Syllabus

2012년 3월 10일 토요일
오후 1:11

Security Engineering(보안공학)

1. **Objectives:** to tackle an information protection problem by ***(1) drawing up a threat model, (2) formulating a security policy,*** and ***(3) designing specific protection mechanisms*** to ***(4) implement the policy***

   여기서 (2)는 requirement analysis이며, 요구사항 분석결과에 따라 모든 것이 달라지기 때문에 제일 중요하다.

2. **Software Engineering:** 소프트웨어의 위기로 인해 등장한 소프트웨어 공학, 생산성을 증가시킬 목적으로 1968년 NATO Science Committee에서 처음 정의됨. Defined SE as "the applications of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software", SW개발인력이 많아지면 얼마나 빨리 생산이 가능한지에 대한 SDLC와 같은 체계화된 개발과정이 필요했다. 체계적으로 SW를 생산하고 팀 단위 협업하기 위한 체계와 성과의 측정을 위한 방법으로서 소프트웨어 공학이 필요성이 제기되었음

3. **Traditional Software Engineering**

- SDLC: Requirements -> Design -> Implementation -> Testing -> Deployment -> Maintenance
- Sec. E: Security Policy -> Design -> Implementation -> Testing -> Deployment
    - Testing: Process evaluation(CMM, SW-CMM), Product evaluation(CC, CMVP)
    - SW-CMM은 SW개발조직의 성숙도를 말함
- Security Engineering + Security = Security Engineering

4. **Motivation:** SW위기와 더불어 유지보수비용 급증, 보안결함을 초기단계에서 일찍 찾을수록 비용은 감소한다는 연구결과가 있었음(요구사항 분석단계 1: 유지보수단계 100의 비용차이)

5. **GOAL**
    - Security flaws are identified only at the later stages of the application lifecycle(보안공학의 목적은 보안결함을 되도록 가장 앞 단계에서 발견하여 비용을 줄이고자 하는 것).
    - ***Minimize the number of security vulnerabilities in design, implementation, and document lifecycle as early as possible(각 단계별로 어떻게 하면 보안결함을 빨리 발견할 것인가에 관한 method를 배움)***Security Engineering is a specialized field of engineering that focuses on the security aspects in the design of systems that need to be able to deal robustly with possible sources of disruption, ranging ***form natural disasters to malicious acts(보안공학은 자연재해도 포함하는 개념)***
      Malicious acts + natural disaster를 포함
    - It is similar to other systems engineering activities in that its primary motivation is to support the delivery of engineering solutions that satisfy predefined functional and

user requirements, but with the added dimension of preventing misuse and malicious behavior. These constraints and restrictions are often asserted as ***security policy(보안공학의 목적은 가능한 빠르게 보안결함 및 약점을 찾아내고 제거하고자 하는 것(방법론)이다, 보안약점은 from natural disaster to malicious acts)***

- Ex) MS SDL은 MS의 SW개발에 있어 S-SDLC를 적용함으로써 보안약점과 비용을 획기적으로 감소시킴

6. **References**

	Public sector	Private Sector	Military Sector
Government	(1) NIS NCSC(국가사이버안전센터): 모든 공공기관 (2) MOPAS(행정안전부): 전자정부쪽 만 담당	KCC(방송통신위원회)	Cyber Command(사이버사령부)
Government Affiliate Agency	NSRI(국가보안기술연구소)	KISA ETRI	ADD(구 무기개발) NSRI(현재)
과거의 Government Affiliate Agency	ETRI	ETRI	ADD

최근에는 분야별로 세분화되는 추세(전력, 금융), Security Control Tower를 만들자!(현재는 NIS에서 역할대행)

Introduction

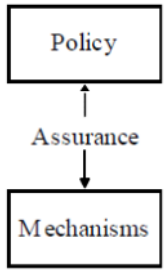
2012년 10월 6일 토요일
오후 2:55

1. **Security Engineering 왜 하나?(Ross Anderson)**
   Security Engineering is about building system to remain ***dependable*** in the face of malice, error and mischance. As a discipline, it focuses on the tools, processes and methods needed to design, implement and test complete systems, and to adapt existing systems as their environment evolves
   Dependable system을 만들어야 하는 것이 보안공학의 목적

2. **Security vs. Dependability**
   Malice is different from error
   a. Security: The Chinese Government won't be able to read this file
   b. Reliability: Bob will be able to read this
   c. Dependability: Reliability(Accidental Failure) + Security (Intentional Failure)
   Proving a negative can be much harder

3. **Information Assurance vs. Security**
   a. Information Assurance의 목적: Dependable systems(자연재해포함), includes reliability and emphasizes strategic risk management over tools and tactics
   b. Information Security의 목적: Secure System, protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction

4. **Relationships**



(1) 요구사항 분석, 요구사항에 대한 만족(SW Engineering)
(2) Security Policy: ***Statement of requirements*** that explicitly definesthe security expectations of the mechanism(s), Security policy를 모호하지 않게 서술하는 것이 관건
(3) ***Assurance:*** 절차나 방법론, Provides ***justification*** that the mechanism meets policy through assurance evidence and approvals based on evidence
(4) Mechanisms: Executable entities that are designed and implemented to meet the requirements of the policy

4. **Type of Assurance**
   Requirement Analysis -> Design -> Implementation -> Operation
   (Policy Assurance) -> Design Assurance -> Implementation Assurance -> Operation Assurance
   a. ***Policy Assurance***: is evidence establishing security requirements in policy is complete, consistent, technically sound
      i. Security Objectives: are high-level security issues
      ii. Security Requirement: are specific, concrete issues
   b. ***Design Assurance:*** is evidence establishing design sufficient to meet requirements of

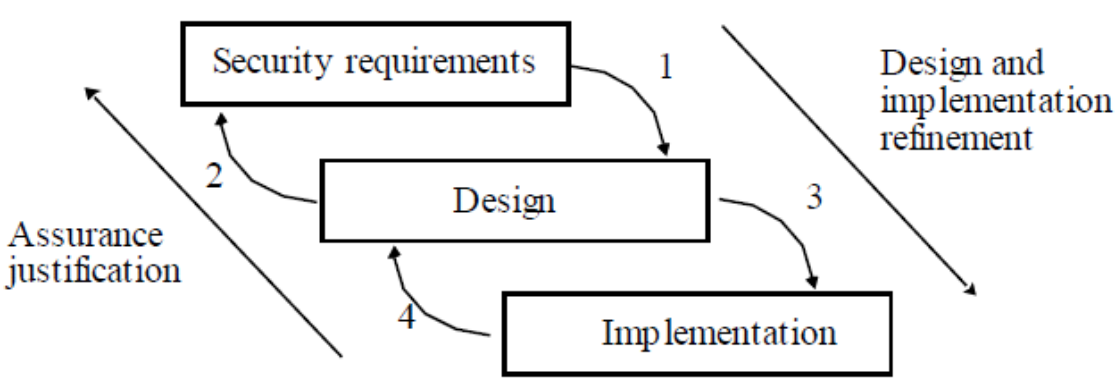
security policy

(예) Access Control: requirement analysis, Windows: Testing, Cryptology: Design, CC/CMVP: Testing(Evaluation)

c. ***Implementation Assurance:*** is evidence establishing implementation consistent with security requirements of security policy

d. ***Operational Assurance:*** is evidence establishing system sustains the security policy requirement s during installation, configuration, and day-to-day operation. ***As called "administrative assurance"***

5. Lifecycle



(2): Policy Assurance, (4): Implementation Assurance

6. Security Policy
   a. State what should be protected
   b. A Security Policy is a statement of what is, and what is not, allowed
   c. And how this should be achieved

7. Security Policy Example - MLS(Multi-Level Security)
   a. MLS system are widely used in government
   b. Basic Idea: a clerk with 'secret' clearance can used documents at 'confidential' and 'secret' but not at 'Top Secret'
   c. 60s/70s:
   d. Initial Attempt:

8. Bel-LaPadula: NRU, NWD, *-Property, *-Policy, Security Policy는 formalize하여 기술해야 함, 최초로 Security Policy를 formulate하여 기술함

9. Formal Mehods
   a. Informal Method: English, other natural languages
   b. Semi-formal Method
   c. Formal Method: Finite State Machines, Petri Nets, Z, ANNA, VDM, CSP, etc

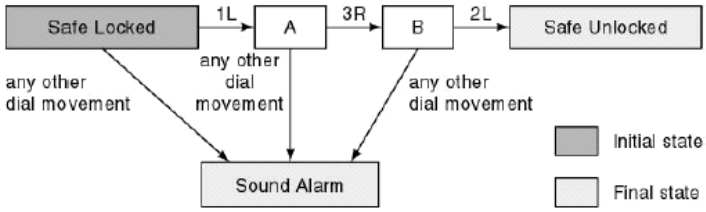
10. FSM Example(금고의 기능)

    [Informal Method]

    (M202, Open University, UK) A safe has a combination lock hat can be in one of

three positions, labeled 1, 2, and 3. The dial can be turned left or right (L or R). Thus there are six possible dial movements, amely 1L, 1R, 2L, 2R, 3L, and 3R. The combination to the safe is 1L, 3R, 2L; any other dial movement will cause the alarm to go off.

[FSM Formal Method]



[State Transition Diagram]

Table of Next States

Dial movement \ Current state	Safe locked	A	B
1L	A	Sound Alarm	Sound Alarm
1R	Sound Alarm	Sound Alarm	Sound Alarm
2L	Sound Alarm	Sound Alarm	Safe Unlocked
2R	Sound Alarm	Sound Alarm	Sound Alarm
3L	Sound Alarm	Sound Alarm	Sound Alarm
3R	Sound Alarm	B	Sound Alarm

[Transition Table] 57

(1) 자연어로 쓰면 부정확하기 때문에 수식으로 정확하게 표현하기 위해 Formal Method를 사용

(2) 보안시스템의 보안요구사항을 분석하고자 할 때 누구나 읽어도 같은 의미로 이해하기 위해 Formal Method를 사용

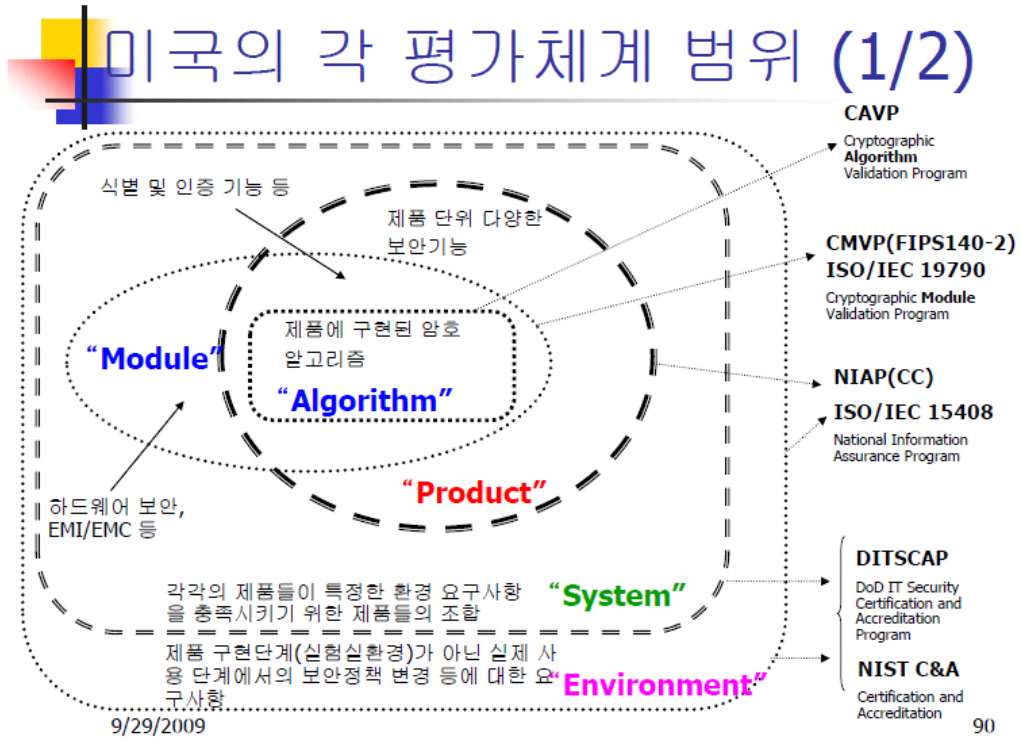
- 자연언어 중 가장 객관적인 것은? 법
- 다른 예제: SEED(암호알고리즘)을 Formal Method인 Petri Nets로 표현

