다.

핵심

Defense in Depth는 Single Point of Security Failure에 의존하지 않는다.

22. Example: Layering in a Real System

예를 들어 중요한 Database를 보호한다고 생각해보자.

Internet

⇓

Firewall

⇓

VPN

⇓

User Authentication

⇓

Role-Based Access Control

⇓

Database Authentication

⇓

Encrypted Database

공격자가 Firewall을 우회했다고 해서 바로 Database에 접근할 수 있는 것은 아니다.

중요

Layering은 각각의 Mechanism이 완벽하다고 가정하지 않는다. 각 Mechanism의 실패 가능성을 전제로 여러 방어선을 둔다.

23. Least Astonishment

System과 User Interface는 사용자가 예상하는 방식으로 동작해야 한다.

특히 Security Setting은 사용자가 직관적으로 이해할 수 있어야 한다.

예를 들어 사용자가 File을

Private

으로 설정했다면,
사용자는 일반적으로

"다른 사람은 이 File을 볼 수 없다"

고 기대한다.

실제 동작도 이러한 기대와 일치해야 한다.

핵심

Security Mechanism의 동작이 사용자의 기대와 다르면 잘못된 Security Decision과 Configuration Error를 유발할 수 있다.

24. Psychological Acceptability vs. Least Astonishment

두 원칙은 비슷하지만 초점이 조금 다르다.

Usability-related Principles

Principle	Main Idea
Psychological Acceptability	Security Mechanism이 지나치게 불편하여 사용자가 우회하거나 비활성화하지 않도록 한다.
Least Astonishment	Security Mechanism이 사용자가 예상하는 방식으로 직관적으로 동작하도록 한다.

Easy to Use

+

Easy to Understand

핵심

Secure System은 기술적으로 안전할 뿐 아니라 사용자가 올바르게 사용할 수 있도록 설계되어야 한다.

25. Comparing Similar Principles

몇몇 Principle은 서로 비슷해 보이지만 초점이 다르다.

Similar but Different

Principle	Main Focus
Least Privilege	각 User / Process가 가지는 권한을 최소화
Separation of Privilege	중요한 Access에 여러 조건이나 Privilege를 요구
Least Common Mechanism	여러 User가 공유하는 Mechanism을 최소화
Isolation	System, Process, Resource 사이에 Security Boundary를 형성
Encapsulation	Protected Resource에 지정된 Interface를 통해서만 접근
Modularity	Security Function을 독립적인 Module로 구성
Layering	여러 Security Mechanism을 겹쳐 사용

26. Applying the Principles to One System

Student Information System을 예로 생각해보자.

Example

Principle	Example
Fail-Safe Defaults	기본적으로 Student Record 접근을 Deny
Least Privilege	학생은 자신의 정보만 열람 가능
Complete Mediation	Record 접근 시 Permission을 확인
Separation of Privilege	관리자 Login에 MFA 적용
Isolation	Public Web Server와 Student Database를 분리
Layering	Firewall + Authentication + Access Control 사용
Psychological Acceptability	사용자가 이해하기 쉬운 Security Procedure 제공

핵심

하나의 Secure System에도 여러 Security Design Principle이 동시에 적용된다.

27. Applying the Principles to a Drone System

같은 Principle은 Cyber-Physical System에도 적용된다.

Drone Security Example

Principle	Example
Least Privilege	ROS2 Node가 필요한 Topic에만 접근
Fail-Safe Defaults	인증되지 않은 Command는 기본적으로 거부
Isolation	Flight Controller와 External Network를 분리
Modularity	Security Monitoring을 별도 Module로 구성
Layering	Communication Authentication + Access Control + Anomaly Detection 적용
Complete Mediation	중요한 Command 입력마다 Authorization 확인

핵심

Security Design Principle은 Web이나 Server뿐 아니라 Robot, Drone, Autonomous System에도 동일하게 적용할 수 있다.

28. The Most Important Principles

모든 Principle을 암기하기보다 다음 핵심적인 사고를 기억하는 것이 중요하다.

▶ Keep it simple	Economy of Mechanism
▶ Deny by default	Fail-Safe Defaults
▶ Check every access	Complete Mediation
▶ Do not rely on secret design	Open Design
▶ Give only necessary privilege	Least Privilege
▶ Separate critical resources	Isolation
▶ Use multiple protection layers	Layering / Defense in Depth

핵심

Security Design Principle은 Attack을 하나씩 막는 기술이 아니라, 취약점이 생길 가능성과 공격 성공의 영향을 구조적으로 줄이는 방법이다.

29. Security Principles and Risk

Security Design Principle의 궁극적인 목적은 Risk를 줄이는 것이다.

Good Security Design

⇓

Fewer Vulnerabilities

⇓

Smaller Attack Surface

⇓

Limited Attack Impact

⇓

Reduced Risk

예를 들어 Least Privilege는 공격 자체를 항상 막지는 못한다.

하지만 Account가 Compromise되었을 때 공격자가 사용할 수 있는 권한과 피해 범위를 제한한다.

핵심

좋은 Security Design은 공격 가능성을 줄이는 것뿐 아니라, 공격이 성공했을 때의 피해도 제한한다.

30. Summary (1)

Fundamental Security Design Principle은 Secure System을 설계하기 위한 기본 원칙이다.

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

핵심

초기의 Security Principle들은 단순성, 안전한 기본값, Access Control, Privilege Management를 강조한다.

31. Summary (2)

추가적으로 다음과 같은 System Architecture 원칙이 중요하다.

- Isolation
- Encapsulation
- Modularity
- Layering
- Least Astonishment

Simple

Minimal Privilege

Separated

Modular

Layered

Understandable

핵심 문장

Secure System은 하나의 강력한 Security Mechanism으로 만들어지는 것이 아니라, 좋은 Design Principle을 시스템 전체에 적용함으로써 만들어진다.

32. A Question to Ask When Designing Security

새로운 Security System을 볼 때 다음 질문을 해볼 수 있다.

- Is the design unnecessarily complex?
- What happens by default?
- Is every important access checked?
- Does the system rely on hidden design?
- Does each user have only necessary privileges?
- Are critical components properly isolated?
- What happens if one Security Mechanism fails?
- Can users understand and correctly use the mechanism?

Security Insight

Security Design Principle은 “이 시스템은 안전한가?”라는 질문을 구체적인 설계 질문으로 바꾸어 준다.

33. Bridge to the Next Section

지금까지는 Secure System을 어떤 원칙으로 설계해야 하는지 살펴보았다.

그러나 공격자 입장에서는 또 다른 질문을 할 수 있다.

Where can I attack?

즉 시스템에는 공격자가 접근할 수 있는 여러 Attack Surface가 존재한다.

Next

다음 Section에서는 Attack Surface와 Attack Tree를 이용하여 공격 가능한 경로를 구조적으로 분석한다.

Thank You

Questions?