1. **운영체제 보안 개관**

개관

운영체제 관점에서 보호와 보안에 대한 개념 운영체제에서 보안에 대해 할 수 있는 일

학습목표

1. **보호와 보안의 개념을 이해**
2. **운영체제에서 보안이 달성하고자 하는 목표를 이해**
3. **운영체제에 대한 침투유형을 이해**
4. **운영체제 수준에서의 보안기법과 보안 모델을 이해**
5. **안전한 운영체제에 대해 이해 주요용어**

보호 보안 보호영역 변조 위조 트랩도어 비밀 채널 암호화 공개키 개인키 인증 식별 접근제어

* 1. **보안의 개요 [225]**

과거의 컴퓨터, 보안

사용자 실수로부터 시스템을 보호, 프로세스가 다른 프로세스의 동작을 방해하는 것을 막기 현대의 컴퓨터, 보안

컴퓨터에 대한 접근권을 불법적으로 얻고, 이를 이용하여 컴퓨터의 정보를 유출하거나, 컴퓨터를 사용할 수 없게 만들거나, 다른 컴퓨터에 침투하는 데 이용하는 등의 악용이 가능

* + 1. **보호와 보안 보호**

실행되는 각 프로세스에 대해 자신이 사용하는 자원이 다른 프로세스에 영향을 받지 않도록 하는 것

각 프로세스가 프로세서의 사용시간, 사용하는 데이터, 자료가 관리되는 작업, 사용하는 장치 등을 대상으로 컴퓨터 시스템 내 부 자원 각각의 영역을 보장

보안

원격 접속하는 사용자에 대해서도 인증, 암호화 매커니즘 등을 통해 사용자가 누구인지 정확하게 인식하고, 사용자가 권한을 가진 자원만 이용해 허락된 일만 할 수 있도록 보장

=> 시스템이 정상적으로 동작함으로써 1) 저장된 자료가 결함이 없도록 하며, 2) 시스템을 신뢰할 수 있도록 하는 것 보호와 보안의 목적

시스템 프로세스와 사용자 프로세스에 대해, 자신이 권한을 가지지 않은 자원에 대해 접근하는 일을 막는 "접근 제어"를 규정 하기 위한 작업

* + 1. **보호영역**

**컴퓨터 시스템은 프로세스와 객체(object)의 집합이다. 프로세스**

자신의 목적을 수행하기 위해 필요한 자원을 접근 과거에는 모든 자원을 모든 프로세스가 사용하게 함

특정 프로세스가 자신에게 필요 없는 자원을 붙잡으면 다른 프로세스가 작업을 진행할 수 없고,

시스템 프로세스 등 상대적으로 중요성이 높은 프로세스가 잘못 동작하게 되어 전체 시스템이 붕괴될 수 있음. 프로세스에 자원을 할당하는 기본 원칙

"각 프로세스는 자신이 작업을 수행하는 데 필요한 최소한의 자원만 사용할 수 있다"

**프로세스 P가 A를 호출했을 때 A가 알 수 있는 내용은 자신이 원래 알고 있던 것 + P가 전달한 매개변수만 보호영역 (protection domain)**

각 자원에 대해 해당 프로세스가 할 수 있는 연산이 무엇인지를 정의

각 프로세스는 자신의 보호영역 하나를 가지고 있으며, 이 안에 포함된 자원은 이 프로세스가 쓸 수 있다. 접근권한

어떤 프로세스가 각 객체에 대해 연산을 수행할 수 있는 능력 영역

접근권한의 집합

**각 객체(자원)마다 <객체 이름, 권한 집합>의 순서쌍으로 주어짐 프로세스 보호영역 접근권한 <파일F, {읽기, 쓰기}>**

같은 객체에 대해 보호영역에 따라 할 수 있는 일이 다름. 각 프로세스에 자신이 필요한 객체만 할당할 수 있음

* 1. **보안의 개요 (2) [228]**
     1. **운영체제 보안 운영체제에서 보안**

운영체제에 포함된 취약점을 이용하여 생길 수 있는여러 가지 공격으로부터 운영체제가 관리하는 자원을 불법적으로 이용하 는 것을 막는 정책과 기법

현대 컴퓨터 운영체제에서는 모든 자원(CPU, 메모리, 프린터, 모니터, 저장장치 등)을 다수의 사용자가 동시에 사용하고, 네 트워크를 통해 원격으로 접속하는 것 또한 허락하기 때문에 사용자들이 안정적으로 자원을 이용할 수 있고, 저장된 정보가 소 실되거나 외부로 유출되지 않도록 하는 것이 매우 중요

적절한 1) 접근제어 설정과 2)정보의 암호화를 통해 구현할 수 있으며, 이렇게 구현된 시스템이 실제로 보안을 제공한다는

것을 보일 필요가 있음.