11. Security Engineering Process 1
    a. Requirement Analysis
    b. Formal method로 Security Policy를 완벽하게 기술함
    c. 보안시스템을 Design(Secure Design): 디자인 후 안전한지 분석(그러나 현재에는 암호분야에만 한정되어 적용되고 있다)
    d. 암호를 Design함에 있어서 안정성 분석은 다음 두 가지로 분류
        i. Formal models: 안전성을 자동화하여 증명(정적 분석도구와 비슷, 복잡한 공격탐지 X, 시간이 빠름)
        ii. Computational: 안전성을 수학적으로 수동화 증명(복자반 공격탐지 가능, 시간이 오래 걸림)

12. 현대암호화 고전암호의 구분: 1976년도를 기준으로 나뉨, DH -> RSA, 민간의 암호기술이 군 정부의 암호기술과 대동소이하거나 앞질렀기 때문에 가능, 그 이전에는 군, 정부에서 개발하였으나 76년 이후에는 공개키 암호방식이 학계와 민간부분에서 고안되었다.
    1976: concept of public-key, 1982: Provable Security(안전성증명시작), 1991: Secure Design Method(Practical Approach by random oracle model)

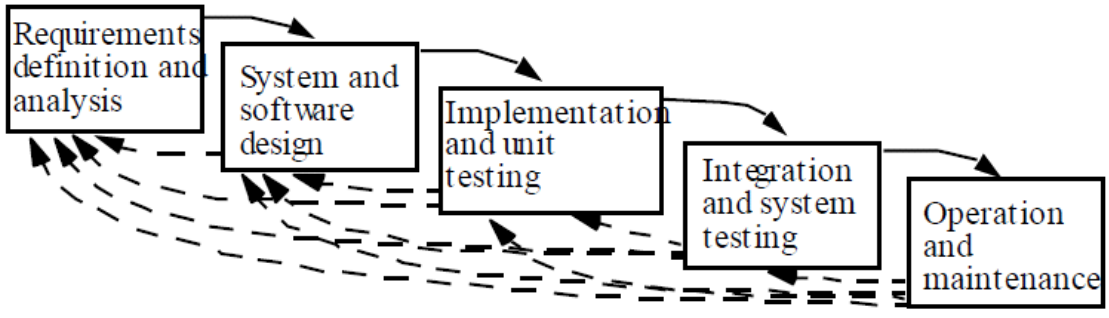
13. Life Cycle
    Conception -> Manufacture -> Deployment -> Fielded Product Life

a. Conception: Idea, Proof of concept, High-Level requirements analysis
b. Manufacture: Develop detailed plans for each group involved, Implement the plans to create entity
c. Deployment: Delivery(Assure..., Distribute...), Installation and Configuration(Ensure product...)

14. Security Engineering Process 2
    a. Requirement Analysis
       보안 요구사항을 분석한 뒤 security policy를 formal method을 이용하여 기술
    b. Design
       Secure Design(=Provably Secure Design), 가장 잘 되어 있는 것: 암호학
    c. Implementation & Testing
    d. 

       미국의 Security Evaluation을 위한 제도들

15. Relationship of Stages



16. Key Points
    a. Assurance critical for determining trustworthiness of systems
    b. Different levels of assurance, from informal evidence to rigorous mathematical

evidence

c. Assurance needed at all staged of system life cycle

Foundations

2012년 10월 6일 토요일
오후 5:23

1. Risk Score Calculation
    a. Risk = Expected Asset Loss * Vulnerabilities * Threats
       추상적인 개념, 자산식별은 매우 어렵다(보안경제학), 취약성(어떻게 계산하나), 위협(어떻게 계산하나)
    b. ALE(Average Loss Expectancy) = Probability of Loss * Total Loss Potential

2. Assets: 자산은 유, 무형의 모든 자산을 포함한다.
   예를 들어, 데이터 가치를 매기려면 데이터를 먼저 분류해야 한다. 데이터를 분류하기 위해서는 (1) 데이터 식별, (2) 분류, (3) 데이터 가치산정의 단계를 갖는다.
   그 밖에, Software, Hardware, Data and Information, Reputation들은 측정이 매우 어렵다.
   자산의 식별은 쉬우나 평가는 어렵다. 즉, Data, Information, Reputation들은 측정이 매우 어렵다.

3. 취약점(Vulnerabilities)
   취약점은 우연적(accidentally = secure), 또는 의도적인(intentionally = reliable) 침해사고로서 자산을 손상시킬 수 있는 시스템의 약점(Weakness)을 말한다.
   Dependable = secure + reliable이며, accidental은 disaster를 포함하는 개념이다.
   예를 들어, 접근 통제의 부적절성, 약한 방화벽 설정, 잘못된 계정 설정 등이 있다.

4. 위협(Threat)
   위협은 취약점을 이용하여 발생되는 것이다. 공격자가 취약점을 이용하여 자산에게 손상을 미칠 수 있는 행동을 의미한다.

(1) Identifying Threats(위협 정의)

- Threat Lists
    - Start with laundry list of possible threats
    - Identify the threats that apply to your application
- STRIDE
    - Categorized list of threat types
    - Identify threats by type / category
- Optionally draw threat trees
    - ...
    - ....

(2) MS's STRIDE Threat Model(위협 분류 및 카테고리화)

MS에서 모든 취약점은 6가지 범주에서 위협 분류가 가능하도록 설계된 방법론

- Spoofing

- Tampering
- Repudiation
- Information Threats
- DoS
- Elevation of Privileges

(3) Threat Trees(공격 트리 및 공격시나리오 수립)

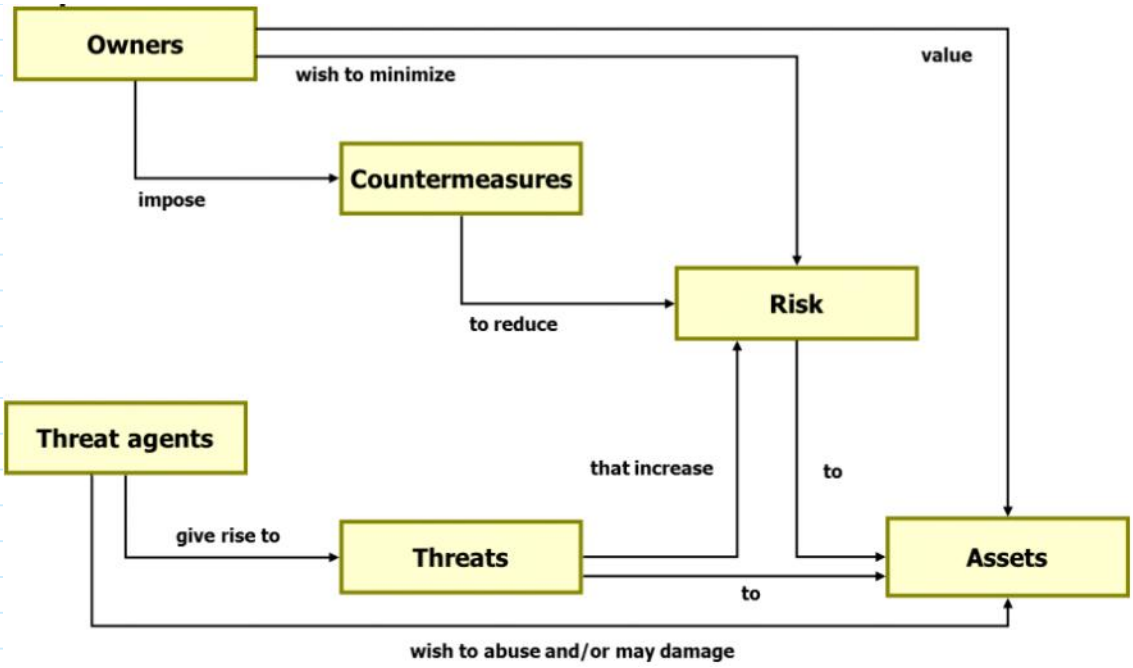
(4) Documenting Threat(위험계산, 문서화 및 계산)

5. 위험(Risk)
   a. 정량적 위험분석
      i. 수학적 이론에 근거한 확률 이론(장점)
      ii. 입력 값에 따라 결과값의 품질이 달라짐(단점)
      iii. 항상 유연한 것은 아님(단점)

   b. 정성적 위험분석(Risk = Probability x Damage Potential, 근데 이건 너무 간단 -> DREAD 모델)
      i. 좀 더 유연하고, 적용하기 쉬움(applicable, 장점)
      ii. 보안전문가의 판단에만 의존하여 값이 달라짐(단점)
      - DREAD Model
        - Damage Potential(잠재적 성공, 데미지, 영향도)
        - Reproducibility(특정상황, 얼마나 자주 일어나는가)
        - Exploitability(공격자가 얼마나 전문성을 갖고 있어야 하는가)
        - Affected users(얼마나 많은 사람들이 영향을 받는가)
        - Discoverability(얼마나 많은 사람들이 알고 있는가)

      DREAD모델은 (1) Greater, (2) Rates(1-15), (3) …
      점수 매기는 모델이다. 순위를 산정할 수 있다.

6. 대응방안 및 위험완화 전략(관계 도표)



사람은 자산을 갖고 있음. 위험이 완화되기를 원함.

Threat agents(공격자)는 위협을 불러일으켜서 위험을 증가시킴, 공격자의 타겟은 자산(Assets)임.

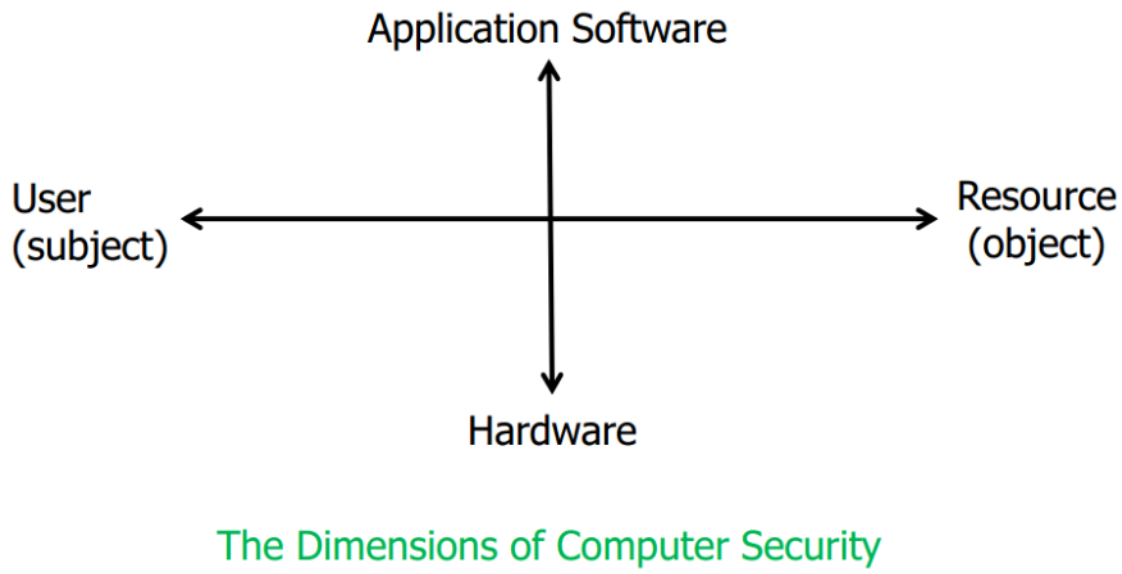
(ch. 4 Threat Modeling 참고)

