차례

7.1 개요

- 인가(authorization)
  - 인증된 사용자에게 허용된 행동을 한정하는 과정
  - 어떤 사용자가 로그인을 통해 ' 누구인지 ' 가 확인되었다면, 이제 '무엇을 할 수 있는가'를 결정해야함
  - 보안 정책에 따라 접근 가능 자원을 판단하고, 허용 또는 거부 결정을 내림

- 접근제어의 핵심역할
  - 인가는 접근제어의 중심단계로, 실제 보안정책이 시스템에 행동제약 형태로 반영 되는 부분

- 보안 정책 실행의 실제 구현
  - 인가는 추상적인 보안 목표(기밀성, 무결성)을 운영 수준에서 구체화하는 절차

7.1 인증 vs 인가

- 인증(Authentication)
  - 사용자의 정체성을 확인하는 절차
  - 로그인 ID, 비밀번호, 생체인식 등으로 '이 사용자가 누구인가?'를 검증
  - 목적: 신원을 보장하는 것

- 인가(authorization)
  - 인증된 사용자에게 허용된 행동을 한정하는 과정
  - 접근 권한 부여(Grant)와 제한(Deny)를 모두 포함
  - 목적: 행동의 범위를 통제하는 것

7.1 인가의 필요성

■ 동일 권한의 위험성

- 모든 사용자가 같은 권한을 가진다면, 데이터 유출 삭제 조작의 위험이 커짐

■ 데이터 중요도 기반 권한 분리

- 데이터는 중요도에 따라 접근 등급을 달리해야 함
  - 일반 데이터 (일반 직원 접근 가능)
  - 기밀 데이터(관리자 전용)
  - 최고기밀(보안 관리자만 접근 가능)

■ 내부자 위협의 현실성

- 보안 사고의 상당수는 내부 권한 남용에서 발생
- 외부 공격보다 내부의 실수나 악의적 행동이 더 큰 피해를 유발할 수 있음

■ 인가의 본질

- 인가는 단순한 통제가 아니라, 신뢰를 구조화 하는 절차

7.1 접근제어의 3요소

- 주체
  - 접근을 시도하는 존재
  - 사용자, 프로세스, 애플리케이션 등
- 객체
  - 보호되어야 하는 대상
  - 파일, 데이터베이스, 시스템 자원 등
- 접근권한
  - 주체가 객체에 수행할 수 있는 행동의 종류
- 인가 결정의 핵심 로직
  - 누가(Subject), 무엇(Object), 어떻게(Operation) => 세가지를 매칭해 접근 허용 여부를 판단

7.1 보안의 세 가지 목표

■ 기밀성

- 허가되지 않은 접근을 차단하는 것
- 데이터는 오직 인가된 사용자만 접근 가능해야 함
- 인가 시스템은 기밀성을 직접 보장하는 핵심 구성요소

■ 무결성

- 데이터가 변조되지 않도록 보호
- 시스템은 비인가 사용자의 수정 요청을 차단

■ 가용성

- 인가된 사용자가 필요할 때 접근할 수 있어야 함
- 지나친 통제는 오히려 서비스의 가용성을 해칠 수 있음

■ 인가와 보안의 관계

- 인증이 신원을 보장한다면, 인가는 보안 3요소를 유지하는 것

7.1 인가의 작동 단계

■ 인가의 전체 흐름

- 인가는 인증이 끝난 다음 단계에서 작동
- 사용자가 로그인에 성공하면, 그 사용자의 권한이 시스템 내부에서 평가

■ 인가의 세가지 절차

- 사용자 인증
  - 사용자 ID, 비밀 번호 등을 통해 '누구인가?'를 확인
  - 이 단계에서 사용자는 단순히 인정된 존재
- 정책 확인
  - 시스템은 Access Policy를 조회
  - 정책에는 사용자 역할, 데이터 등급, 시간대, 위치 등 포함
- 권한 부여 또는 차단
  - 정책 조건을 만족하면 접근 허용(Grant), 그렇지 않으면 차단(Deny)
  - 이 결정은 OS, DBMS 등에서 자동 수행

7.1 인가 정책의 예시

- Role-Based Access Control(RBAC)
  - 사용자의 역할(Role)에 따라 권한을 부여하는 방식
  - 관리가 간단하고 대규모 조직에 적합
  - Ex)
    - 의사: 진료기록 읽기 수정 가능 / 간호사: 읽기만 가능 / 행정직원: 환자정보 접근 불가

- Discretionary Access Control(DAC)
  - 자원 소유자가 접근 권한을 임의로 지정
  - 장점: 유연함/ 단점: 잘못 설정 시 보안 취약

- Mandatory Access Control(MAC)
  - 시스템이 정책을 강제 적용
  - 사용자는 권한을 변경할 수 없음
  - 가정 엄격하지만, 운영 유연성은 낮음

7.1 인가의 주요 기술요소

■ 인증정보 전달 구조 (Token, Session 등)

- 인가는 인증 결과를 기반으로 동작하므로, 인증 정보가 안전하게 전달되어야 함
- 세션(Session) 기반:
  - 서버가 로그인한 사용자의 상태를 서버 메모리에 저장
  - 장점: 구현이 단순 / 단점: 서버 부하 및 확장성 제한
- 토큰(Token) 기반:
  - 인증 정보를 암호화하여 토큰(JWT 등) 형태로 클라이언트가 보유서버는 요청 시 토큰을 검증
  - 장점: 확장성 뛰어남, 분산환경(클라우드, API) 적합

■ 정책 결정 포인트 (PDP: Policy Decision Point)

- 인가 정책을 해석하고 결정하는 지점
- PDP는 보안정책 데이터베이스(Policy Store)와 연동되어 동적 판단 수행

■ 정책 시행 포인트 (PEP: Policy Enforcement Point)

- PDP의 결정을 실제로 강제(Enforce)하는 지점
- 방화벽, API Gateway, Proxy 등에서 요청을 차단하거나 허용하는 역할

7.1 인가의 평가기준

■ 보안성(Security)

- 인가 시스템의 가장 기본적 기준
- 불법 접근, 권한 상승, 내부 남용을 방지해야 함

■ 효율성(Performance)

- 보안이 아무리 뛰어나도, 응답 속도가 느리면 실용성이 떨어짐
- PDP의 의사결정 과정과 정책 조회 속도는 시스템 부하에 직접적 영향을 줌
- 대규모 서비스에서는 캐싱 및 분산처리로 성능 확보 필요

■ 유연성(Flexibility)

- 사용자, 조직, 환경이 끊임없이 변하기 때문에 정책 수정이 용이해야 함
- 새로운 서비스가 추가되어도 인가 시스템 전체를 수정하지 않도록 설계해야 함

■ 관리편의성(Manageability)

- 관리자 입장에서 정책 등록·변경·삭제가 직관적이어야 함
- 복잡한 권한 구조는 오히려 보안 사고를 유발
- GUI 기반 정책관리 도구 또는 중앙집중형 IAM이 효과적

