PART 01. 정보보호 개요

01. 정보보호관리의 개념 (출제빈도 1.3%)

[정보보호(Information Security)]

• 정의

- 정보의 수집, 가공, 저장, 검색, 송신, 수신 중에 발생하는 정보의 훼손, 변조, 유출 등을 방지하기 위한 관리적, 기술적 수단, 또는 그러한 수단으로 이루어지는 행위

- **기밀성, 무결성, 가용성, 인증성, 부인방지를 보장하기 위해 기술적 · 물리적 · 관리적 보호대책을 강구하는 것**

• 3대 목표 + 속성(서비스)

1. **기밀성(Confidentiality)** : 오직 인가된 사람 · 프로세스 · 시스템만이 알 필요성(Need-to-know)에 근거하여 시스템에 접근할 수 있어야 한다.
2. **무결성(Integrity)** : 정보의 내용이 무단으로 생성 또는 변경되거나 삭제되지 않도록 보호되어야 한다.
3. **가용성(Availability)** : 시스템이 지체 없이 동작하고, 자원이 필요할 때 권한이 있는 사용자가 이용할 수 있어야한다.
4. **인증(Authentication)** : 통신 대상에 대한 인증, 데이터의 출처에 대한 인증
5. **부인방지(Non-repudiation)** : 전송이나 수신한 사실을 부인하지 못하도록 하는 것
6. 접근제어(Access Control)

[보안 공격(Security Attack)]

Attack	Passive/Active	Threatening
Snooping 트래픽 분석(Traffic Analysis)	Passive(소극적 공격)	Confidentiality
Modification(변경) Masquerading(가장) Replaying(재연, 재전송) Repudiation(부인)	Active(적극적 공격)	Integrity
Denial of Service	Active	Availability

[시점별 통제]

- **예방통제** : 사전에 위협과 취약점에 대처하는 통제
- **탐지통제** : 위협을 탐지하는 통제. 빠르게 탐지할수록 대처하기 용이
- **교정통제** : 이미 탐지된 위협이나 취약점에 대처/감소시키는 통제
- 예방 → 탐지 → 교정 순으로 적용

[주요 보안 용어]

- 자산(Asset) : 조직이 보호해야 할 대상
- 취약점(Vulnerability) : 위협의 이용대상으로 관리적, 물리적, 기술적 약점을 의미
- 위협(Threat) : 손실이나 손상의 원인이 될 가능성을 제공하는 환경의 집합
- 위험(Risk) : 공격자(위협 주체)가 취약점을 이용하여 위협이라는 행동을 통해 자산에 악영향을 미치는 결과를 가지고 올 가능성

PART 02. 암호학

02. 암호학 개요 (출제빈도 2.1%)

[암호화와 복호화]

* 평문(Message, Plain-text) : M, P
* 암호문(Cypher-text) : C
* 암호화(Encryption) 알고리즘 : E
* 복호화(Decryption) 알고리즘 : D
* 키(Key) : K

• 암호화, 복호화의 기호적 표현
  - 암호화 : C = E(K,P) / $\mathrm{C} = E_k(\mathrm{P})$
  - 복호화 : P = D(K,C) / $\mathrm{P} = D_k(\mathrm{C})$

[치환 암호와 전치 암호]

• 치환 암호(대치 암호, Substitution Cipher) : 평문의 문자를 다른 문자로 교환(대체)하는 암호기법
• 전치 암호(Transposition) : 문자 집합 내부에서 자리를 바꾸는(재배열) 암호기법. 평문의 문자 집합과 암호문의 문자 집합이 일대일 대응 규칙을 갖는다.

[블록 암호와 스트림 암호]

• 블록 암호(Block Cipher) : 평문을 특정 비트의 블록으로 잘라낸 후 암호화 알고리즘을 적용하여 암호화. ex) DES, AES
• 스트림 암호(Stream Cipher)
  - 데이터 흐름(스트림)을 순차적으로 처리해가는 암호 알고리즘. ex) LFSR
  - 데이터 흐름을 순차적으로 처리하기 때문에 내부 상태를 가지고 있다.
  - 긴 주기와 높은 선형 복잡도가 요구된다.
  - 속도는 블록 암호화 방식보다 빠르지만 암호화 강도가 약하므로 이동 통신 등의 무선 데이터 보호에 적합.

[위치에 따른 암호화 구분]

• 링크 암호화(Link Encryption) : 통신 링크 양쪽 끝단에 암호화 장치를 놓는 것. 대형 네트워크에서 사용하면 필요한 암호화 장치 수가 많아지고, 중간 노드에서 데이터가 평문으로 노출된다는 단점이 있다.
• 종단간 암호화(End-to-End Encryption) : 두 종단 시스템에서 수행됨. 트래픽 분석에는 취약하지만 링크 암호화 방식보다 높은 수준의 보안 서비스를 제공한다.