7. Definitions
    a. Security: is about protecting assets. This is involves prevention, detection, reaction(recover, restore assets)
        Cf. secret(2인 끼리 공유하여 비밀), private(1인, 혼자만 알고 있는 비밀)
    b. Confidentiality: prevent unauthorized disclosure of information(정보에 대한 불법적인 노출을 막음)
        Confidentiality involves privacy, secrecy
        Privacy: protection of private data
        Secrecy: protection of organizational data
        고전 secrecy -> 현대암호 private로 패러다임 변화(공개키 암호시스템: 모든 사람으로부터 나를 지키겠다)
    c. Integrity: prevent unauthorized modification of information
        Cf. 보안시스템이 달성해야 할 목표: C, I, A (IS 관점에서의 3가지), C, I, A, Accountability, Non-repudiation(IA관점에서의 6가지)
    d. Availability: 컴퓨터 시스템이 다음의 내용을 따르고, 의미하고 있음
        i. Service are accessible and useable whenever needed by an authorized entity
        ii. For this we need ***fault-tolerance(HA: 고장감내성)***
        iii. Faults may be accidental or malicious
        iv. DOS attacks are an example of malicious attacks
    e. Accountability: Audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party(감사증적과 책임추적성)
            For this, Audit information must be kept and protected, access control is needed(로그정보에 대한 보호)
    f. Non-repudiation(부인방지): provide unforgeable evidence that a specific evidence that

a specific action occurred
보내고, 안 보낸 척, 배달하고 안 배달한 척, 등기우편, 한국의 샵메일 도입...전자문서 거래법

g. Dependability: = Reliability (Accidental Failures) + Security (Intentional Failures)
h. Survivability(원상태로 복구가능성): deals with the recovery of the system after ***massive failure***

8. Design Principles



The Dimensions of Computer Security

< Four fundamental design policy >

- 디자인 결정 1원칙: ***focus of control(data or operation or user)*** 제품의 설계중심을 ***data(resource, object)에게 맞출 것인가, 사람(user, subject)***에게 맞출 것인가? 가로열

- 디자인 결정 2원칙: ***man-machine scale,*** 보안메커니즘을 어디에 둘 것인가? 세로열

- ***디자인 결정 3원칙: 복잡성(Complexity) vs. 보증(Assurance)*** 보안 초점을 단순함(simplicity)이나 보안성(security)에 둘 수 있나? 높은 보증 수준을 얻기 위해서 보안시스템은 상세하고 가능한 구체적으로 검사되어야 함. 이러한 이유로 복잡성과 보증간에는 명백한 trade-off 관계가 성립한다. 보증 수준을 향상시키기 위해서는 시스템을 단순하게 더욱 더 단순하게 할 필요가 있다? 즉, 복잡할 수록 보증수준은 떨어짐.

- 디자인 결정 4원칙: centralized vs. decentralized 보안통제 과업(security control tasks)는 중앙집권화 되어야 하는가? 또는 개별적인 요소로서 남겨져야만 하는가?

- 디자인 결정 5원칙: how to prevent the attacker from accessing the layer below the protection boundary?
  - The layer below: 하위계층(low)에서 들어오는 공격을 상위계층에서 못 먹는다(최초 보안성 심의시 가정이 수립됨). 이를 보완하기 위해 물리적 또는 운영보안, 관리적 보안통제를 활용하여 기술적 디자인의 한계를 극복할 수 있다.

9. Layers of IT System
   a. Application: users run application programs tailored to meet specific requirements
   b. Services: Application programs make use of services provided by a software packages

like a Database Management System(DBMS) or an Object Reference Broker(ORB).

c. OS: The software packages run on top of the OS which controls access to recourses
d. OS Kernel: The OS may have a kernel that mediates every access to the processor or memory
e. Hardware: (processor & memory) physically stores and manipulates data.

10. Onion model of protection mechanism

11. ***Assurance***
    a. Assurance provides a level of confidence that
        i. The security function is accurately enforced on the security system and
        ii. The security function cannot be compromised or bypassed
        Thus, Assurance is ***the level of trust*** that it really does.

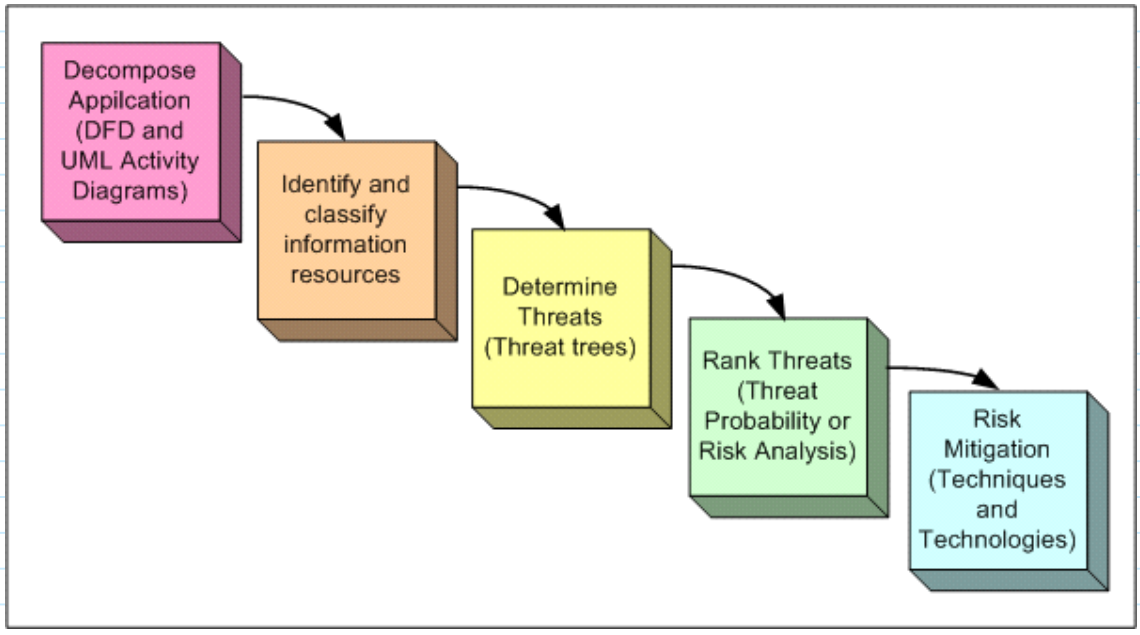
12. Level of Trust, The Level Below
    a. An attacker with access to the 'layer below' is in a position to subvert protection mechanisms further up.
    b. When you reach the stage where you cannot apply computer security mechanisms or do not what to do so, you can still put in place physical or organizational security mechanisms.

Identifying Threats & Threat Risk Modeling

2012년 10월 11일 목요일
오후 11:45

1. Threat Risk Modeling
   It is security analysis to determine the most important security risks to a system. The goal is to reduce the risk to an acceptable level by determining threats to mitigate and the steps to mitigate the identified threats.

2. Process of Threat Risk Modeling



3. How to determine Threats?
   a. Threat Lists
      i. Threat Lists laundry list of possible threats
      ii. Identify the threats that apply to your application
   b. STRIDE
      i. ***Categorized*** list of threat types
      ii. Identify threats by type / category
   c. Optionally draw ***threat trees***
      i. Root nodes represent attacker's goal
      ii. Trees help identify threat conditions

4. MS's STRIDE Threat Model

S **Spoofing**
Can an attacker gain access using a false identity?

T **Tampering**
Can an attacker modify data as it flows through the application?

R **Repudiation**
If an attacker denies an exploit, can you prove him or her wrong?

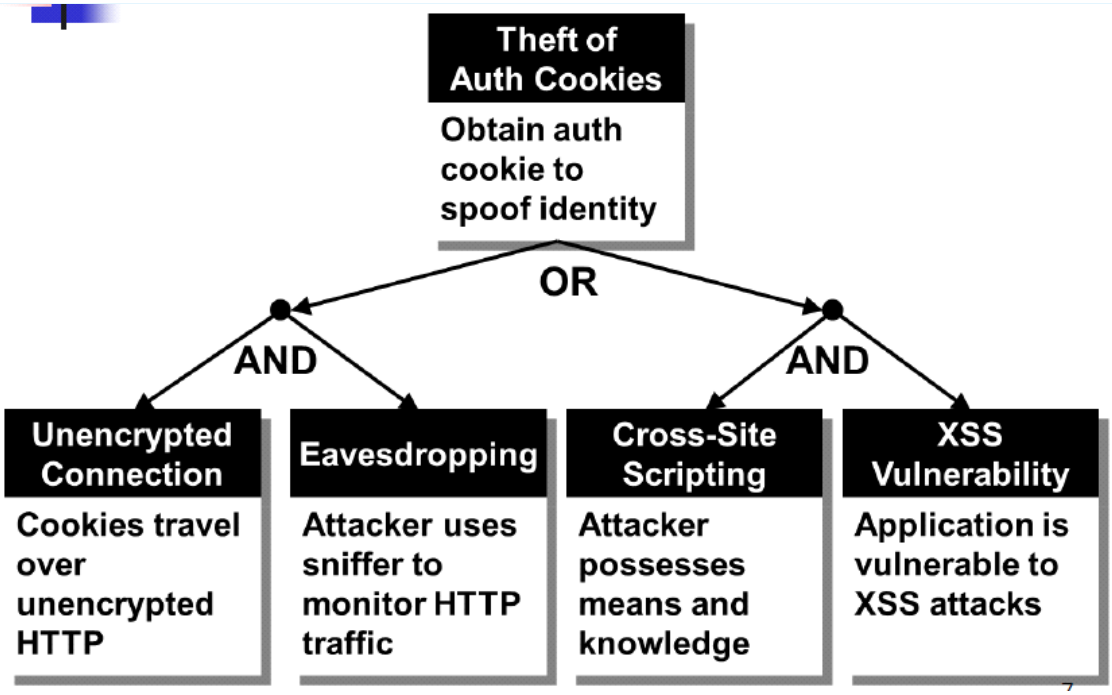
I **Information disclosure**
Can an attacker gain access to private or potentially injurious data?

D **Denial of service**
Can an attacker crash or reduce the availiability of the system?

E **Elevation of privilege**
Can an attacker assume the identity of a privileged user?

위변조, 물리적 부정조작, 부인, 정보 노출, 서비스 거부, 권한 상승

5. Threat Trees



Theft of Auth Cookies: 공격자의 목표

밑의 예하 구조의 Trees: 공격자의 목표를 달성하기 위한 Methods, Scenario

위협 트리를 분류한 후 -> 문서화하는 것이 관건

6. Documenting Threats(문서화)

Document threats using a template

Theft of Auth Cookies by Eavesdropping on Connection	
Threat target	Connections between browsers and Web server
Risk	
Attack techniques	Attacker uses sniffer to monitor traffic
Countermeasures	Use SSL/TLS to encrypt traffic

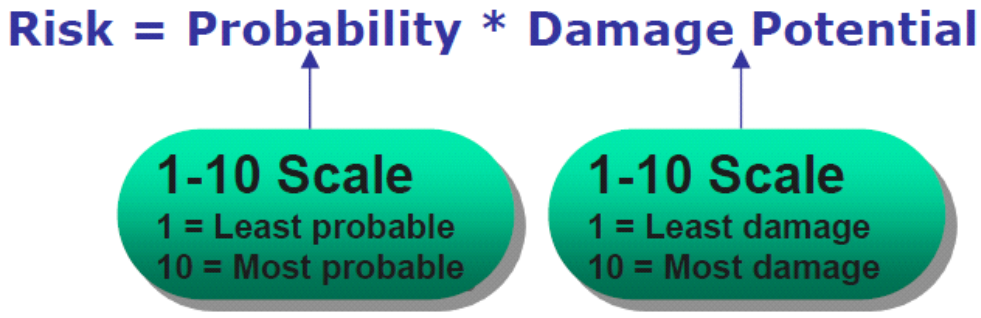
Theft of Auth Cookies via Cross-Site Scripting	
Threat target	Vulnerable application code
Risk	
Attack techniques	Attacker sends e-mail with malicious link to users
Countermeasures	Validate input; HTML-encode output

7. Risk
    a. 정량적 위험분석
        i. 수학적 이론에 근거한 확률 이론(장점)
        ii. 입력 값에 따라 결과값의 품질이 달라짐(단점)
        iii. 항상 유연한 것은 아님(단점)

    b. 정성적 위험분석(Risk = Probability x Damage Potential, 근데 이건 너무 간단 -> DREAD 모델)
        i. 좀 더 유연하고, 적용하기 쉬움(applicable, 장점)
        ii. 보안전문가의 판단에만 의존하여 값이 달라짐(단점)
        - DREAD Model
            - Damage Potential(잠재적 성공, 데미지, 영향도)
            - Reproducibility(특정상황, 얼마나 자주 일어나는가)
            - Exploitability(공격자가 얼마나 전문성을 갖고 있어야 하는가)
            - Affected users(얼마나 많은 사람들이 영향을 받는가)
            - Discoverability(얼마나 많은 사람들이 알고 있는가)

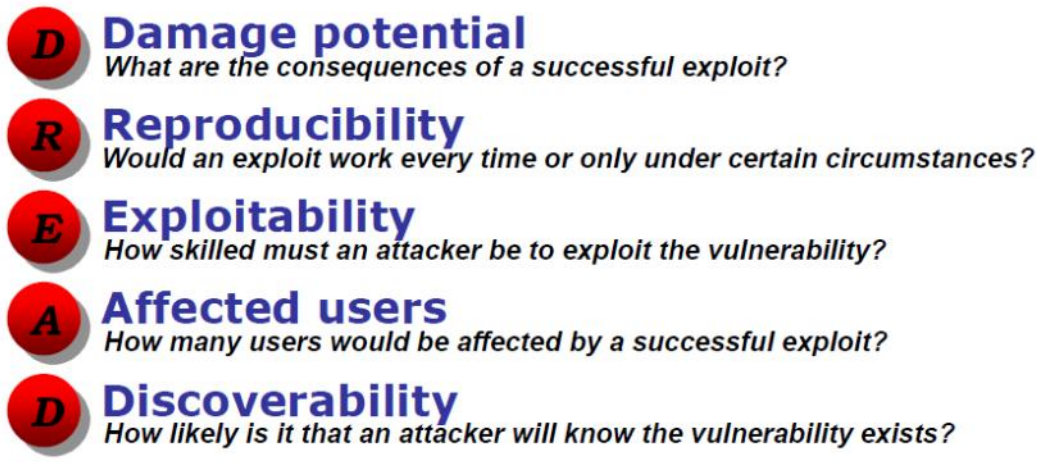
        DREAD모델은 (1) Greater, (2) Rates(1-15), (3) ...
        점수 매기는 모델이다. 순위를 산정할 수 있다.