7.1 인가와 인증의 상호작용

■ 보안 시스템의 통합적 동작

- 대부분의 IAM(Identity & Access Management) 시스템은Authentication Layer와 Authorization Layer를 통합 구조로 운영
- 이로써 사용자 경험(UX)과 보안 정책 간의 일관성을 유지

7.1 현대 인가 시스템의 트렌드

■ Zero Trust 접근제어

- 기존에는 내부망을 '신뢰할 수 있는 공간'으로 간주했지만,
  Zero Trust는 "아무도 기본적으로 신뢰하지 않는다"는 철학
- 모든 요청은 매번 인증과 인가를 다시 수행한다.
- 구현 예: Google BeyondCorp, Microsoft Zero Trust Architecture

■ Attribute-Based Access Control (ABAC)

- 사용자 속성(Attribute), 환경(Context), 자원 특성을 기반으로 인가 결정
- 예:
  - "부서=재무부" & "시간=근무시간" & "위치=사무실"일 때만 접근 허용
- RBAC보다 세밀하고, 상황인지(Context-Aware) 접근제어 가능

■ 클라우드 환경의 IAM (Identity & Access Management)

- 클라우드 서비스(AWS, Azure, GCP 등)는 IAM을 통해 사용자 계정, 역할(Role), 정책(Policy)을 중앙에서 관리
- 토큰 기반 인증과 PEP/PDP 구조를 통합하여 대규모 멀티테넌트 환경에서도 일관된 인가 정책 유지

7.2 인가의 역사 개요

■ 인가 발전의 배경

- 인가(Authorization)는 단순한 기술이 아니라, "신뢰 수준을 관리하는 국가적 체계" 에서 발전
- 1970~80년대, 정보 시스템이 군사·정부기관 중심으로 확산되면서 "컴퓨터를 얼마나 믿을 수 있는가?"라는 질문이 등장

■ 보안 수준 평가의 필요성

- 각 기관이 제각각 기준으로 시스템을 평가하던 시절, 신뢰할 수 있는 통합된 보안 등급 체계가 요구

■ 국가 주도의 표준화 시도

- 미국 정부는 국가 차원에서 보안정책을 표준화하기 위해 'TCSEC(오렌지북)'을 제정
- 이후 유럽과 일본이 자국 기준을 도입하면서, 국제적으로 공통 평가 기준(Common Criteria)으로 통합

7.2 오렌지북의 탄생

■ 오렌지북의 정식 명칭

- TCSEC (Trusted Computer System Evaluation Criteria)
- 1983년, 미국 국방성(DOD) 주도로 발표

■ 개발 목적

- 군사·정보기관이 사용하는 컴퓨터 시스템의 보안 신뢰도를 정량 평가
- 제품 제조사에게 보안 설계 가이드라인 제공
- 정부 조달용 시스템의 표준 인증 프레임워크 제공

■ 의미

- 정보보안이 '기술자의 감'에서 '공식 기준'으로 바뀌는 전환점
- "보안을 측정 가능하게 만든 최초의 제도"

7.2 오렌지북의 목적

■ 자동 데이터 처리 시스템의 보안 수준 평가

- 당시 대형 메인프레임과 네트워크가 보급되면서 시스템 간 정보 교환이 폭발적으로 증가
- 이에 따라 데이터 접근, 저장, 송신 과정의 보안 등급화 필요성이 대두

■ 제조사에 대한 보안 설계 기준 제공

- 하드웨어·소프트웨어 업체는 TCSEC 기준에 맞춰 설계해야 '신뢰 가능한 제품(Trusted Product)'으로 인정
- 사용자(정부·기관)에게 선택 기준 제공
- 제품 구매 전 "이 시스템은 B1 등급, 저건 C2 등급"과 같은 보안 레벨별 비교 기준을 제공

■ 직관적 예시

- '보안의 에너지효율 등급제'와 유사: A1 = 최고 신뢰, D = 무보호

7.2 오렌지북의 보안 등급 체계

등급	명칭	유형	설명
D	Minimal Protection	최소 보호	아무런 보안 요구 사항 없음
C1, C2	Discretionary Protection	재량적 보호	사용자가 직접 권한을 부여
B1, B2, B3	Mandatory Protection	필수적 보호	시스템이 강제적으로 보호 수행
A1	Verified Design	검증된 보호	보안기능이 수학적으로 검증됨

7.2 D 등급 – 최소 보호

■ 정의

- 가장 낮은 수준의 보안 등급
- 인증, 접근제어, 감사기록 등의 요구사항이 전혀 없음

■ 적용 예

- 테스트용 서버, 개인 실험 환경, 단기 프로젝트 시스템
- 단순히 '작동만 되는' 수준의 임시 체계

■ 의의

- 등급 분류의 기준점(Reference Baseline)으로 존재
- 오렌지북의 시작점이며, '보안 없음(No Protection)'의 명시적 정의

7.2 C 등급 – 재량적 보호

C1: 기본적 사용자 식별 및 접근제어

- 사용자 ID 기반 접근제어
- 각 사용자별로 '누구의 데이터인가'를 구분 가능

C2: 강화된 접근통제

- 세분화된 접근제어 + 감사 로그(Audit Trail) 추가
- 사용자는 자신의 파일에 대해 읽기/쓰기 권한을 개별 설정 가능

특징

- 보호 주체: 사용자(Discretionary)
- 윈도우, 유닉스의 파일 권한 설정 구조가 대표적 예

비유

- "열쇠를 가진 사람이 직접 문을 잠그거나 여는 방식"

7.2 B 등급 – 필수적 보호

■ 핵심 개념

- 사용자의 재량을 넘어서, 시스템이 보안을 강제(Mandatory) 한다.

■ 세부 단계

- B1: 라벨(Label) 기반 보호
  - 각 데이터 객체에 보안등급(Label) 부여
  - 예: “Secret”, “Top Secret”, “Confidential”
  - 사용자의 보안 등급이 객체보다 낮으면 접근 불가
- B2: 구조화된 보호(Structured Protection)
  - 보안 커널(Security Kernel)이 명확히 분리
  - 시스템 내부 접근경로까지 보호
- B3: 보안 영역(Security Domain)
  - 독립된 보안 영역으로 프로세스가 격리
  - 침입이나 오류가 다른 영역으로 전이되지 않음

■ 의의

- 군사·정부 시스템에 실질적으로 적용된 최초의 '시스템 수준 보안'