시스템 접속 및 자원에 대한 신뢰할 수 있는 기록을 남겨서 이를 통해 침입자를 추적할 수 잇어야 한다. 운영체제 보안 개념

# 객체(object)

운영체제가 관리하는 모든 자원 주체

객체를 사용하고자 하는 프로세스 (또는 사용자) 주체는 자신의 임무를 수행하기 위해 객체를 사용

사용

**읽기, 쓰기, 실행하기 등 권한**

객체를 사용하려면 주체가 객체를 사용할 수 있는 권한을 가지고 있어야 한다.

각 주체에 객체에 대한 권한을 적절히 설정하고, 권한이 없는 주체가 객체를 사용하는 것을 막음으로써 보안을 구현할 수 있다.

**운영체제 보안에서 달성하고자 하는 기본 목표 기밀성(confidentiality)**

운영체제가 관리하는 자원들은 이를 탐지할 수 있어야 한다.

# 가용성(availability)

1) 만약 주체가 합법적으로 사용할 수 없다면 사용되어서는 안되며, 2) 만약 사용되었다면

운영체제가 관리하는 자원들은 만약 주체가 사용하는 데 문제가 없다면 반드시 사용할 수 있어야 한다. 무결성(integrity)

객체에 저장되어 있는 정보가 항상 정확하다는 뜻이다.

무결성을 얻기 위해서는 객체는 합법적인 경우에만 수정될 수 있어야 한다. 운영체제에서 침해

정보침해

세 가지 목표가 달성되지 못하고, 안전하게 유지되어야 할 정보가 불법적으로 읽히거나 다른 값이 덮어 쓰이는 것 현대 운영체제에서의 "침해"

공격자가 의도적으로 다른 사용자의 권한을 도용하여 정보를 유출하거나 덮어 쓰는 경우 주로 소프트웨어 결함(vulnerability)을 이용하는 형태

**예) 스택 오버플로(stack overflow) 공격**

공격자는 스택에 스택의 크기보다 큰 값을 넣어서 코드 및 데이터 영역에 원하는 값을 쓰는 것을 시도 스택에 값을 넣을 때, 스택의 크기보다 작은 값을 넣는지 확인하지 않는 경우 발생

시스템 권한을 가지고 동작하는 프로그램을 이용하여 다른 프로그램을 실행시키게 할 수 있음. (두 번째) 실행된 프로그램도 시스템 권한을 가지기 때문