8. Simple Risk Analysis



9. Simple Risk Analysis 에서 보다 정밀하게 된 MS's DREAD Analysis Model
    a. Greater granularization of threat potential
    b. Rates each threat on scale of 1-15

c. Developed and widely used by Microsoft



손실은 얼마나?, 특정조건에서 발생하는지, 아닌지(얼마만큼의 조건을 만족시켜야 하나?), 해커의 전문성(천재만 가능한지, 바보도 가능한지), 얼마나 많은 사람이 영향을 받는지, 이 취약점을 얼마나 찾을 수 있는가?

	High (3)	Medium (2)	Low (1)
Damage potential	Attacker can retrieve extremely sensitive data and corrupt or destroy data	Attacker can retrieve sensitive data but do little else	Attacker can only retrieve data that has little or no potential for harm
Reproduc-ability	Works every time; does not require a timing window	Timing-dependent; works only within a time window	Rarely works
Exploitabilty	Bart Simpson could do it	Attacker must be somewhat knowledgeable and skilled	Attacker must be VERY knowledgeable and skilled
Affected users	Most or all users	Some users	Few if any users
Discoverabilty	Attacker can easily discover the vulnerability	Attacker might discover the vulnerability	Attacker will have to dig to discover the vulnerability

(1) 위험 분석은 대응방안을 추천하는 데 제시한다

(2) 위험분석은 시간과 비용 문제로 인해 항상 정당화 될 수 있는 수단은 아니지만, 위험 완화를 위한 차원에서 도움이 될 수 있다.

(3) 베이스라인(기본수준) 보호: security requirements for typical cases with recommended countermeasures

Access Control (1)

2012년 10월 11일 목요일
오후 10:40

1. Security Policy

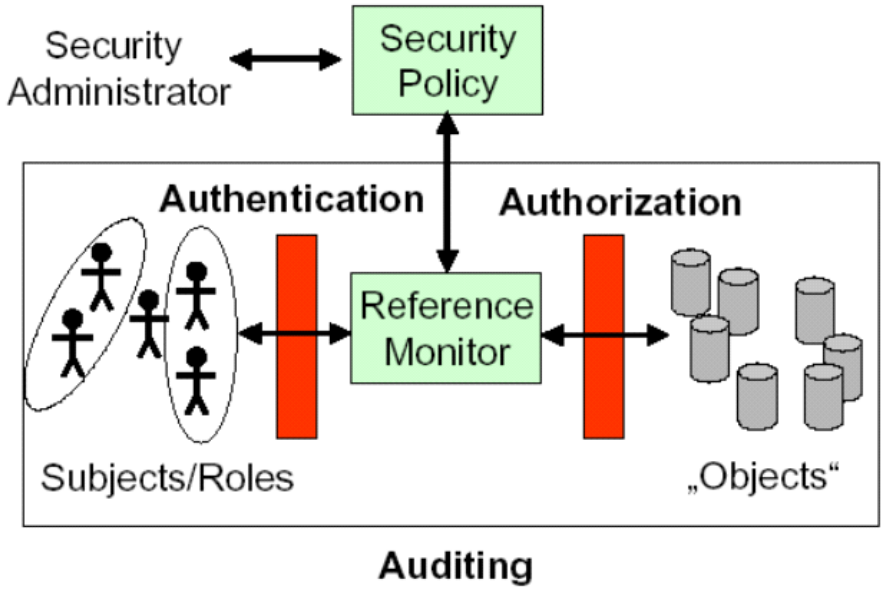
   Security Policy는 무엇이 보호되어야 하고, 어떻게 해야만 보안통제의 목적을 달성할 수 있을까에 대한 요건을 정의한 것. C, I, A의 관점에서 정책은 다르게 다뤄질 수 있음

- MLS(Multi-Level Security) systems are widely used in government
- Clearance, 비밀취급 인가등급에 따른 접근제어 방식

2. Formulating the Security Policy
- BLP: 1973, Bell-LaPadula, NRU, UWD, *-Property
- Minimize the TCB(Trusted Computing Base) in a reference monitor

3. 접근의 3단계
- Identification, Authentication, Authorization

4. Reference Monitor
- Reference Monitor which enforces the authorized access relationships between subjects and objects of a system
- 참조모니터는 접근제어를 위한 추상적 개념으로서, 객체로의 접근이 요청될 때 작동,
- 참조모니터의 3가지 속성: Completeness(완전성, 우회불가능), Isolation(격리성), 검증가능성(분석하고 테스트할 정도로 작아야 함)
- Security Kernel은 참조모니터가 구현된 형태의 예
- TCB: Security Kernel + Other Protection Mechanisms
-



5. Identification vs. Authentication
- ***Authentication:*** means verifying the user is actually who he says he is(or who she says she is), ***본인이 누구라는 것을 시스템에 밝히는 것, 반드시 유일한 것을 사용해야 함 = verification, 1:1 Matching*** , Proving
- ***Identification:*** 임의의 정보에 접속할 수 있는 주체의 능력이나 주체의 ***자격을 검증하는 단계.*** 시스템이 본인임을 주장하는 사용자가 본인이 맞다는 것을 인정해 주는 것, 1:N Matching,

establishing

6. Authentication Method
- Something you know, Something you have, Something you are
- 2 factor authentication

7. 인증 시 로그인 실패와 같은 에러메시지는 최대한 공개되면 안된다. 공격자는 에러메시지를 통해 내부시스템에 관한 정보를 유추하여 보다 효율적인 공격을 수행할 수 있게 되기 때문이다.

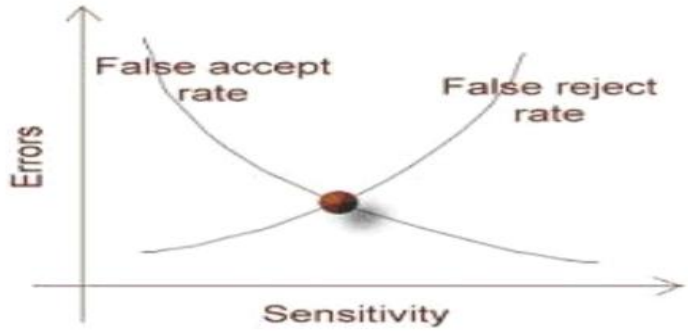
8. Password Cracking & Authentication
- ***Brute-forcing attack: 무차별 공격***
- ***Dictionary attack:*** 사전에 등재된 단어를 우선적으로 대입하여 무차별 공격
- 일반적인 대응책: ***임계치설정, 지연시간(delay time)설정***

- 또 다른 공격2: Rainbow attack, 해시테이블을 미리 생성하여 무차별공격
- 대응책: 공개된 난수 값인 salt값을 계정별로 조합하여 해시테이블 생성(shadow file)및 유지
    - ***Salt의 효과: salt를 사용함으로써 shadow파일이 외부에 노출되었다 할지라도 공격자의 offline무차별/사전공격을 매우 어렵게 하기 때문에 안전성을 증가시킬 수 있다.*** 즉, Large salt values를 사용함으로써 다수의 사용자의 개인정보를 보호할 수 있다. Salt는 공격자가 내부시스템에 대한 모든 정보(shadow파일 등)를 얻었다 할지라도 PW와 같은 정보를 알아내는 시간을 지연시킨다. 대표적인 예는 ***Key Stretching Algorithm이 적용된 PBKDF2***와 같은 알고리즘을 사용하면 된다.
    - 그러나 위의 방법이 완벽하게 해결되는 것은 아니다. Key Stretching Algorithms이 적용된 인증시스템에서 모든 사용자에게 생성되는 salt의 값은 다르기 때문에 다수에게는 안전하나 특정사용자에게 target화된 공격은 경우의 수가 한정적일 수 밖에 없으므로 안전하다고 볼 수 만은 없다(***Chosen-victim attack***이라 한다)

- 또 다른 공격3: Replay Attack, 재생공격은 암호화된 데이터들도 재생공격이 가능하다.
- 대응책: 인증과정에서 난수 값이 ***Nonce 나 timestamps***를 첨가하도록 해야 한다.

- 인증수단 강화(대응책)
    - ***Grid OTP(보안카드):*** 경우의 수가 한정되어 보안강도가 낮다
    - ***RSA SecurID OTP***: PIN + Token Code를 혼합하여 PW생성, 인증 / Authenticator와 ACE/Server는 같은 마스터키와 seed를 공유하고 있음. Timebased > Counterbased 기반 OTP 2가지가 있다. Timebased가 더 보안성이 좋다.
    - ***RSA SecurID OTP With transaction signing:*** 카드형 키패드가 있는 OTP를 사용, 계좌번호와 금액을 카드에 입력하여 생성된 값(OTP기반 서명)을 컴퓨터에 입력하여 거래
    - ***Passcard(교통카드):*** 스마트카드 별로 Key가 존재함, 우리나라에서 업체별로 key가 달라서 교통카드가 호환이 되지 않았었음, Challenge-Response방식을 채택

9. 생체인증
    - 생체인증은 ***신체적 특성***에 기반한 인증방법과 ***행동적 특성***에 기반한 인증방법이 존재한다.
    - 특성에 기반한 인증방법은 홍채, 망막, 정맥, 지문과 같은 것이며, 행동적 특성에 기반한 인증방법은 걸음걸이가 있다.



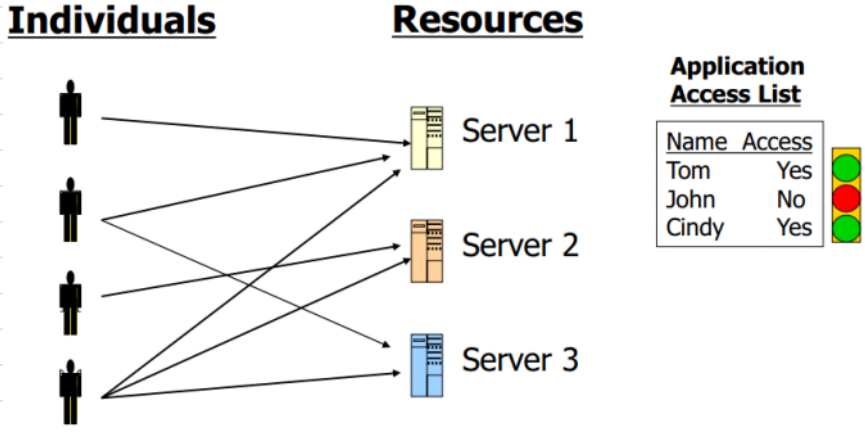
- Type 1 Error: False Rejection Rate(FRR), 잘못 거절될 확률(나는 진짜인데 에러난 것) -> 일반적인 에러
- Type 2 Error: False Acceptance Rate(FAR), 잘못 수락될 확률(가짜인데 승인된 것) -> 도둑놈
- EER(Equal Error Rate)= CER, Trade off가 일어나는 지점. 교차점.
  - Most ***helpful*** when comparing the different biometic system
  - Most ***important*** state for determining system's accuracy
  - Rating stated as a percentage and represents the point at which FAR=FRR
  - EER은 생체인증의 정확도를 설정하는 지점이라 할 수 있다.