7.2 A 등급 – 검증된 보호

■ A1: Verified Design

- 보안정책이 수학적 방법(Formal Verification)을 통해 증명된 단계
- 보안 아키텍처, 소스코드, 테스트 절차까지 전 과정 검증

■ 적용 대상

- 군사, 핵발전소, 정보기관 등 극도로 민감한 환경
- 모든 구성요소(하드웨어·OS·애플리케이션)가 신뢰 경로에 포함되어야 함

■ 의의

- 보안의 형식적 증명(Formal Proof)을 도입한 최초 사례
- 이후 Common Criteria의 'EAL7(최고 등급)'로 계승됨

7.2 오렌지북의 의의

■ 최초의 공식 보안등급 시스템

- 이전까지 주관적이던 보안 평가를 정량적 기준으로 제시
- 이후 전 세계 정부·방위산업에서 채택

■ 보안제품 개발의 표준화 촉진

- 보안이 "기능의 옵션"이 아니라 "설계의 필수요소"로 전환
- 제조사들이 TCSEC을 준수해야 정부 납품 가능

■ 국제표준 발전의 기반

- 오렌지북은 후에 ITSEC(유럽), CTCPEC(캐나다)로 이어지고 최종적으로 공통평가기준(CC: Common Criteria)으로 통합됨

7.2 오렌지북의 한계

■ 미국 중심의 군사기준

- 민간용, 상용 시스템에는 적용이 어렵고 비용이 과도함
- "보안은 완벽하지만 현실성이 떨어진다"는 비판

■ 상용 및 인터넷 환경 부적합

- PC·인터넷이 확산되며, 네트워크 기반 위협에 대응 불가
- 동적 권한 변경, 클라우드 환경에 맞지 않음

■ 기술 발전 속도 미흡

- 급속히 진화하는 IT 생태계에 비해 평가절차가 너무 정적
- 보안보다 '평가 인증 자체'가 목적화되는 문제 발생

■ 새로운 표준의 등장 배경

- 이러한 한계로 인해 국제사회는 공통평가기준(CC)을 새로 수립
- 오렌지북은 "보안 평가의 출발점이자 한계점"으로 평가된다.

7.2 공통 평가 기준(Common Criteria, CC) 등장

■ 오렌지북의 한계를 극복하기 위한 새로운 시도

- 1990년대, 각국이 독자적인 보안 평가 기준(TCSEC, ITSEC, CTCPEC)을 운영하며 상호 호환성 문제 발생
- 국가 간 보안제품 교류 및 조달 과정에서 인증 중복이 심각한 문제로 부상

■ 국제 협력 프로젝트의 결과

- 미국, 캐나다, 유럽, 일본 등이 협력하여 "공통 평가 기준(Common Criteria, CC)" 제정
- 1999년 ISO/IEC 15408로 국제표준화
- 목적: "국가 간 상호 신뢰 가능한 보안 인증 체계" 수립

■ 철학적 변화

- TCSEC이 "보안 등급" 중심이었다면, CC는 "보증(Assurance)" 중심으로 전환
- "이 시스템이 얼마나 검증 가능한가?"를 평가

7.2 CC의 핵심 개념

■ CC의 기본 구조

- 보안 제품의 기능 요구사항(Functionality) 과 보증 요구사항(Assurance) 을 명확히 구분
- 즉, "무엇을 할 수 있는가"와 "그 기능이 얼마나 신뢰할 수 있는가"를 각각 평가

■ 평가보증등급 (EAL: Evaluation Assurance Level)

- EAL은 보안제품이 얼마나 철저히 검증되었는가를 나타내는 등급
- EAL1 ~ EAL7까지 존재하며, 숫자가 높을수록 높은 신뢰성

■ TCSEC과의 차별점

- TCSEC: '보호수준' 자체 평가
- CC: '검증과정의 깊이' 평가
- 즉, CC는 "검증 절차의 투명성"을 중시

7.2 EAL 등급 체계

등급	명칭	설명 요약
EAL1	기능 시험 수준	최소한의 기능 시험만 수행
EAL2	구조적 시험	기본적인 설계 검증 수행
EAL3	방법적 시험	체계적 테스트 및 설계 분석
EAL4	설계 검증 및 보안 강화	일반 상용 수준의 높은 신뢰성
EAL5	반정형 설계 검증	고급 보안제품 수준, 반자동 검증 포함
EAL6	반형식적 설계 검증	군사용·핵시설용 고보증 등급
EAL7	완전한 형식 검증	수학적 모델 기반 완전 검증 수준

7.2 CC 인증 과정

- **TOE (Target of Evaluation) 정의**
  - 평가 대상 시스템 또는 제품을 명확히 규정
  - 예: 방화벽, 운영체제, 모바일 보안모듈 등
- **보안기능명세(SF: Security Functions) 작성**
  - 제품이 제공하는 보안 기능(암호화, 접근제어, 로깅 등)을 문서화
  - 이를 기반으로 평가기관이 테스트 항목을 정의
- **시험기관의 평가**
  - 공인된 제3자 평가기관이 설계문서, 코드, 테스트 결과를 검증
  - CC 표준에 따라 일관된 절차로 수행
- **인증서 발급**
  - 국가별 인증기관이 EAL 등급을 부여하고 인증서 발급
  - 국제 상호인정협정(CCRA)을 통해 다른 국가에서도 인증 효력 유지

7.2 CC 인증기관

국가	주요 기관	역할
미국	NIST(미국표준기술연구소), NIAP	시험소 인증, CCRA 운영 주체
한국	KISA(한국인터넷진흥원)	공공·국방 보안제품 인증
독일	BSI	유럽 내 대표적 평가기관
일본	IPA	산업용 소프트웨어 인증 관리
프랑스	ANSSI	국가 보안평가 및 인증

7.2 오렌지북 vs 공통평가기준 비교

항목	오렌지북 (TCSEC)	공통평가기준 (CC)
적용범위	군사 중심	국제·상용 제품
등급체계	A~D 단계	EAL1~EAL7 단계
개발시기	1983 (미국 DOD)	1999 (ISO/IEC 15408)
평가기준	보호 수준	보증 수준
적용성	미국 내부	국제 표준

7.2 실제 CC 인증 제품 사례

제품명	제조사	EAL 등급	적용 분야
Cisco ASA 방화벽	Cisco	EAL4	네트워크 보안
HP Smart Card	Hewlett-Packard	EAL5+	인증·결제 시스템
Samsung Knox	Samsung Electronics	EAL2~EAL4	모바일 보안
국내 공공 보안 솔루션	여러 기업	EAL4	정부·공공망

7.2 인가 역사 요약

- **오렌지북 (TCSEC): 출발점**
  - 보안 수준을 계량화한 최초의 기준
  - 군사적 신뢰성 확보를 목표로 함
- **공통평가기준 (CC): 진화**
  - 오렌지북의 철학을 계승하되, 국가 간 인증 통합과 상용화 확장을 달성
- **현대 인가 체계로의 영향**
  - CC 기반의 EAL 개념은 오늘날 IAM, Zero Trust 평가에도 적용
  - 인가 시스템의 설계는 '검증 가능한 신뢰'라는 CC의 철학 위에 서 있다.