**운영체제를 통한 침해 가로채기(interception)**

기밀성에 대한 공격

**공격자가 허락받지 않은 컴퓨터 자원을 접근하는 경우 흐름 차단(interruption)**

가용성에 대한 공격

**시스템 일부를 파괴하거나 사용할 수 없게 만듦 변조(modification)**

무결성에 대한 공격

**공격자가 허락받지 않은 자원에 접근하여 기존에 있던 데이터 내용을 바꾸는 경우 위조(fabrication)**

무결성에 대한 공격

공격자가 허락받지 않은 자원에 접근하여 기존에 없던 불법적인 정보를 삽입 운영체제를 통한 대표적인 침해유형

**트로이 목마(Trojan horse)**

숨겨진 기능이 있는 프로그램을 사용자가 실행하게 만들어서 이 사용자의 권한을 이용하여 시스템에 침투

**(vs 웜, 바이러스) 일반적으로는 자기 자신을 복제하는 기능 X**

**트랩도어, 백도어(trapdoor, backdoor)**

**정상적인 인증절차나 암호화를 피해 갈 수 있는 비밀 통로 비밀 채널(covert channel)**

데이터를 주고받을 수 없는 프로세스 사이에서 정상적인 데이터 전송 메커니즘이 아닌 다른 방법으로 정보를 알아내는 형태의 공격

종류

**타이밍 채널(timing channel)**

네트워크를 통해 전송되는 패킷의 지연시간을 측정하여, 지연이 심한 순간을 정보가 처리되는 시간으로 추론하여 알아내는 것

**저장장치 채널(storage channel)**

**특정한 위치에 정보를 쓰고 읽음으로써 프로세스 사이에 정보를 주고받는 것 웜(worm)과 바이러스(virus)**

웜

그 자신이 하나의 악성 코드 프로그램으로, 자기 자신을 복사하여 다른 컴퓨터에 전파시키는 능력 바이러스

스스로를 복사 (웜) + 다른 프로그램을 '감염'시켜서, 자신의 코드를 다른 프로그램에 추가시켜서 웜과 비슷하게 계 속 다른 프로그램에 전파

* 1. **보안정책 및 보안 매커니즘 보안정책**

보안을 어떠한 관점에서 무엇을 행할 것인가를 결정 보안 메커니즘

보안을 어떠한 방법으로 결정할 것인가

* + 1. **보안정책**
       1. **권한부여(authorization)**

어떤 주체가 어떤 객체를 어떻게 액세스할 수 있는지를 결정

**운영체제의 모든 주체와 객체는 식별(identification) 및 인증(authentication)이 가능해야 함 주체의 자원에 대한 접근 제어를 가능하게 함. 보안등급을 부여할 수 있도록 함.**

식별

어떤 주체나 객체가 무엇인지 신분을 알아내는 것 인증

그 주체와 객체가 정말로 자신이 주장하는 그 주체와 객체가 맞는지 확인하는 것

# 임의적 접근제어(Discretionary Access Control: DAC)

관리자 또는 자원 소유자가 보안 관리자의 개입 없이 주체에 자원의 접근권한을 부여할 수 있다. 접근권한을 부여받은 주체는 다른 주체에 이 권한을 전달할 수 있다.

자원 보호보다 자원의 공유가 중요할 때 적합

* + - 1. **강제적 접근제어(Mandatory Access Contro: MAC)**

운영체제가 각각의 주체가 객체에 대한 접근이나 연산을 할 수 있는지 여부를 정해진 규칙과 비교하여 이를 허가할지 여부를 결정

보안정책이 중앙에서 관리되며, 각 사용자는 이 정책을 넘어서는 행동을 할 수 없다.

주체에는 허가등급, 객체에는 비밀등급이 주어지며, 접근 요청이 올 때마다 이 둘을 비교하여 허가 여부를 결정한다.

임의적 접근제어와 강제적 접근제어의 동시 적용도 가능

시스템을 DAC가 사용되는 부분과 MAC이 사용되는 부분으로 나누어 사용 가능 임의적 접근제어와 차이점

모든 객체와 주체에 대해 중앙에서 일정한 규칙을 통해 접근을 제어, 사용자는 권한을 양도하거나 규칙을 어기고 자원을 접 근하거나 규칙 수정할 수 없음

장점

보안 관리자가 시스템 전체에 대한 보안정책을 구현하고 강제할 수 있기 때문에 보안에서 효율적 단점

보안등급을 강제하는 과정에서 자원의 공유가 복잡해질 수 있음

* + - 1. **역할 기반 접근제어(Role-Based Access Control: RBAC)**

**역할(role) 개념을 사용하여 사용자와 그들의 권한(permission)을 효과적으로 관리할 수 있음**

**RBAC 개념 모델 4요소 사용자(U)**

사람이나 자동화된 에이전트일 수 있음.

한 사용자는 여러 가지 역할을 가질 수 있음. 역할(R)

권한 수준을 정의하는 일의 기능이나 이름 한 역할은 여러 가지 권한을 가질 수 있음

**권한(P)**

특정한 형태로 자원에 대한 접근을 허락하는 것 한 권한이 여러 역할에 주어질 수 있음

**세션(S)**

**U, R, P 사이의 대응 관계 특징**

**사용자 할당(User Assignment: UA)과 권한할당(Permission Assignment: PA)이 다대다 관계임을 알 수 있음 제약 조건(constraints)**

**UA와 PA에서 지켜야 할 규칙 역할계층(Role Hierarchy: RH)**

**역할의 계층적인 구조와 제약조건 부분순서(partial ordering)의 특징을 가짐**

상위 역할은 계층을 통해 하위 역할의 권한을 모두 가진다.

**RBAC 모델 3가지 기본 규칙 역할할당(role assignment)**

**주체는 역할이 주어졌을 때만 권한을 사용할 수 있음 역할권한부여(role authorization)**

**주체가 어떤 역할을 행사하려면 권한을 부여받아야 함 권한부여(permission authorization)**

주체는 현재 행사하고 있는 역할에 권한이 주어졌을 때만 권한을 사용할 수 있음

# 최소권한(least privilege)

사용자는 임무를 수행하기 위해 필요한 최소한의 권한을 받아야 하고, 임무가 끝나면 이 권한을 반환해야 한다.