10. Authorization

- Access Control Security Model, ***Security Model***(=Formal Specification of Security Policy): ***Security Policy를 정형화한 방법으로 명세적으로 기술한 것***
  예) BLP, MAC, RBAC, DAC 등

11. DAC Model(임의적 접근제어)

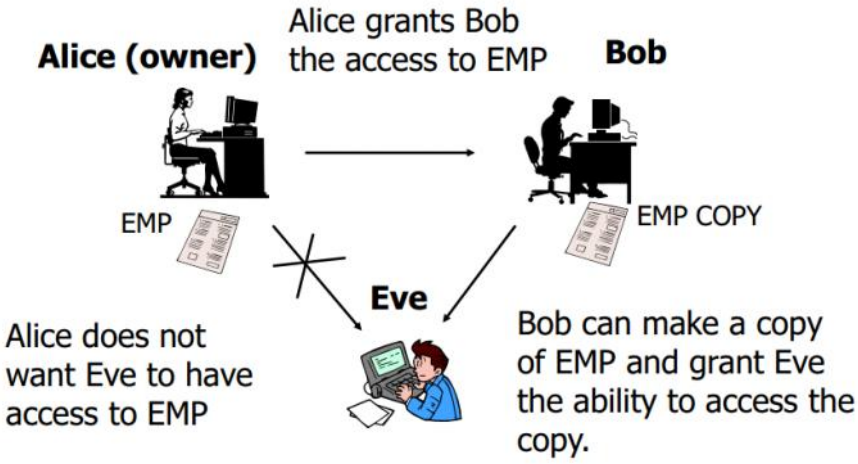
- 임의적 접근제어는 ***데이터 소유주***가 비밀취급등급(clearance)를 결정
- 데이터 소유주는 ***Need to know(알 필요성의 원칙)***에 따라 등급산정
- 개인이나 그룹별로 접근권한을 제어할 수 있음
- ***ACL(접근제어리스트)***에 기반하여 접근제어 설정



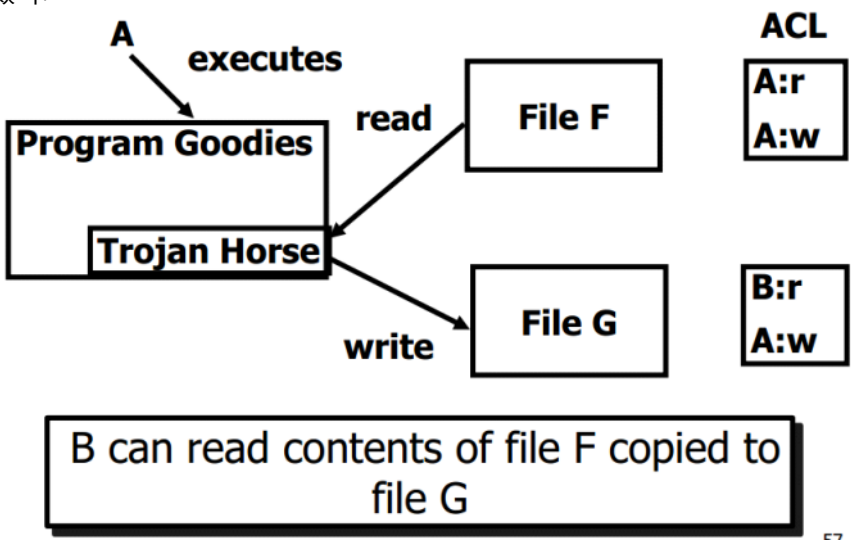
예) Unix의 파일시스템의 접근제어 설정

Operations	Owner	Group	Size	Updated	Name
drwxr-xr-x	jason	Research	512	Jul 3 15:51	Research
-rw-------	jason	research	127092	Aug 28 15:01	Trash
-rwxr-xr-x	jason	research	7632	May 20 18:16	a.out
-rw-r-----	jason	research	0	Nov 25 1998	allfiles.txt

- DAC의 위험성



데이터 소유주인 Alice는 bob에게만 접근권한을 줬는데 bob이 eve에게 임의로 데이터를 제공한 경우, alice는 eve에 대한 접근을 제한하고 싶었는데 eve는 bob을 통해 데이터에 접근할 수 있다.



또한 File F는 Alice에게만 읽기, 쓰기 권한이 있었지만, A가 무의식적으로 실행한 악성코드에 의해 File F의 일부 정보가 File G에 기록되는 경우 File F에 대한 정보가 G로 유출될 수 있다(이때 Alice는 File G에 대해 쓰기 권한이 있고, Bob은 File G에 대한 읽기 권한만 갖고 있었다고 가정한다)

12. MAC(강제적 접근제어): BLP = MLS
    - ***NRU, NWD, *-Property, 엄격함, 계층적, compartment(구획), Partially ordering방법으로 Lattice구조를 갖고 있어야 함,***
    - 1973년도에 군사용을 목적으로 Formal Method로 기술된 최초의 Security Policy임
    - MAC은 비밀취급등급(clearance)과 데이터의 민감(성)등급(sensitivity level)에 의해 결정되는 접근제어 방식

- ***MAC은 (1) Partially ordering 방법으로, (2) Lattice구조를 만족하고 있는 formal method로 기술한 접근제어방식(Security Policy)이다.***
- Partially Ordering Set

  - A Set S with relation ≤ (written (S, ≤) is called a partially ordered set if ≤ is
    - Anti-symmetric
      - If a ≤ b and b ≤ a then a = b
    - Reflexive
      - For all a in S, a ≤ a
    - Transitive
      - For all a, b, c. a ≤ b and b ≤ c implies a ≤ c

  Partially ordering set은 두 개의 변수에 대한 비교연산이다. 위 표현식은 기본적으로 암기하자.
- Lattice구조란 다음의 두 가지 조건을 만족해야 함을 정의하고 있다. 마찬가지로 암기하자.

  - Partially ordered set (S, ≤) and two operations :
    - greatest lower bound (glb X)
      - Greatest element less than all elements of set X
    - least upper bound (lub X)
      - Least element greater than all elements of set X

  - **Lattice :** A finite set together with a partial ordering on its elements such that for every pair of elements there is a least upper bound and a greatest lower bound.

  Glb X, lub X가 무엇인가?

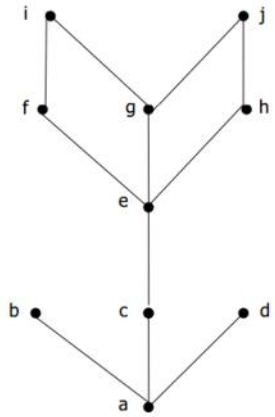
  "Lattice 구조를 갖는다" -> 유한체의 어떤 set(변수)이든 최대치와 최소치를 구할 수 있어야 한다는 뜻이다.
- 이건 또 뭔가...

  - We wish to have unique answers to the following two questions :
    - Given two objects at different security levels, what is the minimal security level a subject must have to be allowed to read both objects?
    - Given two subjects at different security levels, what is the maximal security level an object can have so that it can still be read by both subjects?
  - The mathematical structure that allows us to answer these two questions exist. It is called a lattice.

  위의 두 가지 질문을 만족해야 하나?

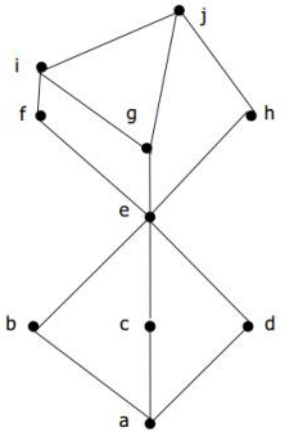
13. Partially Ordering & Lattice Example

- (e.g.) Given two objects



- (e.g.) Given two objects at different security levels b, c, what is the minimal security level a subject must have to be allowed to read both objects?

이것은 Lattice구조가 아니다. b와 c가 서로 접근할 수 있는 최대치와 최소치가 존재하지 않기 때문이다.(어떤 변수든 간에 접근되지 못하는 곳이 있으면 안된다)



- (e.g.) Given two objects at different security levels b, c, what is the minimal security level a subject must have to be allowed to read both objects?

즉, 위와 같은 그림으로 변환될 수 모든 set(변수)들은 접근제어가 가능하게 된다. 그러므로 이와 같은 lattice구조가 될 때 접근제어 정책이 타당하게 수립될 수 있다.

14. Access Control Example

Access control structures

- How are access control rights defined? Many different schemes, but ultimately can be modelled thus:
  - A set S of subjects, a set O of objects
  - A set A of operations (modelled by access rights), we'll consider $A = \{\text{exec}, \text{read}, \text{append}, \text{write}\}$.
  - An **access control matrix**

$$M = (M_{so})_{s \in S, o \in O}$$

where each entry $M_{so} \subseteq A$ defines rights for s to access o.

- Example matrix for $S = \{\text{Alice}, \text{Bob}\}$ and three objects:

	bob.doc	edit.exe	fun.com
Alice	{}	{exec}	{exec, read}
Bob	{read, write}	{exec}	{exec, read, write}

Security levels

- Ideas for **multi-level security** (MLS) systems originated in the military. A **security level** is another kind of security attribute used to label subjects and objects, to describe security policies.
- Security levels (like protection rings) come with an ordering. Typical example:

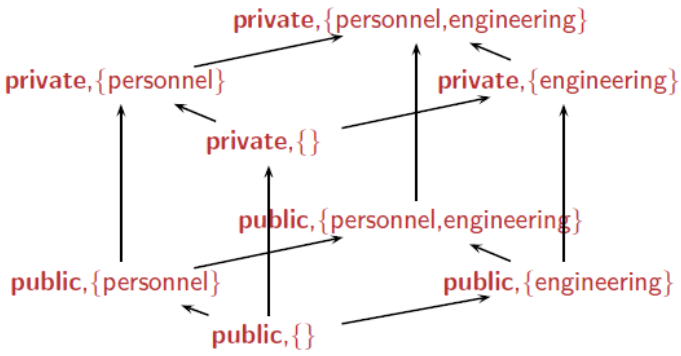
$$\text{unclassified} \leq \text{confidential} \leq \text{secret} \leq \text{topsecret}.$$

- Ordering can express policies like "no write-down" which means that a high-level subject cannot write down to a low-level object. (A user with confidential clearance cannot write an unclassified file: it might contain confidential information read earlier.)
- In practice, more security levels need to be more flexible than linearly-ordered document classifications above. We may want *categorizations* as well, for example, describing departments or divisions in an organization. Then individual levels may not be comparable.

Security lattices

- A *lattice* is a set L equipped with a partial ordering $\leq$ such every two elements $a, b \in L$ has a *least upper bound* $a \vee b$ and a *greatest lower bound* $a \wedge b$. A finite lattice must have top and bottom elements.
- In security, if $a \leq b$, we say that b **dominates** a. The bottom level dominated by all others is system low; the top level which dominates all others is system high.
- Lattices were proposed to model MLS policies because they allow an ordering of security levels such that:
  - Given two objects at different levels a and b, there is a minimal security level $a \vee b$ needed to access both a and b;
  - Given two subjects at different levels a and b, there is a maximal security level $a \wedge b$ for an object which must be readable by both.

A standard construction is to take a set of *classifications* H, with a linear ordering $\leq_H$, together with a set C of categories. Define a *compartment* as a set of categories, and then a security level as a pair (h, c) where $h \in H$ and $c \subseteq C$. Then the ordering $(h_1, c_1) \leq (h_2, c_2) \iff h_1 \leq h_2, c_1 \subseteq c_2$ defines a lattice.



15. Exercise 4.3. Discuss: what are the differences between groups and roles, if there are any differences at all?

sol) Roles and groups serve distinct purposes. You can't assign permissions to a role or capabilities to a group. Here are some additional distinctions:
The identity hierarchy is relevant for groups, but not for roles. If you are a member of a role, you have all of that role's capabilities, regardless of whether you are a direct member of that role and what your other memberships are.
You can deny a permission to a group, but you can't deny a capability to a role. Each role either provides or doesn't provide each capability. No role takes capabilities away from its members.
A group's permissions are not displayed as part of a group definition, but a role's capabilities are displayed as part of a role definition.
A group can be a member of another group, but a role cannot be a member of another role. Instead, one role can contribute its capabilities to another role.