7.3 접근제어 행렬 (Access Control Matrix)

■ 인가를 체계화한 최초의 모델

- 1971년 버틀러 램슨(Butler Lampson)이 제안
- 인가(Authorization)를 수학적 구조로 표현
- 접근제어 문제를 표(Table) 형태로 정리한 "접근제어 행렬 모델"로 발전

■ 핵심 아이디어

- "모든 접근 행위는 주체(Subject), 객체(Object), 행위(Operation)의 3요소로 정의 가능하다."
- 이 세 가지를 하나의 행렬(Matrix)로 표현하면, 인가 정책을 명확히 시각화할 수 있음

■ 의의

- 컴퓨터 보안의 이론적 토대를 마련
- 오늘날 모든 접근제어 시스템(ACL, RBAC, IAM 등)의 출발점

7.3 접근제어 행렬의 기본 개념

■ 구조 구성

- 행(Row) → 주체(Subject): 사용자, 프로세스, 프로그램
- 열(Column) → 객체(Object): 파일, 디렉터리, 데이터베이스, 네트워크 자원
- 셀(Cell) → 권한(Operation): 읽기(Read), 쓰기(Write), 실행(Execute), 삭제(Delete) 등

■ 의미

- 각 셀은 "특정 주체가 특정 객체에 대해 수행 가능한 행위"를 명시
- 인가 시스템은 사용자의 요청이 발생할 때, 이 행렬의 해당 셀을 참조하여 허용 여부를 판단

7.3 접근제어 행렬 예시

주체/객체	보험데이터	회계데이터	OS	회계프로그램
앨리스	rw	r	rx	rx
밥	-	r	-	-
샘	rw	rw	-	rw

7.3 접근제어 행렬의 의미

■ 인가 결정의 실질적 기준

- 행렬의 각 셀은 인가 정책의 "원자 단위(Atomic Rule)"
- 시스템은 접근 요청이 들어올 때,
  (주체, 객체, 연산) 조합을 확인하여 허용 또는 차단을 결정

■ 작동 과정

1. 사용자 요청 발생

2. 시스템이 해당 (주체, 객체) 위치의 셀을 조회

3. 셀에 해당 권한이 존재하면 허용, 없으면 거부

■ 보안 시스템에서의 역할

- OS, DBMS, 네트워크 방화벽 등에서 모두 사용되는 기본 원리
- 오늘날 IAM과 클라우드 권한정책(JSON 기반)도 사실상 행렬의 변형 표현

7.3 접근제어 행렬의 장점

- **단순하고 직관적인 구조**
  - 인가 정책을 한눈에 시각화 가능
  - "누가 무엇을 할 수 있는가"가 명확하게 정리됨
- **정책 명세의 명료성**
  - 각 셀 단위로 정책을 기술하므로, 누락이나 중복을 쉽게 발견
- **중앙집중식 관리**
  - 하나의 테이블에서 전체 시스템 권한을 통제 가능
  - 관리자가 단일 위치에서 정책을 수정할 수 있음
- **교육적 가치**
  - 보안 개념을 설명할 때 가장 기본적인 모델로 활용

7.3 접근제어 행렬의 한계

■ 확장성 문제

- 주체와 객체의 수가 많아질수록 행렬의 크기가 폭증
  예: 1,000명 × 10,000개 파일 = 1,000만 셀
- 메모리 낭비 및 관리 불가능

■ 실시간 접근 처리 비효율

- 요청마다 행렬 전체 탐색 시 성능 저하

■ 해결책

- 현실 시스템에서는 행렬 전체를 저장하지 않고, "분해(Decomposition)"하여 관리
  → ① 객체 중심(ACL)
  → ② 주체 중심(C-list)

7.3 접근제어 목록 (ACL: Access Control List)

■ 개념

- 객체 중심(Object-centric) 접근제어 방식
- 각 객체마다 "누가 접근 가능한가"를 개별적으로 기록

■ 구조

- 파일이나 자원마다 접근 가능한 사용자 리스트를 보유
- 각 사용자에 대해 허용된 연산(read/write/execute 등)을 명시

■ 실제 적용

- UNIX/Linux 파일 시스템의 rwx 권한
- Windows의 "파일 속성 → 보안 탭"

7.3 ACL의 장점

- **객체 중심 관리의 직관성**
  - 파일 단위로 접근권한을 지정하므로, "무엇을 보호할지" 명확
- **관리 용이성**
  - 각 자원마다 권한을 수정·확인하기 쉬움
- **보안적 응용성**
  - 데이터 중심 환경(문서관리, 파일서버, 클라우드 스토리지 등)에 적합
- **실제 예**
  - Windows, Linux, AWS S3 Bucket Policy 모두 ACL 기반

7.3 ACL의 단점

- **사용자 수 증가 시 관리 복잡도 급상승**
  - 수천 명의 사용자가 존재하면, 각 객체의 ACL을 일일이 관리 불가능
- **사용자 삭제 시 문제**
  - 사용자가 시스템을 떠나면, 해당 사용자의 권한을 모든 객체의 ACL에서 직접 제거해야 함
- **위임(Delegation)의 어려움**
  - 사용자 간 권한 전달이 비효율적
  - 중앙집중형 관리 불가
- **대규모 시스템에서는 한계**
  - 클라우드, 조직형 플랫폼에서는 비현실적

7.3 권한목록 (C-list, Capability List)

■ 개념

- 주체 중심(Subject-centric) 접근제어 방식
- 각 사용자(주체)가 자신이 접근 가능한 객체와 권한 목록을 보유
- 일종의 '디지털 열쇠 꾸러미(Digital Keyring)'

■ 핵심 특징

- 각 권한은 Capability(능력 토큰) 형태로 저장
- 사용자는 자신의 Capability를 제시하여 접근 권한을 입증

■ 응용

- 분산시스템, 클라우드 인증 토큰, API Key 구조 등에 활용
- 현대 OAuth, JWT 토큰 기반 인증도 C-list 개념을 계승

7.3 ACL vs C-list 비교

구분	ACL	C-list
중심	**객체 중심**	**주체 중심**
관리 단위	파일/데이터/버킷 등 **자원 단위**	사용자/서비스 계정 등 **주체 단위**
장점	데이터 보안 중심, 파일 단위 설정 직관적	권한 위임·회수 용이, 주체별 가시성 높음
단점	사용자 수 폭증 시 관리 난이도↑, '대리자' 구분 어려움	대규모 토큰 관리 오버헤드, 분실·유출 리스크
적합 환경	문서/폴더 중심, 온프레미스 파일 시스템	API/클라우드·마이크로서비스·OAuth 토큰 생태계

7.3 ACL과 C-list의 관계

- 접근제어 행렬(Subjects × Objects)의 열을 기준으로 분해 → 각 객체에 대한 접근 목록 = ACL
- 같은 행렬의 행을 기준으로 분해 → 각 주체의 능력 목록 = C-list
- 데이터는 동일, 저장·조회 관점만 다름.