# 감사(auditing)

컴퓨터 시스템에서 발생한 이벤트는 해당하는 내용에 대한 정보가 기록되어야 하고, 특별한 일이 없는 한 이 정보는 변조도지 ㅣ 않고 보존되어야 한다.

감사과정을 통해 시스템의 로그(log) 파일을 조사하여 시스템에 발생한 이벤트를 추적하고, 침해 사고 등이 발생했는지 여부 를 확인하고 감시해야 한다.

* + 1. **보안정책 및 보안 메커니즘 [236]**

1. **보안 메커니즘**
   1. **주체 및 객체의 레이블 부여 메커니즘**

시스템의 주체와 객체마다 유일한 식별자를 부여하여 시스템 안에서 서로 구별이 가능 레이블 부여는 접근제어를 구현하는 데 필요하며, 특히 강제적 접근제어를 구현하는 데 필요

* 1. **안전한 암호 메커니즘 비밀키 암호화 알고리즘**

암호화 속도가 상대적으로 빨라서 대규모 데이터의 암호화에 적합 공개키 암호화 알고리즘

암호화하는 데 사용하는 키와 이를 푸는 데 사용하는 키가 다르기 때문에 암호키를 공개할 수 있고 이를 이용하여 인증이 가능함

* 1. **안전한 인증 메커니즘**

**식별과 인증을 위한 여러 가지 방법 중 '패스워드'가 유명 다요소 인증(multi-factor authentication)**

사용자 인증에 두 가지 이상의 방법을 요구하는 것

**UNIX 운영체제**

사용자는 내부적으로 UID(User ID)라는 숫자 식별자로 구별됨. 특정 작업을 위한 사용자가 존재할 수 있으며, 작업에 필 요한 자원에 대한 접근 제어를 쉽게 해준다. 이 사용자들은 일반적인 사용자와 분리되어 관리됨

* 1. **임의적 접근제어를 위한 메커니즘**

**UNIX의 파일 시스템**

UNIX의 경우 장치 등 지원을 파일로 고려할 수 있기 때문에 이 접근제어는 거의 모든 자원에 적용 파일의 소유자가 각 파일마다 세 가지 주체를 대상으로 세 가지 연산을 허락할지 금지할지 결정할 수 있음 주체

자기 자신, 자신이 속한 그룹, 나머지로 나뉨 가능한 연산

**읽기, 쓰기, 실행하기**

(다른 운영체제와 마찬가지로) 접근제어 리스트 메커니즘을 사용

* 1. **보안등급 관리 메커니즘**

사용자에게 다양한 종류의 보안등급을 부여하여 강제적 접근제어가 가능하게 한다

자원의 소유자가 보안등급을 관리하는 것이 아닌, 전체 시스템의 보안을 관리하는 보안 관리자만 보안등급을 관리 보안등급을 적용하기 전에 레이블 부여 메커니즘을 먼저 사용

* 1. **기록 파일 관리 메커니즘**

시스템에서 발생한 이벤트에 대한 기록을 안전한 위치에 보관하고, 접근제어와 암호화를 통해 구현한다.

* 1. **운영자 권한의 분산 메커니즘**

이를 수정할 수 없게 한다.

시스템 관리자의 권한을 세분화하여 목적에 따라 해당하는 역할을 담당하는 운영자에게 줄 수 있다. 운영자

자신의 목적에 해당하는 권한을 사용하며, 다른 영역에 대해서는 권한을 행사할 수 없다.

1. **하드웨어 보호를 위해 사용되는 방법 오류 처리**

사용자 프로세스가 불법적인 명령을 수행하거나 허락되지 않는 기억공간에 접근하는 등 오류가 발생하면, 이 오류를 처리해야 시스템이 정상상태를 유지할 수 있음.

오류는 예외 사항이 발생한 것으로 간주하여 해당하는 오류를 처리하는 루틴으로 제어를 옮기게 된다.