Access Control (2)

2012년 12월 1일 토요일

오후 2:14

1. Multilevel Security (MLS) in formal

2. Bell-LaPadula Model

- 1973년에 기술된 최초의 MLS를 수학적으로 정형화한 형태로 기술한 보안정책
- BLP는 information flow관점에서 기술되었음

• Information flow: (1) 소프트웨어 보안: 변수들 사이에서 정보가 어떻게 흘러가고 있는가?

(2) 접근제어: BLP(자연어를 formal하게 서술)

- BLP의 기본 가정: 설정된 접근권한은 변경되지 않는다.

2. BLP의 문제제기

- BLP는 MLS를 적용한 접근제어 정책으로서 MAC의 방식을 적용하고 있음. 기본적으로 Confidentiality를 달성하기 위한 보안정책 모델임
- BLP모델의 한계: (1) 오로지 기밀성에 관련 문제만을 다루고 있음

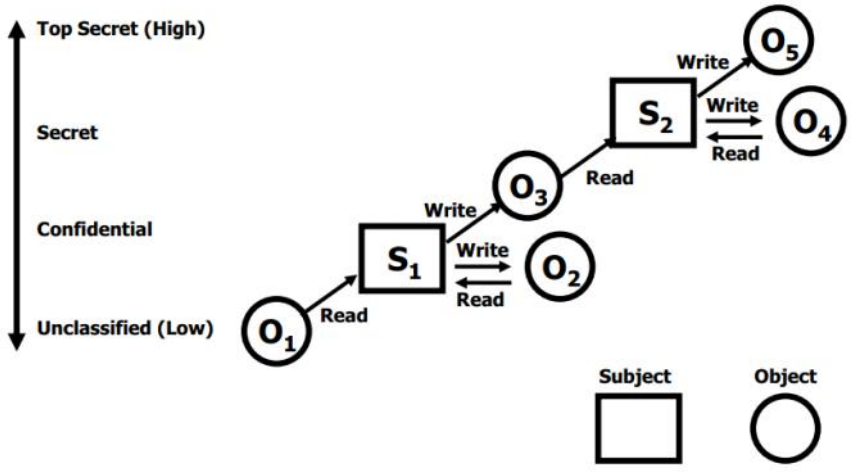
(2) 접근제어 권한이 중간에 변경됐을 때의 문제를 다루고 있지 않음

(3) 은닉채널(Covert Channel)에 의한 정보 흐름(Information flow)의 문제를 다루고 있지 않음

그렇다고 BLP모델이 안전하다, 안전하지 않다 단정지을 수 없다. 왜냐하면 BLP는 Provable Security를 달성하고 있기 때문이다. Secure하다, Secure하지 않다의 문제가 아니다.

3. BLP Model in a nutshell

정보의 흐름 관점에서 BLP모델이 타당한 보안정책이 되기 위해서는 정보가 위로만 흘러야 하지, 아래로 흘러서는 안된다.



4. BLP Model in formal(어떠한 관점에서 BLP가 구현되었나?)

(1) BLP는 reference monitor를 구현할 요건을 갖출 것

(2) 원하지 않은 정보의 흐름을 막을 것

핵심적인 기능만을 추려서 구현되어야 한다. 커널에 넣어야 하므로, 할 수 있는 것과 할 수 없는 것을 명확하게 구분하기 위해서이다.

- Element의 정의(p. 71)

Elements of Access Control

- a set of subjects S
- a set of objects O
- set of access operations A = {execute, read, append, write}
- A set of security levels L, with a partial ordering <=

- State Set의 정의(p. 72)

The State Set

- A state : (b, M, f), includes
- Access operations currently in use b
  - List of tuples (s, o, a), $s \in S$, $o \in O$, $a \in A$.
- Access permission matrix
  - $M = (M_{s,o})_{s \in S, o \in O}$, where $M_{s,o} \subset A$
- Clearance and classification $f = (f_S, f_C, f_O)$
  - $f_S : S \rightarrow L$ maximal security level of a subject
  - $f_C : S \rightarrow L$ current security level of a subject ($f_C \leq f_S$)
  - $f_O : O \rightarrow L$ *classification* of an object

- Simple Security Property(p. 73)

Simple Security Property (SS-Property)

- A state (b, M, f) satisfies the SS-property if
  - $\forall (s, o, a) \in b$, such that $a \in \{\texttt{read}, \texttt{write}\}$
  - $f_O(o) \leq f_S(s)$
- I.e. a subject can only observe objects of lower classification

- *-Property, Start Property(p. 74)

*−Property (Star-Property)

- A state (b, M, f) satisfies the *-property if
  - $\forall (s, o, a) \in b$, such that $a \in \{\texttt{append}, \texttt{write}\}$
  - $f_C(s) \leq f_O(o)$
- and
  - if $\exists (s, o, a) \in b$ where $a \in \{\texttt{append}, \texttt{write}\}$,
  - then $\forall o'$, $a' \in \{\texttt{read}, \texttt{write}\}$, such that $(s, o', a') \in b$
  - $f_O(o') \leq f_O(o)$
- I.e. a subject can only alter objects of higher classification,
- and cannot read a high-level object while writing to a low-level object.

- Discretionary Security Property (DS-Property)

- **Discretionary Security Property (DS-Property)**
  - Mandatory access control properties (SS and *– properties) do not check whether a particular access is specifically permitted.
  - Discretionary Security Property (DS-Property)
    - Defines the capability of a subject to operate on an object.

  In BLP, access must be permitted by the access control matrix $M_{s,o}$.

5. McLean의 BLP에 대한 의구심(Criticism)

- Proposed System **Z = BLP + (request for downgrade)**
- Downgrade하면 문제가 발생하는 것은 아닌가?, not secure? Vs. 그것은 전제조건이 아니다.
- 허용이 되는 것과 허용되지 않는 것을 명확하게 기술하라
- 맥린과 평정속성(Tranquility)
  - 평정속성: 처음 셋팅 그대로 간다는 의미로서 권한이 변경되지 않음을 전제로 하고 있음
  - McLean's seenario is relly **out of scope** for BLP -> 답변: 맥린의 질문은 전제조건을 벗어남
  - BLP considered **tranquil** system: where permissions do not change

6. Convert Channel

(1) 은닉채널이란 당초의 설계자가 의도하지 않은 통신채널을 말한다

(2) *-Property로 막을 수 만은 없음(없애는 것이 불가하다)

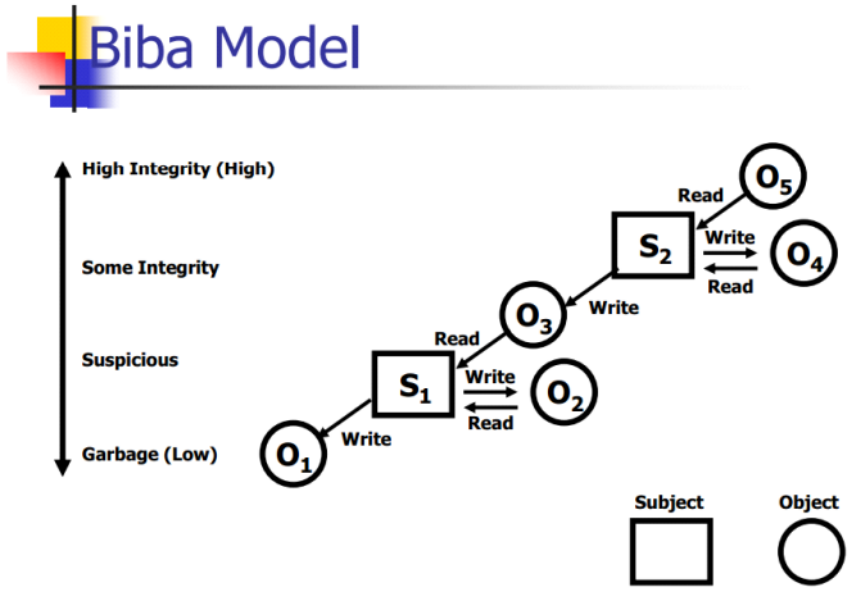
Covert Channel cannot be blocked by *-property

- 이러한 관점에서 은닉채널로 인해 유출되는 정보의 양(Bandwidth)을 줄이는 과정으로서 연구가 진행됨

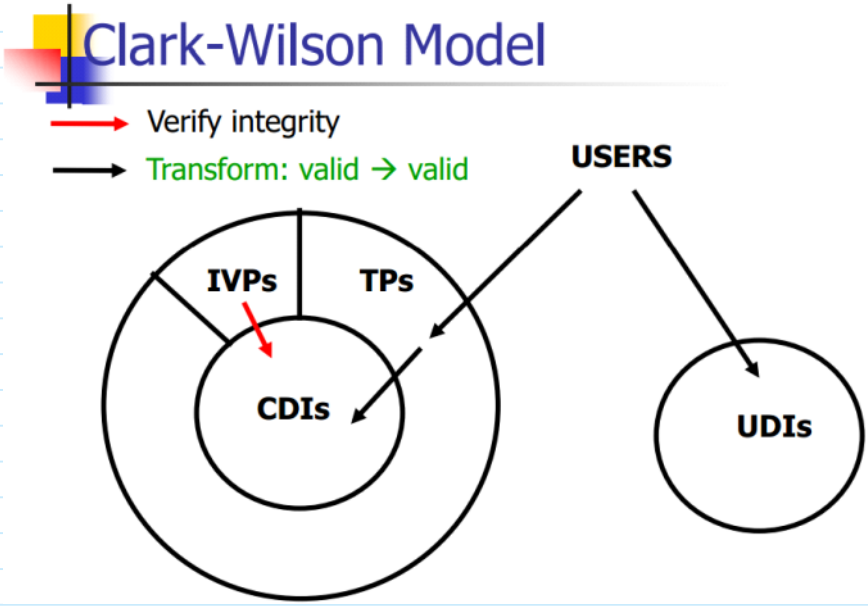
It is generally very difficult, if not impossible, to block all cover channels

7. Biba Model

- Counterpart to BLP, State Machine model similar to BLP
- NRD, NWU(BLP와 반대, BLP: NRU, NWD)
- Biba Model은 무결성 등급을 정의함
- Integrity 관점에서 정보의 흐름은 밑으로만 접근
- 암호의 Integrity와 접근제어에서의 Integrity는 다르다. 암호: 메시지 인증, 접근제어: 메시지가 참일 가능성



8. Clark-Wilson Model
   - (1) Access Control을 상용에 적용하기 시작한 것
   - (2) Well-formed transaction을 통한 무결성 강화(Program 자체를 컨트롤): Data manipulated only by a specific set of programs. Users have access to programs rather than data itself



   -
     - Constrained Data Items (CDI): Data subject to Integrity Control(무결성이 반드시 보장되어야 하는 data group)
     - Unconstrained Data Items (UDI): data not subject to IC(무결성이 별로 안 중요한 Data group)
     - Integrity Verification Procedures(IVP): 무결성 모니터링 프로그램 Test CDI's conformance to integrity constraints at the time IVPs are run(checking that accounts balance)
     - Transformation Procedures (TP): well-formed transaction program, TP라는 적법한 프로그램을 통해서만 접근, 사전에 정의된 프로그램

9. HRU Model(Harrison-Ruzo-Ullman Model)
   - BLP has no policies for **changing** access rights or for the **creation** and **deletion** of subjects and objects

- BLP has no policies for changing access rights or for the creation and deletion of subjects and objects.
- The Harrison-Ruzzo-Ullman (HRU) model defines authorisation systems that address these issues.
- The components of the HRU model:
  - set of subjects S
  - set of objects O
  - set of access rights R
  - access matrix $M = (M_{s,o})_{s \in S, o \in O}$: entry $M_{s,o}$ is a subset of R giving the rights subject s has on object o

Harrison-Ruzo-Ullman Model

- Six primitive operations for manipulating subjects, objects, and the access matrix :
  - enter r into $M_{s,o}$
  - delete r from $M_{s,o}$
  - create subject s
  - delete subject s
  - create object o
  - delete object o

- Security vs. Complexity in HRU Model
  - The access matrix describes the state of the system; commands change the access **matrix;**
  - Checks need to be done in order to avoid undesirable access rights to be granted
  - HRU model has some definitions and thermos about the decidability of **the safety** of the system
    - Saying that HRU model does not help to verify safety in its full generally, but verification is possible **with some restrictions**

Access Control에서의 Matrix란?

접근권한이 어떻든 간에(모든 경우의수에 대하여) reference monitor가 safety한 상태가 유지되는가?