7.3 대리 혼돈(Confused Deputy) 문제

- 정의: 권한을 가진 프로그램이 타 주체 요청을 대신 처리하다가, 누구의 권한으로 실행되는지 혼동하여 과도한 접근을 허용하는 취약점.
- 원인: 시스템이 “요청자(identity)”와 “실행 주체(executor)”의 권한 경계를 구분하지 못함.
- 결과: 프로그램이 자신의 높은 권한을 사용해 사용자 대신 의도치 않은 접근/수정을 수행.

7.3 대리 혼돈 시나리오

- 앨리스가 컴파일러를 실행(컴파일러는 시스템 파일에 쓸 수 있는 높은 권한 보유).앨리스가 디버그 출력 파일명을 **“billing(청구서)”**로 지정.컴파일러가 자신의 높은 권한으로 시스템 경로의 청구서 파일에 덮어쓰기.결과: 민감 데이터 손상/유출.

7.3 왜 대리 혼돈이 위험한가

- 권한 초과 행위: 프로그램이 사용자 요청을 빌미로 자신 권한으로 위험 작업 수행.내
- 부 프로세스 오용: 내부 마이크로서비스 간 신뢰 과잉으로 横전파 (lateral movement).
- 현대 사례: 클라우드 함수(AWS Lambda, GCP Cloud Functions 등)가 과도한 IAM Role을 가진 채 실행 → 타 계정 리소스 접근, 데이터 대량 유출.

7.3 ACL과 대리 혼돈

- 객체 중심(ACL)에선 요청 로그에 '어떤 프로세스/누구 권한으로 접근했는지'가 모호해지기 쉬움.
- 동일 사용자로 들어온 요청이라도 실제 실행 주체가 프로그램인지 사용자 본인인지 구분 어려움.
- 대리자(Deputy) 추적 난이도 ↑ → 권한 오남용 적발이 늦어짐.

7.3 C-list와 대리 혼돈 방지

- **주체 중심(C-list)은 능력 토큰(Capability)을 주체가 직접 제시 → *"누가, 어떤 권한으로, 어느 대상에, 어떤 범위로"*가 명시됨.**
- **권한 위임 흐름이 체인(Who → Whom → Scope)**으로 남아 추적 가능성↑.**
- **세분화된 스코프/만료/서명을 통해 오남용 리스크↓.**
- **적용 예**
  - OAuth 2.0 / OIDC: Access Token(스코프), Refresh Token(수명), Client Credential(대리자 식별).
  - JWT: 발급자/대상/만료/권한 클레임으로 권한 컨텍스트 고정.

7.3 권한 위임(Delegation)의 개념

- **의미: 원 주체가 자신의 권한 중 일부를 다른 주체에 부여하여, 대리자가 자신의 권한 범위 내에서 행동.**
- **원칙: *최소 권한(Least Privilege)*, *명시적 범위(Scope)*, *유효기간 (Expiry)*, *철회(Revocation)*.**
- **현대 구현: OAuth 토큰, 임시 자격증명(STS), 대리 실행 (impersonation), assume-role.**
- **권장 설계**
  - 역할 사슬(Role Chain) 길이 제한, 토큰 단명(short-lived), 재발급 보호, Audience 제한.

7.3 접근제어의 현대적 구현

■ ACL 기반

- Windows NTFS, Linux/Unix rwx/ACL, NAS/Samba, 일부 S3 ACL.
- 장점: 파일/문서 중심 통제가 명확.

■ C-list 기반/영향

- Capability OS(예: seL4 철학), 클라우드 IAM(AWS IAM Policy, GCP IAM, Azure RBAC), API 토큰.
- 장점: 서비스·마이크로서비스·서드파티 위임에 최적.

7.3 접근제어 행렬의 확장

- **RBAC: 행렬의 권한을 **역할(Role)**로 묶어 관리 복잡도 축소.**
- **ABAC: 속성(Attribute: 부서, 위치, 시간, 기기상태) 기반 동적 정책 (Context-Aware).**
- **MAC: 라벨/등급 기반 강제적 통제(군·정부).**
- **정리: 램슨 행렬은 정적 표상이고, RBAC/ABAC/MAC은 이를 조직·속성·라벨로 일반화/동적화한 모델.**
- **적용 기준**
  - 정적·대규모 역할: RBAC
  - 맥락·세밀 정책: ABAC
  - 최고 보안/격리: MAC

7.4 다단계 보안의 개념

- **정의:**

  다단계 보안(Multi-Level Security, MLS)은 서로 다른 보안 등급 (Security Level) 을 가진 주체(Subject)와 객체(Object)가 동일 시스템 내에 존재할 때,정보가 부적절하게 이동하거나 누출되지 않도록 통제하는 구조.

- **“누가 어떤 등급의 정보를 볼 수 있는가?”**
- **“정보가 어느 방향으로 흘러가도 되는가?”**
- **목적:**

  기밀성(Confidentiality) 확보와 내부 유출 차단.

7.4 다단계 보안이 필요한 이유

- **현대 시스템의 현실:**
  - 다양한 부서/프로세스가 하나의 서버에서 작동
  - 내부 DB에 서로 다른 기밀도가 공존 (예: 고객정보 vs. 공용데이터)
- **문제:**
  - 단순한 "읽기/쓰기 권한"만으로는정보 흐름(Information Flow) 통제가 불가능
- **해결:**
  - 보안 등급 간 "흐름 방향"을 제어하는 정책 도입

7.4 주체와 객체의 개념

구분	설명	예시
주체 (Subject)	정보를 접근/처리하는 존재 (사람, 프로세스)	사용자, OS 프로세스
객체 (Object)	보호되어야 하는 정보나 자원	파일, DB, 네트워크 리소스
취급등급 (Handling Level)	주체가 다룰 수 있는 최고 등급	Secret
기밀등급 (Security Level)	객체의 보안 민감도	Confidential

7.4 미국 국방성의 4단계 등급 체계

등급	설명
Top Secret	극비 – 국가안보에 중대한 영향
Secret	비밀 – 심각한 손실 초래 가능
Confidential	기밀 – 제한적 정보 보호
Unclassified	일반 평문 – 공개 가능 수준

7.4 다단계 보안의 기본 원리

- 원칙 1: 하위 등급 사용자는 상위 등급 정보에 접근 불가 (No Read Up)
- 원칙 2: 상위 등급 정보는 하위 등급으로 흘러가면 안 됨 (No Write Down)
- 목적: 기밀 정보의 하향 누출(Downward Leakage) 방지

7.4 Bell-LaPadula 모델 (BLP)

- **연도: 1973년**
- **제안자: David Bell, Leonard LaPadula**
- **목적: 군사 시스템에서의 기밀성(Confidentiality) 보호**
- **핵심: “정보가 하위 등급으로 흘러가는 것을 수학적으로 금지”**
- **BLP 모델의 핵심 문장:**
  - “A secure system must prevent unauthorized disclosure of information.”

