**보안공학 2020**

**<Beginning>**

**Security Engineer** VS **Security Analyst**

(Build / fix) (Break)

**Security by Design : 설계 단계에서부터 보안을 고려**

* 증명이 되어 있어도 모의해킹(Pen-Test)은 시도해야함 : 증명의 전제조건 성립을 확인

“Penetrate & patch”의 사이클 반복으로는 충분하지 않음

1989, 미군에서 Security Engineering 관련 표준 제정

2011, Upgrade

2016, 군 표준이 연방정부 표준으로 확대됨

미국의 경우 군 표준 → 정부 표준 → 민간 표준 → 국제 표준 순으로 전파됨

엄격한 개발 프로세스를 가장 먼저 도입한 회사 : MS, IBM

SDL – Security Development Lifecycle

2002, 빌게이츠 “Trustworty computing” 전 직원 전송

2004, MS 전 제품 SDL 적용 결정

2007, Full SDL 적용한 첫 번째 OS : Windows Vista

개발 단계에서부터 보안을 고려해야하는 이유

* Standalone 위주였던 과거와 달리, 이제는 Embedded로 많은 것들이 서로 엮여서 작동

**Dependability(Trustworthiness)**

* 가용성(availability) : 시스템이 서비스를 요청 받았을 때, 서비스를 제공할 수 있는 능력
* 신뢰성(reliability) : 시스템이 명세화 된 대로 서비스를 제공할 수 있는 능력
* 안전성(safety) : 시스템이 재앙을 초래하는 고장 없이 운영될 수 있는 능력
* 보안성(security) : 시스템이 우연적 혹은 의도적인 침입을 막아낼 수 있는 능력

국내의 모범사례 : 카카오뱅크 (초기 설계부터 보안을 고려해서 복잡도를 낮춤)

특정 기능이 고장일 때 위험도는?

~임을 어떻게 보장할 것인가

SW공학

**개발의 첫 단계 : 요구사항 분석**

Requirement → Architecture → Code → Test → Runtime Monitoring

Provable Security는 Requirement에 대한 증명일 뿐 무조건적인 Security 보장이 아니다

* Proof 의 의미는? : 가능한 범위를 명확히 함으로서 한정된 자원을 효율적으로 활용

보안 도구의 검증은 가능하면 자동화 도구를 이용

* 사람이 수행할 경우 개인 역량에 따른 편차 존재

**<Basic Terms>**

Physical Security → Communication Sec. → Computer Sec. → Information Sec. →

→ Information Assurance → Cyber Security

**Info. Security** : 컴퓨터 및 컴퓨터 내부의 디지털 데이터를 보호 (Protect)

**Info Assurance** : 디지털 외에 모든 정보를 포괄적으로 보호 (Accessible, Authentic, Trustworthy)

* 사이버불링 이나 거짓뉴스는 IS는 아니지만 IA에서는 고려대상

Cyber Defense = Cyber Sec. + Strategy

1991년 걸프전 : 최초의 정보전 (Information War), IS → IA 전환의 계기

* TV 중계가 있었던 전쟁 (How : 항공모함, 미사일 등등)

**Information Assurance**

* Access Information
* Preserve Assurance - Quality (Against Accidental Failure)  
   - Security (Against Intentional Failure) + = → Trustworthy

**Security by Design**

* 설계 단계에서 Security를 Availability, Reliability, Safety 와 조화되도록 고려  
  (설계 복잡도 최소화)
* 단순히 Security의 중요성을 강조하는 게 아님

3차, 4차 산업혁명의 차이

* 공통점 : 정보화 & 자동화
* 차이점 : 3차는 정보화&자동화가 일부 영역이지만 4차는 모든 영역 (ex IOT)
* 편의 장치가 많아질수록 망분리는 어려워지고 embedded의 어려움

UNECE(유엔 유럽경제위원회)에서 자동차 개발에서 보안지침을 명시

* Development phase → Production phase → Post-Production Phase  
  (Cyber Security Management System)

**RMF(Risk Management Framework)**

* = System Development Lifecycle + Security by Design + Risk management
* 목적 : information 대신 information process / checklist 대신 discussion about Risk 로의 전환

Platform IT ↔ non-Platform IT

미사일, 탱크 등 무기체계에 들어가는 IT시스템 국방부 사무실 PC

록히드 마틴 RL 모델 (Cyber Resilience Level) – 직역 : 사이버 복원 레벨

* 사이버 복원력은 변화하는 조건을 예측, 견딜 수 있고 복구하고 적응하여 임무 수행 능력에 필요한 기능을 유지하는 능력입니다.
* F-35 : 전 세계의 F35가 통신 연결되므로 동맹국 중 한 곳이 뚫리면 전 세계의 F-35가 취약해지므로 동맹국에 RMF 표준을 따를 것을 요구

**Trusted vs Trustworthy**

* Trust : 가정 (given assumption)
* Trustworthy : 실제로 입증된

**RISK**

Vulnerability

**Risk는 없앨 수 없고 줄일 뿐**

* Checklist로는 불가능
* Monitoring을 함의, 언제 다시 줄여야 하는가?