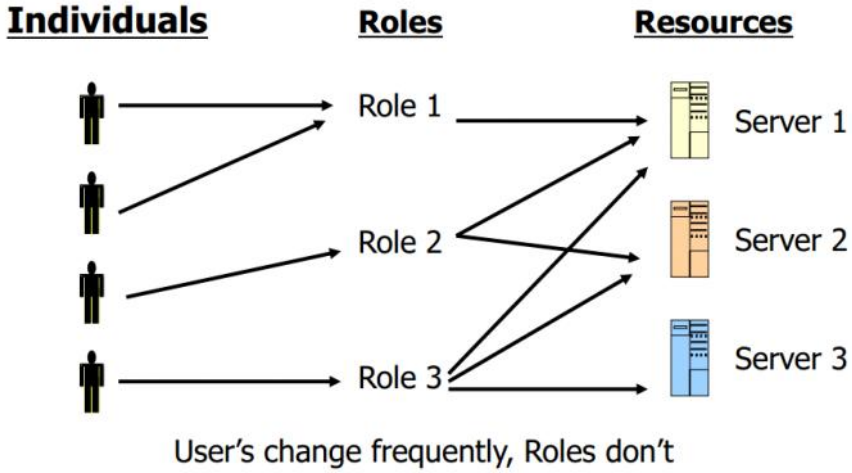
Access Control Matrix의 의 접근권한이 변할 때 보안문제가 없는가 reference monitor가 점검 -> **Safety Property!**

10. Chinese Wall Model (= Brewer and Nash Model)
- 개념: The motivation for this work was to avoid that sensitive information concerning a company be disclosed to competitor companies through the work of financial consultants(경쟁관계 업체간 이익의 충돌을 방지, 주로 금융회사에서 나타남)
- Companies in competition - **Conflict of Interest class**(이익이 충돌하는 대표집단)
- CW Model 에서의 은닉채널은? 집성과 추론으로 인한 컨설팅 회사의 규모파악 등

11. RBAC Model (Role based Access Control Model)

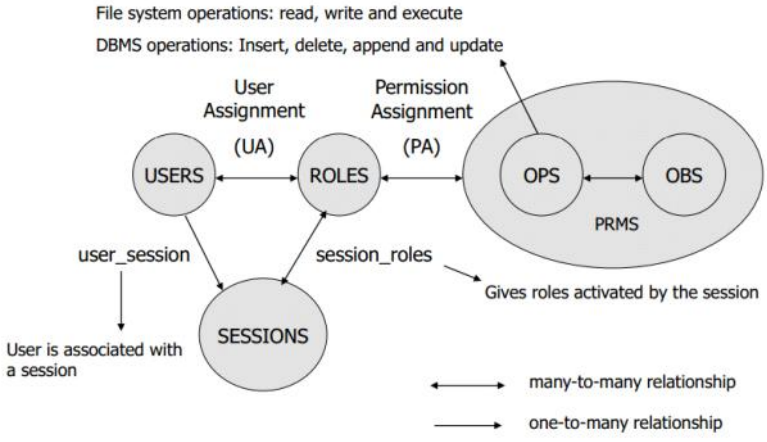
- Originally developed by David Ferraiolo and Rick Kuhn (1992), and by Ravi Sandhu and colleagues (1996).
- A role-based access control (RBAC) model, also called 'nondiscretionary access control', allows access to resources to be based on the role the user holds within the company.
  - It is referred to as nondiscretionary because assigning a user to a role is unavoidably imposed.
- An RBAC is the best system for a company that has high employee turnover.

- **Nondiscretionary Access Control Model** allows access to resource to be **based on the role** the user holds within the company



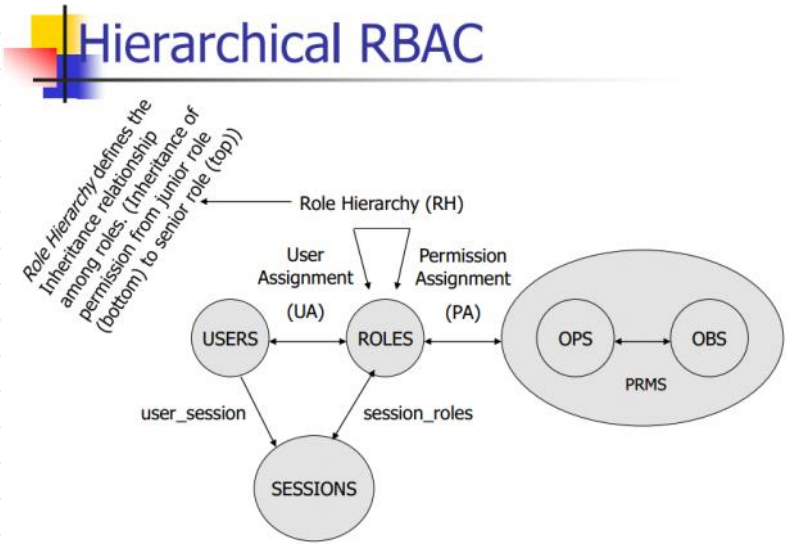
- RBAC Model and Formalize !!
  - Core RBAC (RBAC0)

-



- UA : user assignments
  - Many-to-many
- PA : Permission assignment
  - Many-to-many
- Session : mapping of a user to possibly many roles
  - Permissions : union of permissions from all roles
  - Each session is associated with a single user
  - User may have multiple sessions at the same time (one to many relationship)
  - In each session, multiple roles can be activated simultaneously (many to many relationship)
- **U**sers, **R**oles, **P**ermissions, **S**essions
- PA $\subseteq$ P x R (many-to-many)
- UA $\subseteq$ U x R (many-to-many)
- user : S → U, mapping each session s_i to a single user user(s_i)
- roles : S → 2^R, mapping each session s_i to a set of roles roles(s_i) $\subseteq$ {r | (user(s_i),r) $\in$ UA} and s_i has permissions $\cup_{r \in roles(si)}$ {p | (p,r) $\in$ PA}

- Hierarchical RBAC(RBAC1)

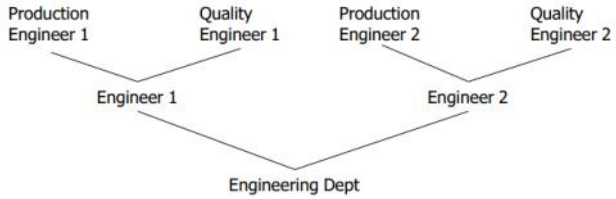


  - General role hierarchies
    - Include the concept of multiple inheritance of permissions and user membership among roles
  - Limited role hierarchies
    - Impose restrictions
    - Role may have one or more immediate ascendants, but is restricted to a single immediate descendent

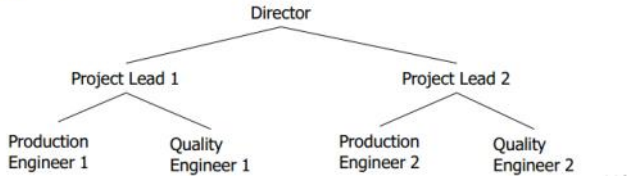
- **In the absence of role hierarchy, what happens?**

-

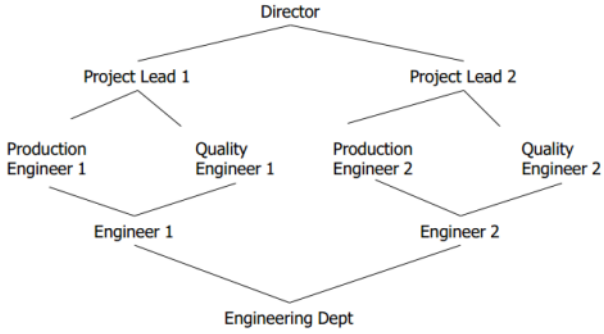
- Tree



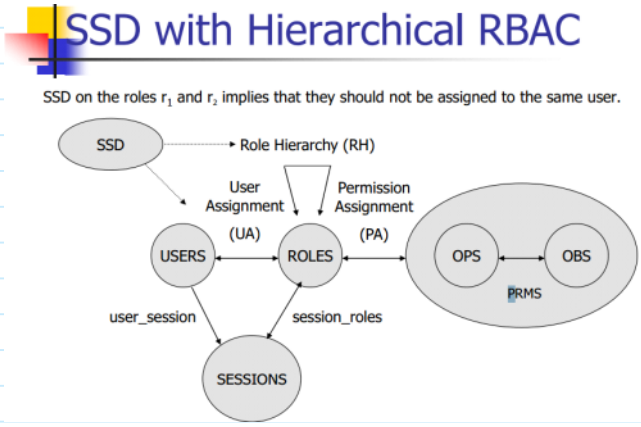
- Inverted Tree



- Lattice

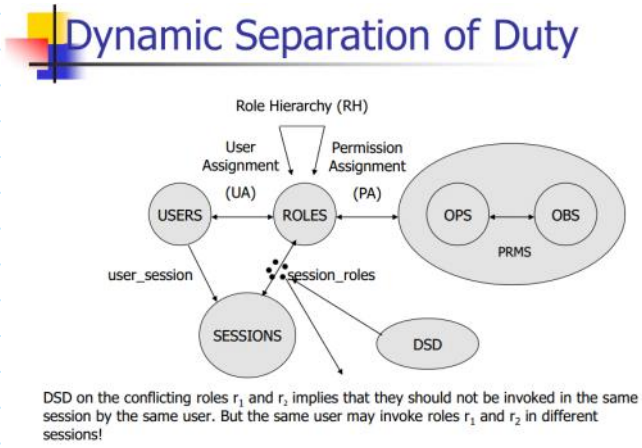


- Constrained RBAC(RBAC2)
  - Static Separation of Duty Relations
    - user cannot be authorized for both conflicting roles – mutually exclusive, e.g., teller and auditor
    - SSoD policies deter fraud by placing constrains on administrative actions and thereby restricting combinations of privileges that are made available to users



  - Dynamic Separation of Duty Relations
    - user cannot act simultaneously in both roles, e.g., teller and account Holder
    - DSoD policies deter fraud by placing constrains on the roles that can be activated in any given session thereby restricting combinations of privileges that

are made available to user



- Constrained RBAC(RBAC3)
  - Combines Constraints and Role Hierarchies

12. **Information Flow : Transmission of information from one "place" to another.**

- Consider a conditional statement
  - `if x == 1 then y = 0 else y = 1`
  - What do we know before & after execution?
  - What about : `if x == 1 then y = 0`
  - No explicit assignment to y in one case
- This is called implicit information flow
- Two categories of information flows
  - **explicit :** assignment operation
    - `y=x`
  - **implicit :** conditional assignment
    - `if x then y=z`
- **Information Flow Security Model :** Based upon flow of information rather than on access controls.
  - Flow of objects are controlled by security policy that specifies where objects of various levels are permitted to flow.
  - Prevents data leaking through "covert channels".
- **How do we measure/capture flow?**
  - By Entropy-based Analysis (Information Theory)
  - By Lattice-based Models

- **How do we measure/capture flow?**
  - By Entropy-based Analysis (Information Theory)
  - By Lattice-based Models

Entropy-based Analysis

- A precise quantitative definition for information flow can be given in terms of Information Theory.
- The information flow from x to y is measured by the equivocation (conditional entropy)
  - H(x | y) of x, given y.

(1) Entropy-based

샤론: A ->B 데이터 전송시 얼마만큼 많이 보내고 싶다.

- 대역폭을 늘리자
- 정보를 압축하자. 얼마나 압축해야 하나?

데이터 압축을 위해 정보이론(Information Theory) 연구시작

y가 주어졌을 때의 x의 정보량: H(x|y) of x, given y

책에 예제가 있다.

전산학자가 바라보는 정보보호 vs. 정보이론이 바라보는 정보보호

전산학자

공격자는 무한한 데이터 수집가능

공격자는 컴퓨팅 파워의 한계가 존재함

정보이론학자

공격자는 무한한 컴퓨팅 파워가 존재함

그러나 공격자는 정보수집을 무한히 할 수없음

Lattice-based Model

- The components of the information flow model are :
  - A lattice
  - A set of labeled objects
  - The security policy : Information flow from an object with label c_1 to an object c_2 is permitted only if :

    $$c_1 \leq c_2;$$

    any information flow that violates this is illegal (covert).

Lattice-based Model

- A system is 'secure' if there is no illegal information flow.
- **Advantages :** it covers all kinds of information flow.
- **Disadvantages :** far more difficult to design such systems.
  - (e.g.) checking whether a given system is secure in the information flow model is an undecidable problem.