**예외(exception), 트랩(trap), 인터럽트(interrupt)는 프로세스를 잠시 중단시키고, 해당하는 오류를 처리하는 운영체제 루 틴을 호출하는 과정을 말한다.**

**예외(exception) 일반적인 명칭**

**트랩(trap)**

운영체제가 발생시키는 예외

**운영체제가 제공하는 시스템 콜(system call)을 호출하기 위해서도 트랩을 이용하기도 함 인터럽트(interrupt)**

**하드웨어에서 발생시키는 예외 인터럽트 벡터(interrupt vector)**

예외 발생 시 호출할 루틴의 주소가 저장된 부분 심각한 오류가 발생했다면?

운영체제는 해당 프로세스를 비정상적으로 종료시켜 시스템을 보호한다.

시스템 콜을 호출하기 위해 트랩을 이용했다면, 프로세스의 동작에 문제가 없는 것이므로 중단된 시점 이후부터 작업을 재 개한다.

* 1. **이중 모드 연산 (dual mode operation)**

배경

사용자 모드는 자신에게 허용된 권한 이상을 행사해서는 안 된다.

대부분의 경우 프로세스는 사용자 모드에서 동작, 반드시 필요한 경우에만 커널 모드로 동작 용어

**사용자 모드 = 비특권 모드(unprivileged mode)**

**커널 모드 = 특권모드(privileged mode) = 시스템 모드(system mode) = 모니터 모드(monitor mode)**

둘 이상의 동작 모드를 하드웨어에서 구현하려면, 현재의 모드가 어느 상태인지를 알려주는 모드 비트(mode bit)가 최소한 한 비트 필요하며, 프로세서도 권한이 다른 두 가지 이상의 상태로 동작하여 운영체제의 모드에 대응할 수 있게 한다.

# 특권 명령 (privileged instruction)

시스템의 상태를 바꾸어 보안에 위험을 줄 수 있는 명령 그 외에는 일반 명령

운영체제에서만, 그리고 커널 모드에서만 호출되어야 함. 만약 특권명령이 사용자 모드에서 호출되었다면?

무언가 잘못된 것으로, 하드웨어는 운영체제에 트랩을 발생시킨다.

* 1. **메모리 보호(memory protection)**

각 프로세스는 자신의 주소공간을 가지는데, 모리를 접근할 수 없어야 한다.

이 주소공간은 다른 프로세스 또는 운영체제와 분리되어 있고, 주소공간 외의 메

**메모리 관리 장치(memory management unit)**

하드웨어에 붙어 있는 물리적 메모리에 이 주소공간을 대응시키는 것을 담당 운영체제를 보호

프로세스가 접근하려는 메모리 주소가 물리적 메모리에서 운영체제가 사용하는 영역일 경우 트랩을 발생시켜 프로그램 을 중단시킴

**이를 구현하기 위해 2개의 레지스터를 사용할 수 있음 기준 레지스터(base register)**

**프로세스가 접근할 수 있는 물리적 메모리 주소의 최솟값 한계 레지스터(limit register)**

기준 레지스터부터 시작해서 프로세스가 접근할 수 있는 주소 범위의 길이

프로세스는 [기준 레지스터의 값, 기준 레지스터의 값 + 한계 레지스터의 값] 범위의 물리적 메모리를 사용할 수 있 다

# CPU 보호 (CPU protection)

**타이머(timer)**

사용자 프로그램이 무한루프에 빠지거나 제어가 운영체제로 복귀하지 않는 것을 방지하기 위해 일정한 간격으로 인터럽트 를 발생시킴

타이머의 인터럽트를 받은 운영체제의 스케줄러는 1) 이 인터럽트를 받고 중단된 프로세스가 자신에게 할당된 시간을 다 썼는지 확인한다. 2) 만약 시간을 다 사용한 상태에서 자신의 차례가 오기를 기다리는 다른 프로세스가 있다면, 이 프로세 스 중 하나로 제어를 옮긴다. 3) 중단된 프로세스는 다음 자기 차례가 되면 중단된 시점부터 다시 진행된다. 4) 또한 인터럽 트가 발생한 횟수를 세면 현재 시각을 측정할 수도 있다.

타이머는 중요한 역할을 담당하기 때문에 타이머의 동작을 수정하는 명령은 반드시 특권명령이어야 한다.

* 1. **입출력 보호(I/O protection)**

입출력 작업은 1) CPU 명령을 수행하는 것에 비해 시간이 많이 걸리며, 2) 또 한 프로세스의 입출력에 다른 프로세스가 영향

을 미쳐서는 안 된다.