**Risk Management**

* Risk Avoidance : Risk 유발 원인, 효과를 같이 제거
* Risk Transfer : 사용자 약관, 보험 처리
* Risk Reduction : 실제로 Risk 원인을 개선 및 보수
* Risk Retention : Risk를 감당
* + Risk Acceptance : 위 4개의 잔여 Risk 총합이 Threshold 미만인가?

**By Security Engineering**

* Risk based approach
* Structured engineering process
* Goal : High Assurance (End-to-End Provable)

Policy Assurance → Design Assurance → Implementation Assurance → Operational Assurance

**Chain of Evidence**

= 서류

1) 수학적으로 증명  
2) 증명이 불가능한 경우, 최대한 상세한 매뉴얼  
 모든 공정상의 절차를 기록

**SDL**

* Training & Awareness를 포함
* (1) Shift Left (보안 팀의 업무를 개발자에게 이동)  
   보안 팀은 개발팀 교육 & 지원 업무 (ex : 툴 개발, 검증, 컨설팅 등)
* (2) Traceability (End-to-End traceability)  
   Rely on root cause analysis & continuous improvement (근본 원인 분석 및 지속적 개선)  
   문서화가 필수적

**<Fundamental Design Principles for SE>**

**Saltzer and Schroeder 의 8가지 원칙 (용어 및 정의 명확히)**

**1. Principle of least privilege (최소 권한의 법칙)**

* 필요한 최소 권한만 부여

**2. Principle of fail-safe default**

* 명확하지 않으면 거절
* 만약 시스템이 터지면 바로 직전의 안전 상태로 돌아갈 것 (중간 애매한 상태 X)

**3. Principle of economy of mechanism**

* 시스템은 가능하면 간단하게

**4. Principle of complete mediation**

* 시스템의 모든 행위(access to object) 권한 체크가 가능해야 함 (실시간 수행이 정말 어려움)

**5. Principle of open design**

* Kerckhoff’s principle, 모든 정보가 공개되도 안전해야 한다 (반대 : security by obscurity)

**6. Principle of separation of privilege**

* 중요 시스템의 권한은 분산 관리 (ex 핵 스위치)

**7. Principle of least common mechanism**

* 공유 자원의 최소화 (한 곳의 문제가 전파되지 않도록)

**8. Principle of psychological acceptability**

* Usable security

Onion model of protection

정교함 + 특화된 서비스  
제공 가능

점점 보편적인 기능 제공 가능

Application

Service

OS

Kernel

Hardware

Centralized vs Decentralized

보안의 일관성 유지에 적합 vs 가용성 확보에 유리

**<Threat Risk Modeling>**

설계에 필요한 보안요구사항을 체계적으로 식별

‘Engineering’ 재현 가능성, 누가 진행해도 일정 수준 이상의 결과

**Threat modeling helps**

* Determine attack surface 공격 지점 정의
* Assign risk to vulnerabilities 취약점의 리스크 할당
* Drive the vulnerability mitigation process 취약점 완화 프로세스 추진

**Threat modeling steps**

1) decide scope of analysis 범위 설정

2) identification of assets 자산 식별

3) identify threats & possible vulnerabilities (STRIDE) 위협 및 가능한 취약점 식별

4) rank threat risk (DREAD) 취약한 위험 순서

5) response to risk (risk mitigation or countermeasure) 위험에 대한 대응 (위험 완화 또는 대책)

Modeling = Abstraction

* Abstraction gap : model과 현실의 차이 ex) DFD

**DFD basic rule**

1) (functional) Decomposition : complete 해야 함

2) process – input/output이 모두 있어야 하고, 명사보다는 동사 위주 명칭

3) Data store – 명사형, 모든 data의 이동은 process를 거쳐야 함

4) Source/Sink

5) Data flow – 모든 flow는 단방향 (불가피한 경우 양 방향을 단 방향 2개로 따로 표기)  
 완전히 같은 데이터인 경우 한 선에서 두 개로 나뉠 수도 있음