[주요 암호 기술]

• 대칭키 암호와 비대칭키 암호
  - 대칭키 암호(Symmetric Cryptography) : '암호화키 = 복호화키'인 암호 알고리즘 방식
  - 비대칭키 암호(Asymmetric Cryptography) : '암호화키 ≠ 복호화키'인 암호 알고리즘 방식
• 하이브리드 암호 시스템 : 대칭키 암호 + 비대칭키 암호
• 일방향 해시함수(one-way hash function)
• 메시지 인증 코드(MAC, Message Authentication Code) : 무결성과 인증을 제공
• 전자서명 : 무결성을 확인하고, 인증과 부인방지를 제공
• 의사난수 생성기(PRNG, Pseudo Random Number Generator) : 키 생성의 역할

[암호 분석(암호 해독) 분류]

• **암호문 단독 공격**(COA, Cipher-text Only Attack) : 암호문 C만을 갖고 평문 P나 키 K를 찾아내는 방법.

• **기지 평문 공격**(KPA, Known Plain-text Attack) : 일정량의 평문 P에 대응하는 암호문 C를 알고 있는 상태에서 해독하는 방법. 암호문 C와 평문 P의 관계로부터 키 K나 평문 P를 추정하여 해독하는 방법

• **선택 평문 공격**(CPA, Chosen Plain-text Attack) : 송신자(암호기)에 접근이 가능하여 평문 P를 선택하면 그 평문 P에 대한 암호문 C를 얻어내 해독하는 방법.

• **선택 암호문 공격**(CCA, Chosen Cypher-text Attack) : 수신자(복호기)에 접근이 가능하여 암호문 C를 선택하면 그 암호문 C에 대한 평문 P를 얻어내 암호를 해독하는 방법

• COA → KPA → CPA, CCA 순서로 공격자의 능력이 향상. 따라서 높은 단계의 공격자에게 안전한 암호 방식을 설계하는 것이 일반적인 암호 설계 방향이다.

[암호 알고리즘의 안정성 평가]

- CC(Common Criteria) : 정보보호시스템에 대한 공통 평가 기준

- CMVP(Cryptographic Module Validation Program) : 미국 NIST와 캐나다 CSE가 개발한 암호 모듈의 안전성 검증을 위한 프로그램. 암호기술의 구현 적합성 평가, 암호키 운용 및 관리, 물리적 보안으로 크게 나뉘어 평가를 수행

[디지털 저작권 관리]

• 스테가노그래피(Steganography) : 메시지 내용이 아닌 메시지 자체를 은폐하는 기법

• 워터마크(Watermark) : 원본의 내용을 왜곡하지 않는 선에서 저작권 정보를 디지털 콘텐츠에 삽입하는 기법

- 강한(강성) 워터마킹 : 공격을 받아도 쉽게 파괴되거나 손상을 입지 않음
- 약한(연성) 워터마킹 : 공격을 받으면 쉽게 파괴되거나 손상을 입음

• **핑거프린팅**(Fingerprinting)

- 디지털 콘텐츠를 구매할 때 구매자의 정보를 삽입하는 기법
- 불법 배포 발견 시 최초의 배포자를 추적하기 위한 기술

• **디지털 저작권 관리**(DRM, Digital Rights Management)

- 디지털 콘텐츠 소유자가 자신의 콘텐츠에 대한 접근을 자신 또는 자신의 위임자가 지정하는 다양한 방식으로 제어할 수 있게 하는 기술적인 방법

- 구성 요소

1. 메타데이터(Metadata) : 콘텐츠 생명주기 범위 내에서 관리되어야 할 각종 데이터의 구조 및 정보
2. 패키저(Packager) : 보호 대상인 콘텐츠를 메타데이터와 함께 Secure Container 포맷으로 패키징 하는 모듈
3. 시큐어 컨테이너(Secure Container) : DRM의 보호 범위 내에서 유통되는 콘텐츠의 배포 단위
4. 식별자(Identifier) : 콘텐츠를 식별하기 위한 식별자
5. DRM 제어기(DRM Controller) : 콘텐츠를 이용하는 사용자의 PC 또는 디바이스 플랫폼에서 콘텐츠가 라이선스에 명시된 범위 내에서 지속적으로 보호될 수 있도록 프로세스를 제어

- DRM이 적용된 기술

(1) PKI 기반의 불법복제 방지 기술 : 콘텐츠를 소비자의 암호화 키를 이용하여 패키징 함으로써 이를 다른 사람들이 이용할 수 없도록 하는 방식. 콘텐츠 배포 서버의 부담이 크다는 단점이 있어 디지털 콘텐츠 유통에 적합하지 않음