입출력은 커널 모드에서 동작하게 하며, 사용자 프로세스가 입출력을 이용하기 위해서는 시스템 콜을 이용하게 함

입출력을 요청한 프로세스는 일단 중단되며, 시스템 콜을 통해 커널 모드로 진입한 후 주변장치에 대한 접근을 통해 입출

력을 수행한다. 작업이 완료되면 결과를 대기 중인 프로세스에 전달하고 사용자 모드로 돌아온다.

1. **보안정책 및 보안 매커니즘 (2) [241]**
2. **암호화 용어**

암호화

원래 정보를 변형하여 특정한 정보를 알지 못하면 변형된 정보에서 원래 정보를 얻어 낼 수 없게 하는 과정이다. 복호화

변형된 정보에서 원래 정보를 구하는 과정

키

암호화 및 복호화에 필요한 특정한 정보 대칭키 암호 시스템

암호화하는 데 필요한 키와 복호화하는 데 필요한 키는 동일 공개키 암호 시스템

암호화하는 데 필요한 키와 복호화하는 데 필요한 키가 다름

현대의 암호 시스템

복호화를 완전히 불가능하게 하는 것보다, 해독에 걸리는 시간을 오래 걸리게 하여 복호화의 의미를 없애는 것을 목표로 함 암호화의 종류

암호 시스템

사용자에 의해 선택되는 1) 비교적 짧고 알려지지 않은 숫자 또는 문자열의 조합으로 이루어진 "암호키"와 2) 이 키를 이 용하여 입력받은 정보를 일정한 규칙 및 단계를 거쳐 암호화하는 "암호화 알고리즘"으로 구성된다.

비밀키 암호 시스템 = 대칭키 암호 시스템 암호화와 복호화에 동일한 키를 사용

# 예시) AES(Advanced Encryption Standard)

암호화 및 복호화 속도가 빠름 공개키 암호 시스템

암호키와 복호키가 서로 다르며, 암호키는 모두에게 공개하고 복호키(개인키)는 비밀로 한다.

**RSA(Rivert-Shamir-Adleman) 암호 시스템 전자서명 등 응용이 가능**

1. **비밀키 암호 시스템 (=대칭키 암호 알고리즘)**

비밀키는 암호화와 복호화를 할 수 있는 사람들 사이에서만 공유됨. 키를 공유하는 문제

비밀키 암호 시스템을 이용하여 암호화된 데이터를 교환하려면, 먼저 키를 만들고 이 키를 공유하는 문제가 발생

**사전 공격방법(dictionary attack)을 피해야 함 : 난수생성기를 이용하여 만든 키를 위 조건을 피해가도록 검사하여 사용 공개키 암호 시스템과 결합하여 키를 공유한다.**

장점

암호화와 복호화 속도가 빠르다. 대량의 데이터를 암호화와 복호화할 때는 유일한 방법이 된다. 암호화와 복호화에 필요한 대부분의 연산이 비트 단위의 AND, OR, SHIFT 연산으로 이루어져 있음

1. **공개키 암호 시스템 = 비대칭키 암호 시스템 암호화에 사용되는 키(=공개키)**

복호화에 사용되는 키(=개인키)

비밀키 암호 시스템이 비밀키가 비공개된 것으로 기밀성을 얻을 수 있다면, 공개키 암호 시스템에서는 개인키를 가지고 있는 사 람만 내용을 알 수 있다. (=> 기밀성)

전자서명

1) 모두가 알고 있는 데이터에 대해 서명하려는 사람은 자신의 개인키로 복호화를 수행 2) 다른 사람들은 공개된 암호키를 통해 다시 암호화를 시도할 수 있고 암호화의 결과물은 원래 데이터가 되며, 이 일을 할 수 있는 사람은 개인키를 가지고 있 는 사람밖에 없기 때문에 서명과 유사한 형태로 해당하는 사용자의 인증이 가능함

장점

짧은 길이의 키로부터 더 높은 수준의 보안을 실현 전자서명 등의 응용이 가능

단점

암호화와 복호화 과정에서 복잡하고 시간이 많이 걸리는 수학 연산이 필요 => 암호화와 복호화에 시간이 더 걸림

대칭키 암호화 시스템과 비대칭키 암호화 시스템의 장점을 조합하여 성능과 보안 모두 얻을 수 있다.