DFD → STRIDE + Attack Library → Attack Tree → Remediation

분석하려는 시스템에 대해서는 CVE 수준까지 알고 있어야 함

**DREAD 방법** : MS에서 만들었으나 2010년 이후 사용 X

* Damage 피해가 얼마나 클지
* Reproducibility 공격을 재현하기 쉬운지
* Exploitability 공격하기 위한 작업량이 얼마나 많은지
* Affected users 영향받는 유저는 얼마나 많은지
* Discoverability 발견하기 얼마나 쉬운지

Design Assurance

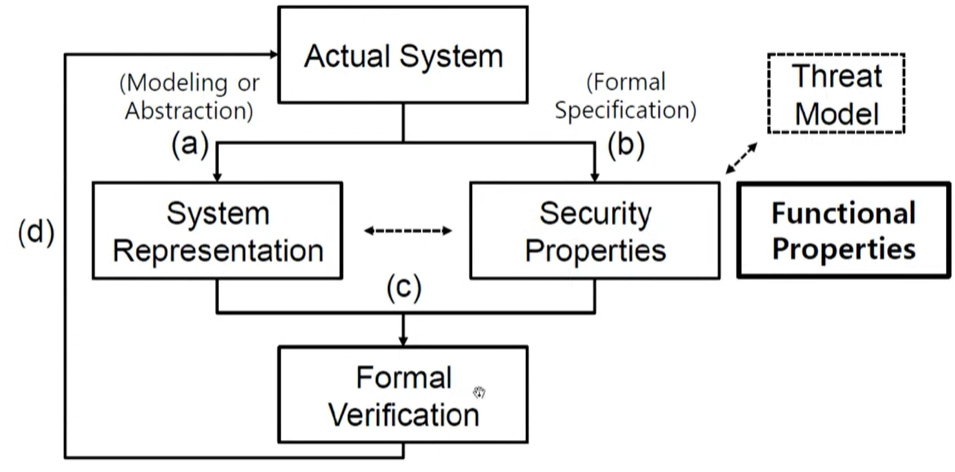
기호로 표기

* 해석 오류 방지
* 설계의 자동 검증 (by model checker)  
  > 디자인이 명제를 반영하는가  
  > 명제들 사이에 서로 모순되는 사항이 없는가

**<Security Policy Modeling>**

**Formal Modeling** : 자연어를 수학적 기호로 표기하고 이를 검증하는 것

* Formal Specification : 수학적 기호로 표현하는 일
* Formal Verification : 내가 만든 Specification이 옳은 지 검증하고, 프로그램이 맞는지 검증하는 일

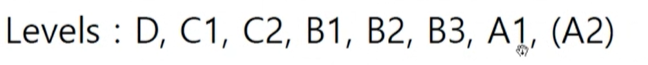


**Security Properties** (기본 Confidentiality, Integrity, Availability 기밀성, 무결성, 가용성)

* Non-interference : 높은 권한 가진 애가 하는 일이 하위 애에게 보여지면 안됨
* Isolation : 두 같은 애의 활동이 서로에게 영향을 주면 안됨
* Type Safety : char != int 같은 타입 변수들끼리 주고 받는 거
* Memory Safety : 메모리 엑세스 할 때 오류 안 생기는 거 (ex. Buffer overflow)
* Memory Integrity : 허가되지 않은 사용자에게 메모리 상 변경 안되는 것
* Execution and code integrity : 코드 무단으로 사용 및 수정되지 않게 하는 것

이런 것들을 통해서 Security Properties를 입증하게 됨

**Orange Book : 최초의 보안 운영체제 평가 기준 (TCSEC)**

D등급부터 A2등급까지, 

A1부터는 설계가 제대로 돼서 Formal하게 적힌 Security Requirement를 수학적으로 충족한다. = Design Assurance

A2는 프로그래밍 코드가 수학적으로 됐는지 증명 = Code Assurance

A2는 시기적으로 어렵다 판단하고 당시에 없앰

**Security Policy**

* What과 how에 대해 자연어로 기술 > Confidentiality, Integrity and Availability 기밀성, 무결성, 가용성