Lattice-based Model

- One must also distinguish between
  - **static** enforcement : considers the system (program) as a static object.
  - **dynamic** enforcement : considers the system under execution.

of the information flow policies.

Lattice-based Model: Formulate requirements 장점을 가짐. 정책이 잘 구현됐는지 자동화되어서 점검이 가능하다.

Security policy를 정형화하면 Security Model이다.

13. 접근제어관리

- First an organization must choose the access control model (DAC, MAC, RBAC).
- Then the organization must select and implement different access control technologies.
- Access Control Administration comes in two basic forms :
  - Centralized
  - Decentralized

Centralized Administration

- One entity is responsible for overseeing access to all corporate resources.
- Provides a consistent and uniform method of controlling access rights.

Centralized Administration

- One entity is responsible for overseeing access to all corporate resources.
- Provides a consistent and uniform method of controlling access rights.
- Types of Centralized Access Control
  - Radius (Remote Authentication Dial In User Service)
  - TACAS (Terminal Access Controller Access Control System)
  - Diameter

상용제품들, 일반화된 보안정책

Decentralized Administration

- Gives control of access to the people who are closer to the resources
- Has no methods for consistent control, lacks proper consistency.

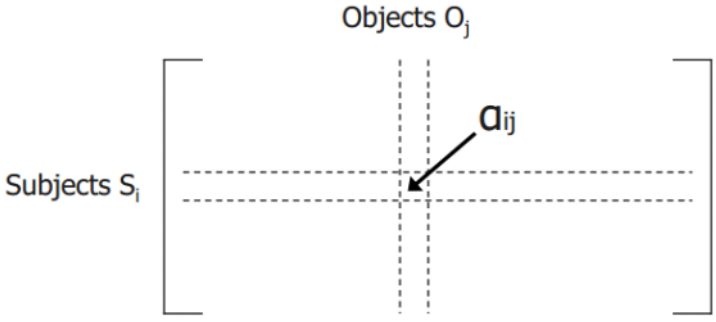
분산형으로 SPKI, SDSI, PKI기반 분산형 접근제어(속성형)

14. Access Control Structures

Structures (= Implementation Technique)

- Access Control Matrix
- Capabilities
- Access Control List (ACL)
- Role Based Access Control

Access Control Matrix :



Matrix는 중앙집중형에서 구현 가능

Capabilities & ACL

- **Capabilities (aka. Directory-based Access Control) :**
  - Rows of access control matrix
  - Store with user in the form of a token, ticket, or key.
  - e.g., Kerberos, Android
- **Access Control List :**
  - Columns of access control matrix
  - Store with the resource
  - e.g., Windows NT

Capabilities & ACL

Capabilities & ACL

- Example

	R_1	R_2	R_3	R_4
S_1	*rw*	*rwx*		
S_2		*x*	*rwx*	*rwx*
S_3	*rwx*	*r*		*r*

Capabilities:

S_1: {(R_1, *rw*), (R_2, *rwx*)}

S_2: …

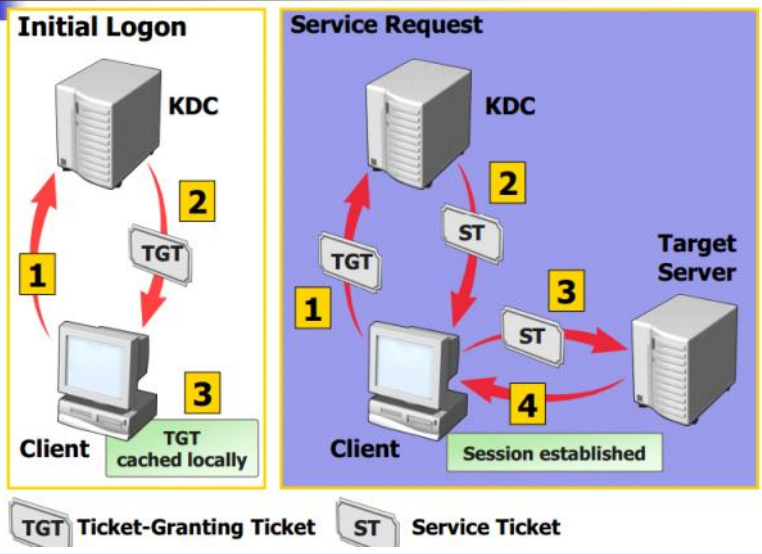
Access Control lists:

R_1: {(S_1, *rw*), (S_3, *rwx*)}

R_2: …

분산형 구현방법 두가지

(e.g.) Kerberos v5



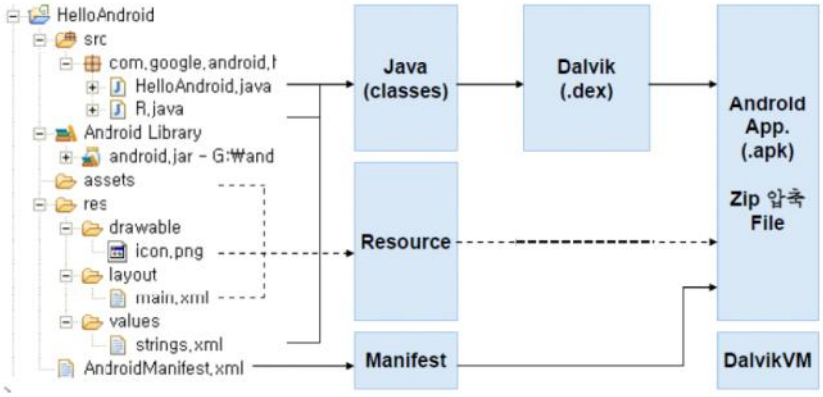
커버로스(SSO)는 왜 만들어졌나?

(1) 사용자 인증, (2) 서비스권한 획득을 분리하기 위해 만들어짐

그러나 이것은 일반적으로 동시에 일어남, 즉, 사용자인증은 한번만 이루어지며 접근제어는 여러 번 일어나게 된다.

커버로스(SSO)는 결국 일관된 보안정책(접근제어모델)을 적용하기 위해 도입되었다.

(e.g.) Android Access Control



Manifest는 퍼미션정보를 담고 있다. --> **Capabilities List**

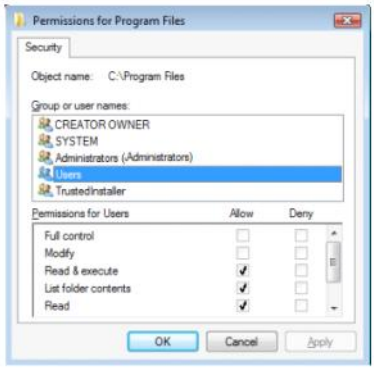
"이 프로그램은 GPS사용에 대한 접근을 허용하시겠습니까?"

Manifest가 변경되면 변경확인 요구

(e.g.) Windows Access Control

- **Access control lists (ACLs) :**
  - NTFS permissions
  - Share permissions

	Object_A	Object_B
USER_1	READ	WRITE
USER_2	EXECUTE	NONE
USER_N	READ WRITE	READ



많이 사용. 윈도우, F/W

ACLs VS.RBAC

Rule Based Access Control

- Uses specific rules that indicate what can and cannot happen between a subject and an object.
- Not necessarily identity based (The DAC model is identity based).
- Traditionally, rule based access control has been used in MAC systems as an enforcement mechanism.
  - Today, rule based access is used in other types of systems and applications as well (e.g., routers and firewalls).

CUI :Constrained User Interfaces 화면 접근제어 통제

- Restrict user's access abilities by not allowing them certain types of access, or the ability to request certain functions or information
- Three major types
  - Menus and Shells
  - Database Views
  - Physically Constrained Interfaces

CDAC :Content Dependent Access Control 패킷필터링 + 내용

- Access to an object is determined by the content within the object.
- This is often used in databases.
  - E.g., the content of the database fields dictates which users can see specific information within the database tables.
- Content dependent filtering is used when corporations employ email filters that look for specific strings, such as "confidential," "SSN," "top secret," etc.

CBAC: Context Based Access Control

- Makes access decision based on the context of a collection of information rather than content within an object.
  - Bases access decisions on the state of the situation, not solely on identity or content sensitivity.
- For example, firewalls make context based decisions when they collect state information on a packet before allowing it into the

- Makes access decision based on the context of a collection of information rather than content within an object.
  - Bases access decisions on the state of the situation, not solely on identity or content sensitivity.
- For example, firewalls make context based decisions when they collect state information on a packet before allowing it into the network.
  - A stateful firewall understands the necessary steps of communication for specific protocols.
  - Stateful : Something that understands the necessary steps of a dialog session. Is an example of context dependent access control.

15. Execution Monitors

- Enforcement mechanisms monitoring execution steps of a target system.
- Terminating target's execution if a violation of security is about to occur.
- Do not have a model of the system – can't predict.
- Cannot modify the system.

프로그램 실행상태를 보고 있다가 보안정책이 위반하는 상태를 보게되면 정지시키고 alert을 보여줌, 디버깅, tracing, auditing, logging에 사용

Line by line으로 reference monitor의 기능을 구현한 것

- View a program as defining an (infinite) set of (possibly infinite) execution traces.
- All executions on all possible inputs + powercut

프로그램이 내가 설정한 정책대로 수행되는 지 궁금? EM- BP가 걸리는 지 아닌지 확인, 모든 Input을 다 넣어서 확인해야 하는데...

랜덤으로 input해 보자..?

Security Policies - informal

- Define security policies as a subset of possible program execution traces
- Security policy set defines a predicate S

전체의(모든 가능한INPUT에 의해) 시퀀스 경우의 수가 발생.

EM은 일부만 수행한다.??

Enforcing Security Policies - informal

- Allows some traces that satisfy security policy
- Enforcement mechanism M is a concrete implementation that defines a subset of S

Execution Monitoring - informal

- Focusing on one Execution Trace
- Easy to do (just observe and constrain)
- EM can often approximate desired policy
- EM closely related to safety properties

EM의 설계원칙 ONE!, SAFETY!

Safety and Liveness - informal

[Lamport 77]

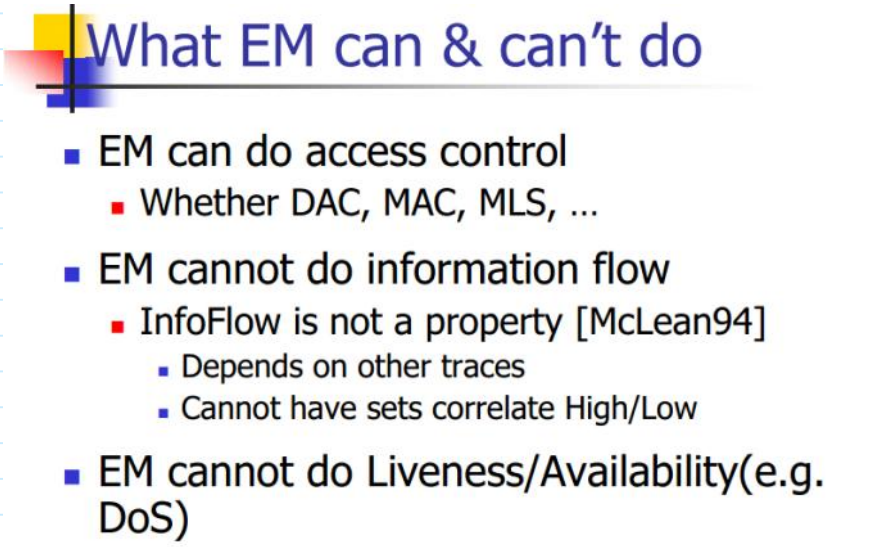
- **Safety Properties :** Some "bad thing" doesn't happen.
  - The 'safety' property of access matrices in the HRU model meets indeed this description.
- **Liveness Properties :** Some "good thing" eventually happens.

EM의 검증 2가지 방법

EM으로 쉽게 모니터링 가능한 정책(Enforceable policy): DAC, MAC, MLS

안되는 정책: Information Flow

Enforceable policy implies safety property!
슈나이더의 보안정책은 em으로 검증이 불가능하다... 왜냐하면 safety property를 만족시키지 못하고, 또한 Liveness Property를 만족하고 있기 때문이다..(결국에는 잘되겠지)



EM은 정보흐름 추적과, 가용성에 관련된 정책을 검증하지 못한다.
즉, EM으로 구현된 Security Kernel은 DDoS공격 대응의 방법과 같은 보안정책을 검증할 수 없음