7.4 BLP 모델의 두 가지 규칙

규칙	명칭	의미	설명
Simple Security Property	상향 읽기 금지 (No Read Up)	주체 S는 자신의 등급보다 높은 객체 O를 **읽을 수 없음**	기밀 보호
***-Property (Star Property)**	하향 쓰기 금지 (No Write Down)	주체 S는 자신의 등급보다 낮은 객체 O에 **쓸 수 없음**	정보 유출 방지

7.4 BLP를 수식으로 표현

- **읽기 조건 (Read):**

  $L(O) \leq L(S)$ → 객체의 등급이 주체 등급보다 낮거나 같아야 함

- **쓰기 조건 (Write):**

  $L(S) \leq L(O)$ → 주체의 등급이 객체보다 낮거나 같아야 함

7.4 BLP의 동작 예시

주체	등급	객체	객체 등급	허용 여부
앨리스	Secret	보고서	Top Secret	읽기 불가 (No Read Up)
밥	Secret	회계자료	Confidential	읽기 가능
찰리	Top Secret	발표자료	Secret	쓰기 불가 (No Write Down)

7.4 BLP 모델의 한계

■ 1. 무결성(Integrity) 미고려

- BLP는 "읽기 금지, 쓰기 금지"로 기밀성만 유지 → 데이터 신뢰성 검증 불가능

■ 2. 현실적 제약

- 실제 시스템은 '읽기'와 '쓰기'를 함께 수행해야 함

■ 3. 유연성 부족

- 등급(Level) 변경이 어려워, 동적 환경(예: 클라우드, 협업)에 비적합

7.4 Biba 모델의 등장

- 연도: 1977년
- 제안자: Kenneth J. Biba
- 목적: 정보의 무결성(Integrity) 보호
- 핵심 철학:

  Bell–LaPadula(BLP)가 “정보 유출 방지(Confidentiality)”에 집중했다면,

  Biba는 “정보 오염 방지(Integrity)”에 집중함.
- 즉, BLP와 방향이 정반대!

7.4 Biba 모델의 규칙

■ 쓰기 접근 규칙 (No Write Up)

- $I(O) \leq I(S)$ 일 때만, 주체 S가 객체 O에 쓸 수 있음.
- → 낮은 무결성 수준의 주체가 높은 무결성 객체에 데이터를 쓸 수 없음.

■ 읽기 접근 규칙 (No Read Down)

- $I(S) \leq I(O)$ 일 때만, 주체 S가 객체 O를 읽을 수 있음.
- → 높은 무결성 주체가 낮은 무결성 데이터를 읽을 수 없음.

7.4 Biba 모델의 규칙

구분	**BLP 모델**	**Biba 모델**
초점	기밀성(Confidentiality)	무결성(Integrity)
읽기 제한	상향 읽기 금지 (No Read Up)	하향 읽기 금지 (No Read Down)
쓰기 제한	하향 쓰기 금지 (No Write Down)	상향 쓰기 금지 (No Write Up)
방향성	정보 유출 방지	데이터 오염 방지
적용 분야	군사·정보기관	금융·산업제어·의료시스템

7.4 저수위 원칙 (Low-Water-Mark Principle)

■ **정의:**

- 주체 S가 객체 O를 읽으면→ $I(S)=\min(I(S),I(O))$

■ **의미:**

- 낮은 무결성의 데이터를 읽는 순간,주체의 무결성 수준도 함께 낮아짐.

■ **결과:**

- 시스템 내에서 "신뢰 수준 하락(Integrity Degradation)"이 전파됨.

7.4 BLP와 Biba의 병합적 접근

- **현대 시스템은 기밀성과 무결성을 동시에 요구.**
- **따라서 BLP + Biba의 복합적 적용 필요.**
- **예시:**
  - 군사 시스템: "극비 정보 보호 + 명령 신뢰성 유지"
  - 의료 데이터베이스: "환자 데이터 기밀 + 진료기록 무결성 보장"

7.4 구획화 (Compartment) 개념

- **등급 + 주제(Compartment) 로 세분화한 다차원 보안.**
- **예시:**
  - Top Secret {Nuclear}
  - Top Secret {Cyber}
  - 두 구획 간 접근은 서로 차단.
- **의의:**
  - 같은 등급이라도 업무영역이 다르면 접근 불가.

7.4 구획화의 원리

- **“Need to Know (알 필요가 있을 때만)”**
- **보안등급이 같더라도 업무 관련성(Compartment Membership) 이 없으면 접근 불가.**
- **기술적 구현:**
  - MAC(Mandatory Access Control) 기반 라벨링
  - 사용자 속성 기반 접근제어(ABAC)로 발전.

7.4 다단계 보안의 실제 적용

분야	적용 사례
네트워크	내부망/외부망 분리, 방화벽 정책 계층화
데이터베이스(DB)	관리자/사용자 권한 분리, 레코드 단위 접근제어
클라우드 IAM	리소스별 최소권한(Least Privilege) 정책
산업 시스템	제어망과 관리망 분리 (ICS/SCADA 보안)

7.4 BLP 모델의 안정성 논쟁

■ 비판 (1987, McLean):

BLP는 지나치게 정적이며, 현실 시스템의 동적 변화를 설명하지 못함.

→ “System Z”를 통해 한계 제시.

■ 개선 (Bell & LaPadula):

“Stability Property(안정 특성)” 개념 추가

→ 시스템 상태 변화 후에도 일관된 보안성 유지 보장.

7.4 현대적 관점의 다단계 보안

■ 전통적 MLS 한계:

고정된 등급체계는 현대의 동적 시스템(클라우드, AI, IoT 등)에 비효율적.

■ 현대적 접근:

등급 중심(Level-based)이 아닌 정책 기반 접근통제(PBAC: Policy-Based Access Control) 로 확장.

■ 핵심 기술:

ABAC(Attribute-Based Access Control) — 속성(Attribute)에 기반한 접근제어.

→ 등급 대신 “조건(Who, When, Where, Why)”으로 접근을 결정.

7.4 다단계 보안의 도전과제

■ 1. 데이터 복제 및 캐싱 문제

- 클라우드·AI 시스템에서 동일 데이터가 여러 노드에 복제.
- 등급 변경 시, 모든 사본의 보안 레벨 동기화 어려움.

■ 2. 동적 환경의 보안판단 복잡성

- AI 서비스, IoT, 엣지 컴퓨팅 등에서는
  "접근 시점마다 등급·속성·위험도가 변함."
- 정적 MLS로는 실시간 상황 반영 불가.

■ 3. 보안성과 효율성의 균형

- 과도한 통제 → 생산성 저하
- 느슨한 통제 → 정보 유출 위험
- → 결과적으로 적응형 보안(Adaptive Security) 개념으로 진화.