* 1. **A가 먼저 비밀 정보를 생성한 후 B의 공개키를 이용하여 암호화하고, B에게 전달함. B의 공개키는 A를 포함한 모두에게 알 려져 있음.**
  2. **B도 비밀 정보를 생성하고, 이를 A의 공개키를 이용하여 암호화하고 A에게 전달한다. A의 공개키도 B를 포함한 모두에게 알 려야 함.**
  3. **A와 B는 자신이 수신한 정보를 개인키를 이용하여 복호화하고, 이 정보로부터 비밀키를 생성한다.**

비밀키를 생성하는 알고리즘은 A, B 모두 사전에 약속한 방법이 있고, 어떤 정보 입력해서 비밀키를 만들 것인지만 서로 통신 을 주고받는다고 가정

* 1. **이제 A와 B는 동일한 비밀키를 알고 있다.**

비밀키 자체는 네트워크를 통해 전송되지 않으며, 비밀키를 만들 수 있는 사람은 A, B밖에 없기 때문이다.

* 1. **A와 B는 공유한 비밀키를 이용하여 비밀키 암호 시스템으로 데이터를 주고 받음.**

현대의 보안 프로토콜인 SSL, IPsec 등 대부분은 위에서 설명한 절차를 이용하여 비밀키를 교환하여 통신함

1. **운영체제 보안 모델 [245]**

보안 모델

컴퓨터 시스템과 사용자의 보안을 설명하는 추상적인 모델로, 보안정책을 어떻게 명시하고 이를 지키게 할 것인지 다룸.

**이 책 보안 모델은 유한상태기계(finite state machine)을 이용하지만 계산 모델, 분산 컴퓨팅을 이용한 보안 모델도 있음.**

* 1. **참조 모니터 모델**

주체가 객체를 접근하여 연산하는 과정에서 참조 모니터가 접근제어를 수행하는 모델

주체와 객체 사이의 접근 권한을 정의한 데이터베이스가 있고, 참조 모니터는 이 데이터베이스를 참조하여 보안정책을 수행 참조 모니터는 언제나 수행되어야 하며, 공격자가 모니터의 동작을 바꿀 수 없어야 한다.

특징

참조 모니터 모델은 주체와 객체 사이에서 단순 접근의 허용 여부만 결정하는 단일 레벨의 보안 모델로 간주할 수 있다. 접근제어를 위해서는 보안 데이터베이스에 저장되어 있는 주체와 객체의 보안등급을 각각 참조하게 된다.

특정 주체가 특정 객체에 대해 접근할 수 있는지 여부만 판단한다.

주체가 접근한 객체에 포장된 기밀 정보를 유포하는 것을 막을 수 없기 때문에 정보의 안전한 흐름을 보장할 수 없다.

* 1. **정보 흐름 모델**

정보의 유형에 따라 정보가 흐르는 방향을 제어하는 모델

모든 허가된 정보 흐름을 허용하며, 모든 허가되지 않은 정보 흐름을 방지한다. 정보 흐름

동일한 분류 수준에 있는 주체와 객체 사이, 그리고 서로 다른 분류 수준에 있는 주체와 객체 사이일 수 있음.

* + 1. **벨 라파듈라(BLP) 모델**

# 기밀성 유지에 초점 (write up, read down) = (no write down, no read up)

**쓰기 접근은 객체의 보안 수준이 주체의 보안허가 수준보다 높거나 같아야 가능함. 단, 신뢰되는 주체(trusted subject)는 이 규칙에 제약받지 않는다.**

읽기 접근은 객체의 보안 수준이 주체의 보안허가 수준보다 낮거나 같아야 가능하다.

**중요도에 따라 극비(top secret), 비밀(secret), 미분류(unclassified) 또는 공개(public) 등의 레벨로 나누고 있다. 특징**

'안전한 상태'를 먼저 정의하고, 접근 및 연산을 수행할 때마다 안전한 상태가 계속 보장되는지를 증명하여 시스템이 항상 안전한 상태에 있음을 보임

접근 및 연산이 항상 정의된 보안정책을 따른다. 장점

정보의 기밀성을 고려함 단점

무결성은 쓰기를 제어하는 수준에서만 고려한다.

특히 신뢰되는 주체는 쓰기가 제어되지 않기 때문에, 면 무결성이 깨질 수 있다.

비밀 채널에 대한 설명이 부족하다.

보안등급이 동적으로 변화하는 환경에 적용하기 어렵다.

* + 1. **비바(Biba) 모델**

= 비바 무결성 모델

낮은 등급에서 높은 등급으로 쓰기가 가능하여 관리가 소홀하

주체와 객체를 무결성 등급에 따라 같은 수준끼리 그룹으로 묶는다. 모든 주체는 자신보다 높은 등급 객체에 쓰기 연산을 할 수 없다.