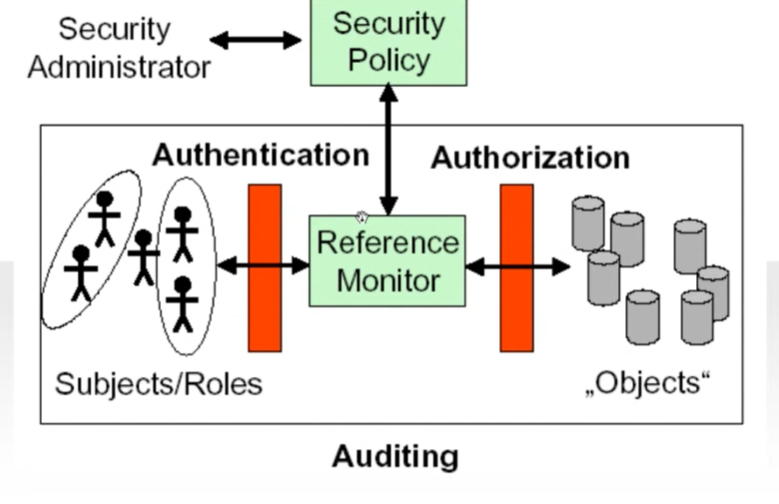
>수학적 기호로 , formal하게

**Security Policy Model (SPM)**

* SP를 formal method를 통해 정형화하고, 이를 formal verification까지 하는 것

**SP Assurance**

요구사항 알맞게 구성했는지, 충돌되지 않고 완벽한지, 테스트가 가능한지, 설계까지 잘 연결돼 반영



Reference Monitor : 개념적 뷰어, 접근통제 결정을 중재하는 OS 요소 / 모든 접근 요청이 PASS하는 Single Point / 객체로의 접근이 요청될 때만 작동

Security Kernel : 개념적 뷰어의 구현

Trusted Computing Base(TCB) : 개념적 뷰어를 포함한 주변 장치

Authentication : ID를 증명하는 방법 (ID, PW 치기) – 인증

Identification : 사용자의 ID를 구축하는 방법 (사건현장에서 지문 추출 후 신원정보 조회) – 식별

Authorization : 권한 관리

**DAC(임의적 접근 통제)**

* 사용자(또는 그룹)의 ID에 기반해 접근을 통해 (ex. UNIX 파일 시스템)
* Copy에 대한 통제가 없음

**MAC(mandatory 강제적 접근 통제)**

* Security Level을 부여 - Linear Model의 경우 최소권한 법칙 X  
   - MLS(Multi Level Security)모델 필요, 영역별 구분

**Bell-La Padula (BLP 모델)** : 최초의 formal & mathematic 한 MAC 모델 제시

* 기밀성만 다룸, 접근 권한 수정에 대한 정책이 없음, 은밀 채널(Covert Channel)로 정보 흐름을 다루지 않음

설계가 요구조건을 만족하는지 검증

* Covert Channel : 상호 협력 필요 (sender / receiver) – 은밀채널
* Side Channel : 불필요

**Chinese Wall model**

* SS-Property : Prevents a subject from being exposed to a conflict of interest   
   주제가 이해의 상충에 노출되는 것을 방지(?)
* \*-Property : write access granted if no other object can be read that  
   다른 객체를 읽을 수 없는 경우 쓰기 액세스 권한 부여  
   - belongs to a different company dataset (다른 회사의 데이터 셋에 속함)  
   - contains un-sanitized information (?)  
   un-sanitized 정보가 회사 데이터셋에서 유출되는 것을 방지
* 직무분리와 이해상충 개념을 접근통제에 반영한 개념, 정보흐름 모델을 기반으로 이해충돌을 야기하는 주체와 객체들 사이에 정보가 흐르지 않게 함.
* Sanitized data : 시스템에 해를 끼칠 수 있는 정보가 포함되어 있는지 확인하기 위해 소프트웨어에서 확인하는 컴퓨터 데이터 (일반적으로 사용자 입력)입니다. 예를 들어, 게시판에 주석에 HTML을 포함하면 컴퓨터에서 HTML을 명령으로 해석할 수 있으므로 메시지를 보는 사람에게 해로울 수 있습니다.

**RBAC모델 (Role based Access Control)**

Session 존재 의의 : 한 사람이 복수의 role을 소지할 수 있을 때  
 시간에 따른 변화를 고려  
 (ex. A가 은행의 직원이자 고객일 수 있다, 동시 X)

Static separation of duty : (시간과 무관하게) 함께 가질 수 없는 role (ex. 감사팀 vs 피평가자)

Dynamic separation of duty : 동시에 가질 수 없는 role (ex. 직원 & 고객)

**Information flow**

* Explicit : assignment (y=x) < 분석 Easy
* Implicit : conditional assignment < 분석 Hard  
  “if x then y=z” y를 관찰함으로써 x에 대한 정보 획득 가능

Information Flow Security Model

: entropy-based vs lattice based

**Lattice**

* Guarantee secure information flow as an ordering relation