■ Adaptive Security 예시:

- 실시간 리스크 스코어 기반 동적 권한 조정
- AI가 상황(Context)을 판단해 "지금은 접근 허용/차단"

7.4 다단계 보안 요약

모델	초점	주요 원칙
BLP	기밀성 (Confidentiality)	상향 읽기 금지 (No Read Up), 하향 쓰기 금지 (No Write Down)
Biba	무결성 (Integrity)	하향 읽기 금지 (No Read Down), 상향 쓰기 금지 (No Write Up)
구획화	지식 한정 (Need-to-Know)	동일 등급 내 업무영역별 접근 분리

7.5 은닉 채널이란 무엇인가

■ **정의:**

시스템 설계자가 의도하지 않은 정보전달 통로.

즉, 공식적인 보안정책(Policy)을 우회하여 비밀정보가 전달되는 통로.

■ **핵심 특징:**

- 접근통제 정책을 회피.
- 로그에 남지 않음.
- 시스템 동작의 "부수효과(side-effect)"를 이용.

7.5 은닉 채널의 직관적 예시

■ **상황:**

두 사람이 벽으로 분리된 방에 있음.

말로 소통 불가하지만, 전등 스위치로 신호 전달 가능.

■ **신호 방식:**

- 불 ON → ‘1’
- 불 OFF → ‘0’
- 일정 패턴 = 메시지 전송

■ **→ 직접 통신 금지 정책이 있어도,**

“공유된 환경(전등)”을 통해 의사소통 가능.

7.5 은닉 채널의 구성 요소

구성요소	역할	예시
Sender (송신자)	정보를 은밀히 전달하려는 주체	해커 프로세스 A
Receiver (수신자)	신호를 수신하는 주체	프로세스 B
Shared Resource (공유 자원)	양쪽이 동시에 접근 가능한 시스템 자원	파일, 캐시, CPU
Observable Property (관찰 가능한 특성)	수신자가 인식 가능한 변화 요소	응답시간, 이름 변경, 점유율 등

7.5 은닉 채널의 구성 요소

유형	설명	예시
저장형(Storage Channel)	공유 자원의 상태를 바꿔 정보 전달	파일 이름, 캐시, 메모리 변수
타이밍형(Timing Channel)	처리 속도, 응답 시간, 이벤트 간격으로 정보 전달	CPU 점유율, 네트워크 지연시간

7.5 저장형 은닉 채널 예시

- **A 프로세스: 파일 이름 변경(temp → flag)**
- **B 프로세스: 변경 감지 후 '1'로 인식**
- **또 다른 예시:**
  - CPU 캐시 hit/miss 이용
  - 파일 존재 여부, inode 값 등
  - OS 변수의 상태 차이

7.5 타이밍 은닉 채널 예시

- **CPU 사용률로 신호 전달:**
  - 점유율 90% → ‘1’
  - 점유율 10% → ‘0’
- **응답시간 조절:**
  - 1초 대기 후 응답 → ‘1’
  - 즉시 응답 → ‘0’
- **네트워크 전송 간격으로 정보 인코딩 가능**

7.5 네트워크에서의 은닉 채널

■ TCP/IP 프로토콜의 구조적 취약점 활용:

- Reserved field (예약 필드)에 비트 정보 삽입.
- TTL(Time-To-Live) 값 변조.
- 패킷 크기/순서 패턴을 이용한 은닉 전송.

■ 실제 사례 – TCP 헤더 은닉

- TCP 헤더의 Reserved 비트 (6bit)
- 평소엔 000000해커는 이를 101011 등으로 변경 → 메시지 삽입

■ 특징:

- 로그에는 아무 이상 없음.
- IDS/Firewall 통과 가능.

7.5 사이드채널 공격과의 관계

구분	은닉 채널	사이드채널
목적	의도적 통신	비의도적 누출
매개	공유 자원	물리적 신호(전력, 소음 등)
예시	캐시·타이밍 기반 통신	CPU 전력분석, EM 방출 분석
공통점	보안정책 외부에서 정보가 새나감	

7.5 은닉 채널의 위험성

- **시스템 로그에 남지 않음.**
- **방화벽, IDS 등으로 탐지 어려움.**
- **암호화 채널 내부에서도 은닉 가능.**
- **내부자(Insider)에 의해 악용될 가능성 높음.**

- **은닉 채널 용량 계산**
  - 은닉 채널의 대역폭(Bandwidth)을 제한하면 피해 감소.
  - 예: 1 bit/sec 제한 시
    - 100MB 파일 유출 → 약 25년 소요
    - AES 256bit 키 유출 → 약 5분 소요

7.5 은닉 채널 탐지의 어려움

- **정상 동작처럼 보임 → 탐지 어려움.**
- **“이상 징후(Anomaly)” 기반 머신러닝 탐지만 가능.**
- **완벽한 차단 불가능 → 완화(Mitigation) 중심 전략 필요.**

방어 전략	설명
대역폭 제한	정보 전달 속도를 느리게 해 실질적 피해 최소화
무작위화(Randomization)	응답 시간·자원 스케줄을 랜덤화
로깅 강화	시스템 이벤트·접근 이력 상세 기록
동기화 차단	공유 자원 접근 시 타이밍 동기화 불가하게 설계

7.5 현대 보안에서의 은닉 채널

분야	은닉 채널 형태
클라우드	VM 간 캐시 공유를 통한 데이터 유출
IoT	전자파, 전력소모 패턴으로 신호 전송
AI 모델	학습 파라미터나 출력 확률을 통한 정보 누출

7.5 은닉 채널 탐지 연구

- 통계적 패턴 분석: 비정상적 분포 탐지
- 네트워크 타이밍 분석: 지연 패턴 기반 이상 탐지
- 머신러닝 기반 이상탐지: 정상 트래픽 대비 규칙성 판별
- 적응형 보안 시스템: 실시간 탐지 + 정책 자동수정

7.5 은닉 채널 탐지 연구

- 통계적 패턴 분석: 비정상적 분포 탐지
- 네트워크 타이밍 분석: 지연 패턴 기반 이상 탐지
- 머신러닝 기반 이상탐지: 정상 트래픽 대비 규칙성 판별
- 적응형 보안 시스템: 실시간 탐지 + 정책 자동수정

7.6 추론제어란?

- 정의: 합법적 질의(Query)·통계를 통해 민감정보를 간접 유추하는 것을 예방/완화하는 기술 집합.
- 목표: “데이터 접근은 허용하되, 결과로 인한 유출은 차단.”
- 적용 범위: 통계DB, BI 대시보드, 공개데이터 포털, 프라이버시 보존 ML(Privacy-Preserving ML).