(2) DOI(Digital Object Identifier) 기반의 저작권 보호 기술 : 저작권 관리 정보를 바탕으로 저작권 인증을 부여하는 기술. 불법복제 및 불법사용 방지 기능이 제공되지 않아 적극적인 저작권 보호가 불가능

03. 대칭키 암호 (출제빈도 3.3%)

• 대칭키 암호 = 관용 암호 = 공통키 암호 = 비밀키 암호

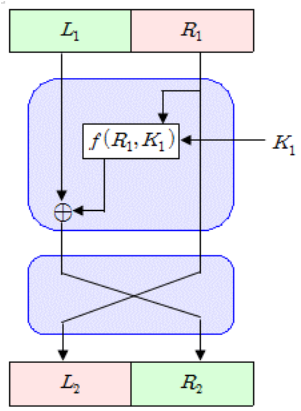
[블록 암호]

• 확산과 혼돈
  - Shannon이 정의한 개념
  - **확산(diffusion)** : 암호문 C와 평문 P 사이의 관계를 숨기는 것. C ↔ P
  - **혼돈(confusion)** : 암호문 C와 키 K의 관계를 숨기는 것. C ↔ K

• P-box (전치 장치, Permutation box) ← 확산 제공
  - 단순 P-box : n 비트를 입력 받아 n 비트를 출력, 유일하게 역함수가 존재
  - 축소 P-box : n 비트를 입력 받아 m 비트를 출력 (n > m)
  - 확장 P-box : n 비트를 입력 받아 m 비트를 출력 (n < m)

• S-box (치환 장치, Substitution box) ← 혼돈 제공
  - 역함수가 존재하는 S-box : 입력 비트 = 출력 비트
  - 역함수가 존재하지 않는 S-box : 입력 비트 ≠ 출력 비트

• **Feistel 암호 구조**
  - 암호 강도를 결정짓는 요소 : 평문 M 의 길이, 키 K의 길이, 라운드의 수
  - 암호화 과정 = 복호화 과정



• **SPN 암호 구조**
  - Substitution-Permutation Network
  - S-box, P-box 반복적으로 사용하는 구조

[블록암호에 대한 공격]

- 차분 분석 : 평문의 한 비트를 변경하면 암호문은 전혀 다른 비트 패턴으로 변화하므로, 이 변화 형태를 분석하는 공격
- 선형 분석 : 평문 비트와 암호문 비트를 일부 XOR하는 과정으로 근사적 선형 관계를 찾는 방법.
- 전수 공격 : 가능한 모든 경우에 대해 공격하는 방법. 경우의 수가 적은 경우 효율적인 방법
- 통계적 분석 : 통계적인 자료를 통해 해독하는 방법
- 수학적 분석 : 통계적 분석을 포함하여 수학적 이론을 이용하여 해독하는 방법

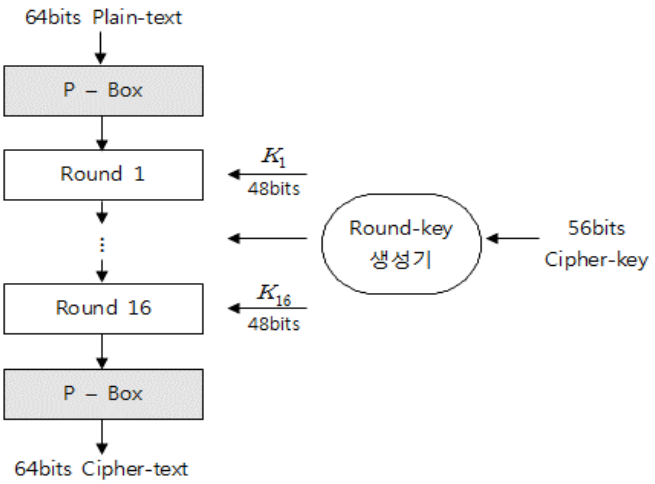
[스트림 암호]

- 동기식 스트림 암호
  - 키 스트림은 평문 or 암호문 스트림과 **독립적**이다.
  - 암호화, 복호화에서 상호 동기화가 필수적이며, 전송도중 변조되어도 후속 암호문에 오류의 영향이 없다.
  - One-Time Pad
    1) 암호화를 수행할 때마다 랜덤하게 선택된 키 스트림을 사용
    2) 이론적으로 해독이 불가능하다는 것이 Shannon에 의해 증명됨.
  - 귀환 시프트 레지스터 (Feedback Shift Register, FSR)
  - 선형 귀환 시프트 레지스터 (Linear Feedback Shift Register, LFSR) : 많은 스트림 암호가 이를 이용하며, 선형성 때문에 공격에 취약
  - 비선형 귀환 시프트 레지스터 (Non-Linear Feedback Shift Register, NLFSR)
- 비동기식(자기 동기식) 스트림 암호
  - 키 스트림은 평문 or 암호문 스트림에 **종속적**이다.
  - 암호문이 전송도중 변경되어도 자기 동기화가 가능
  - 변조된 암호문이 후속 암호문 복호화에 사용되지 않아 후속 암호문에 오류의 영향이 없다.