권한이 없는 주체가 데이터를 수정하는 것을 막고, 권한이 있는 주체가 권한이 없는 주체가 수정한 데이터를 사용하는 일을 막아 무결성을 보장한다.

# read up, write down = (no read down, no write up)

**현재 운영체제 중 FreeBSD, Linux 등 내부적 구현**

1. **보안 커널 [249]**

**보안 커널(security kernel)**

기존 운영체제 커널에 보안 기능을 통합시킨 것

자주 수행되고 중요한 일을 커널에 둠으로써 커널 공간과 사용자 공간 사이의 전환을 줄여서 시스템의 성능을 높일 수 있다. 보안 커널을 사용한 운영체제

**컴퓨터 사용자에 대한 식별 및 인증, 임의적 접근제어, 강제적 접근제어, 객체 보호, 침입탐지(intrusion detection) 등의 보안 기능 요소를 갖추어야 함.**

**ex) 리눅스 OS 커널 => 침입 탐지, 강제적 접근제어 등의 기능보유 전통적인 컴퓨터 시스템 구조**

하드웨어, 운영체제 및 응용프로그램으로 구성됨

**각 계층은 자신의 바로 아래 계층만 접근할 수 있고, 사전에 정의된 계층 사이를 연결하는 기능만 사용할 수 있음. (안전/정상적인 경우) 운영체제와 하드웨어는 보안관련으로 보안경계(security perimeter) 내부에 위치**

응용 프로그램은 이미 정의된 시스템 호출을 통해 보안경계를 통과하여 운영체제 접근함 사용자들은 시스템 외부에서 운영체제와 직접 통신하거나 응용 프로그램을 통해 시스템에 접근한다.

# TCB(Trusted Computing Base)

오류가 약점 등으로 인해 정상적으로 동작하지 않을 경우, 시스템 전체의 보안에 문제가 생길 수 있는 하드웨어, 펌웨어, 소프트웨 어, 물리적 설치장소, 보안정책 등의 집합

TCB 밖 구성 요소의 문제는 해당하는 요소에 대한 보안에만 문제 발생

**TCB의 역할**

기본적인 보호환경 제공

운영체제의 기본적인 작업(프로세스 활성화, 실행영역 교환, 메모리 보호, I/O 연산 등)에 대한 보안성 및 무결성을 감시 참조 모니터를 구현한 형태

보안 커널 정리

**TCB의 하드웨어, 펌웨어, 소프트웨어 요소**

시스템 자원에 대한 접근을 통제하기 위한 기본적인 보안 절차를 구현한 컴퓨터의 중심부

요약

* 1. **운영체제 보안에서 달성하려는 기본 목표는 기밀성, 가용성, 무결성이다.**
  2. **정보침해 위협 요소로는 가로채기, 흐름 차단, 변조, 위조 등이 있다.**
  3. **임의적 접근제어에서는 각 사용자가 자신이 보유하고 있는 자원에 대한 접근권한을 자신의 자율성 판단에 따라 부여하고, 자신이 가진 접근권한을 다른 사용자에게 전달할 수 있다.**
  4. **강제적 접근제어는 각 객체의 비밀등급, 개별 사용자의 허가등급에 따른 중앙의 규칙에 따라 제어가 이루어진다.**
  5. **역할 기반 접근제어에서는 사용자는 역할의 멤버가 됨으로써 권한을 배정받는다.**
  6. **비밀키 암호 시스템은 암호 시스템의 보안이 암호화와 복호화에 사용되는 키를 아는 사람만 정보를 공유할 수 있다는 사실을 이용 한다.**
  7. **공개키 암호 시스템에서는 암호화와 복호화에 사용되는 키가 다르며, 공개되지 않는다.**

암호화에 사용되는 키는 공개되고 복호화에 사용되는 키는

* 1. **참조 모니터는 주체와 객체의 접근궎나을 정의한 데이터베이스를 참조하여 보안정책을 수행한다.**
  2. **벨-라파듈라 모델에서는 기밀성에 중점을 두어, 상위 보안 수준의 정보가 하위 보안 수준으로 흐르는 것을 막는다.**
  3. **비바 모델에서는 무결성에 중점을 두어, 하위 보안 수준의 정보가 상위 보안 수준으로 흐르는 것을 막는다.**