7.6 추론의 예시

- **질의: “A부서 평균 급여?”**
- **A부서 인원 = 1명 → 평균 = 개인 연봉 노출.**
- **변형: A부서 2명일 때 평균+합계를 묶어 질의해도 개별값 역산 가능.**

- **추론 공격(Inference Attack)의 개념**
  - 직접 공격: 데이터 자체 탈취(침해).
  - 간접 공격: 부분 통계(평균·최댓값·카운트)를 다중 조합해 숨겨진 값 복원.
  - 기법: Tracker Attack, Sum/Count 차이 공격, 교집합·여집합 활용.

7.6 추론 제어의 목적

- 균형: 데이터 활용성(유용성) ↔ 프라이버시(보호성).
- 전략: 완전 차단이 아니라 누출 최소화(minimization), 위험 기반 접근.
- 원칙: 최소 공개(Least Disclosure), 필요 최소 집합(Need-to-Answer).

7.6 추론 제어 기법 ① — 질의 크기 제한

- **Cell Suppression / k-익명 집합 크기:**
  - 결과 집합 크기 < k (예: k=3)이면 응답 거부.

- **Small Cell Suppression:**
  - 테이블의 소수집단 셀은 마스킹('*').
- **Top/Bottom Coding:**
  - 극단값은 구간화("≥X", "≤Y").

7.6 추론 제어 기법 ② — N-응답, k% 지배 규칙

- N-응답 규칙: 결과에 N명 이하 데이터만 기여 → 응답 금지.
- k% 규칙: 한 하위그룹이 결과의 k% 이상을 지배하면 차단(예: 상위 1~2명이 평균 대부분 좌우).
- p%-규칙/Threshold Rule 등 조직별 변형.

7.6 추론 제어 기법 ③ — 무작위화(Randomization)

- 노이즈 주입: 결과 혹은 원자료에 작은 확률적 잡음 추가(가우시안/라플라스).
- 라운딩/버킷팅: 수치 반올림, 구간 집계.
- Swapping/Permutation: 행 일부 속성 치환(분포 유지).

- 목표: 개별 식별성↓, 집단 통계 정확도는 최대한 유지.

원래 급여	노이즈(±5%)	결과
5,000,000	4,800,000	ㅇ
6,200,000	6,450,000	ㅇ
4,700,000	4,900,000	ㅇ

7.6 차등 개인정보보호(Differential Privacy, DP)

- 핵심 아이디어: 특정 개인의 포함/제외 여부가 결과에 유의미한 차이를 내지 않도록 확률적으로 보장.
- 메커니즘: 라플라스/가우시안 메커니즘(질의 민감도 Δf 기반 노이즈).
- 프라이버시 예산 ε: 작을수록 보호 ↑ (노이즈 ↑), 정확도 ↓.

- 두 인접 데이터셋 $D1$,$D2$ (단 1명만 다름)에 대해,
  임의 결과 집합 S에 대하여

$$\Pr[M(D_1) \in S] \leq e^{\varepsilon} \Pr[M(D_2) \in S]$$

- Δf(민감도): 한 개인의 변경이 질의 결과에 줄 수 있는 최대 변화량.
- 조성(Composition): 다회 질의 시 ε가 누적 → 예산 관리 필수.

7.6 현실 적용 예시

- 브라우저 사용통계(예: 크롬): 클라이언트 측 랜덤화로 개별 사용패턴 은닉.
- 국가 통계(인구·가계조사): 표본 응답에 노이즈/억제 규칙 적용.
- AI 학습 데이터: DP-SGD(확률적 경사하강 + 클리핑·노이즈)로 학습 중 유출 방지.

7.6 추론 제어의 한계

- 정확도 ↔ 프라이버시 트레이드오프(ε 선택 난제).
- 과도한 노이즈 → 분석력 상실 / 과소 노이즈 → 재식별 위험.
- 쿼리 감시/중복·유사 질의 통제 없으면 우회 공격 가능.
- 완전한 익명화는 불가(보조정보 존재).

7.7 캡차의 등장 배경

■ **등장 배경:**

- 웹 서비스 자동화 공격(bot) 급증
- 계정 생성, 설문 조작, 트래픽 폭주 등의 문제 발생

■ **핵심 개념:**

→ 사람만 풀 수 있는 퍼즐로 봇을 차단

■ **핵심 목적:**

- 인간 사용자만 접근 가능하게 하는 '지능 필터'

7.7 튜링 테스트란?

- **제안: 1950년, 앨런 튜링(Alan Turing).**
- **목적: “기계가 인간처럼 사고할 수 있는가?”**
- **검증.방법: 인간 질문자가 두 응답자(사람/기계) 중 누구인지 구분하지 못하면 → 테스트 통과.**

- **역튜링 테스트(Reverse Turing Test)**
  - 튜링 테스트: “컴퓨터가 사람처럼 보이는가?”
  - 역튜링 테스트: “사람이 컴퓨터가 아님을 증명.”
  - → CAPTCHA = 역튜링 테스트의 구현.

7.7 CAPTCHA의 정의

- **Completely Automated Public Turing test to tell Computers and Humans Apart**

  → "사람과 컴퓨터를 구분하는 완전 자동화 테스트"

- **핵심 특징:**
  - 자동화(Completely Automated)
  - 공개형(Public)
  - 구별 목적(Human vs Machine)

7.7 초기 문자형 캡차

- **방식:**

  왜곡된 문자·숫자를 이미지로 제시 → 사람이 직접 입력.

- **원리:**

  컴퓨터의 시각 인식(OCR)이 인간보다 떨어졌던 시절의 취약점을 활용

- **예: “gK9Rt”와 같이 뒤틀린 문자 제시.**
- **특징:**
  - 사람 눈에는 명확
  - 컴퓨터 인식에는 잡음·왜곡

- **효과: 2000년대 초반까지 강력한 봇 차단수단.**

7.7 이미지형 캡차의 등장

- 형식: “자동차가 포함된 이미지를 모두 선택하시오.”
- 이유: 시각 인식은 인간의 강점 → 초창기 AI는 객체 구분 취약.
- 활용: Google, Facebook 등 대형 플랫폼 보편화.

7.7 reCAPTCHA의 탄생

- **2007년 구글 인수 후 발전:**
  - 사용자의 입력 데이터를 OCR 학습 데이터로 재활용.
  - 사용자는 '보안 + AI 학습' 동시 수행.

- **v2, v3로 진화.**

7.7 reCAPTCHA v2: “나는 로봇이 아닙니다”

- **단순 클릭 기반.**
- **분석 요소:**
  - 마우스 이동 곡선
  - 클릭 속도·패턴
  - 브라우저·IP 이력
- **결과: 행동 기반 인증(Behavioral Authentication).**

7.7 reCAPTCHA v3: 완전 비가시형

- 비가시형(invisible) 캡차: 사용자가 아무 것도 안 함.
- AI 판단: 페이지 상의 행동·스크롤·체류시간·네트워크 패턴을 기반으로 'Human Score' 계산.
- 결과: UX 개선 + 지속적 보안 모니터링.