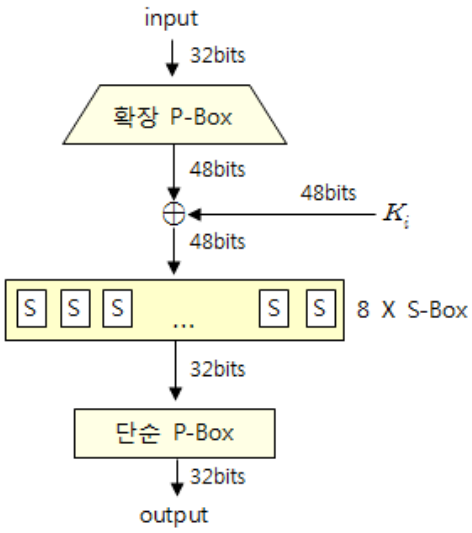
[DES(Data Encryption Standard)]

- 평문 = 64bits / 키 = 56bit (오류검출비트 8bit 제외) / 라운드 횟수 = 16회
- 두 개의 P-box와 16개의 Feistel 라운드 함수로 구성
- 키 생성기에 의해 48bit의 라운드 키가 생성

- DES 구조



• DES 함수 : 확장 P-box → 라운드 키와 XOR 연산 → 8개의 S-box → 단순 P-box로 구성되어 있다.
  - S-Box는 비선형 함수이며, 혼돈을 제공



• DES의 취약점 : 56bits의 작은 키

[3DES(3중 DES)]

• H/W에서 효율적이지만 S/W에서 비효율적
• 처리 속도가 느리다.
• 두 개의 키를 갖는 3DES
  - 암호화 - 복호화 - 암호화 과정을 거쳐서 암호문이 만들어진다.
• 세 개의 키를 갖는 3DES
  - 두 개의 키를 갖는 3DES에 대한 기지평문공격(KPA) 가능성 때문에 사용
• 3DES에서 모든 키를 동일하게 하면, 보통의 DES가 된다.

[AES(Advanced Encryption Standard)]

• NIST(미국 국립기술표준원)에서 공모한 암호 알고리즘으로, **라인달(Rijndael)**이 채택되었다.
• **non-Feistel** 알고리즘, **SPN 구조**.
• 키의 길이에 따라 라운드가 바뀐다.

Key Size	128bits	192bits	256bits
Round	10	12	14

• AES 단계
  0) 평문을 byte 단위로 나눔
  1) SubBytes (바이트 치환) : 바이트 단위로 치환
  2) ShiftRows (행의 이동) : 행 단위로 순환 시프트 수행
  3) MixColums (열의 혼합) : 열 단위로 혼합(Mixing). 높은 확산을 제공. 마지막 라운드에서 수행 X
  4) AddRoundKey : 라운드 키와 XOR 연산

[기타 대칭키 암호 알고리즘]

• IDEA (International Data Encryption Algorithm)
- DES를 대체하기 위한 알고리즘
- K = 128bits, M = 64bits, 블록암호 구조
- 8라운드. 마지막에 한 번 더 키를 적용시킴.
- DES보다 안전하며, PGP(Pretty Good Privacy)의 암호 알고리즘으로 사용되고 있다.

• RC5
- 미국에서 개발, RSA의 Rivest가 개발. (Feistel 구조)
- 비교적 간단한 연산으로 빠른 암호화와 복호화 기능을 제공.
- 모든 H/W에 적합

• Skipjack
- 음성을 암호화하는데 주로 사용

• SEED
- 한국에서 개발된 암호 알고리즘 (Feistel 구조)
- K = 128bits, M = 128bits, Round = 16
- 2009년에 키가 256bits로 확장되어 더 강력한 암호화 기능 제공 (SEED-256)

• ARIA
- NSRI(국가보안기술연구소, 한국) 주도로 개발된 알고리즘 (SPN 구조)
- K = 128/192/256bits, M = 128bits

• HIGHT
- 저전력, 경량화를 요구하는 컴퓨팅 환경에서 기밀성을 제공하기 위해 개발된 알고리즘
- M = 64bits

• LEA
- 128bits 경량 고속 블록 암호 알고리즘
- 다양한 정보보호 서비스에서 대용량 데이터를 빠르게 처리하거나 스마트폰 보안, 사물 인터넷(IoT) 등 저전력 암호화에 사용 가능

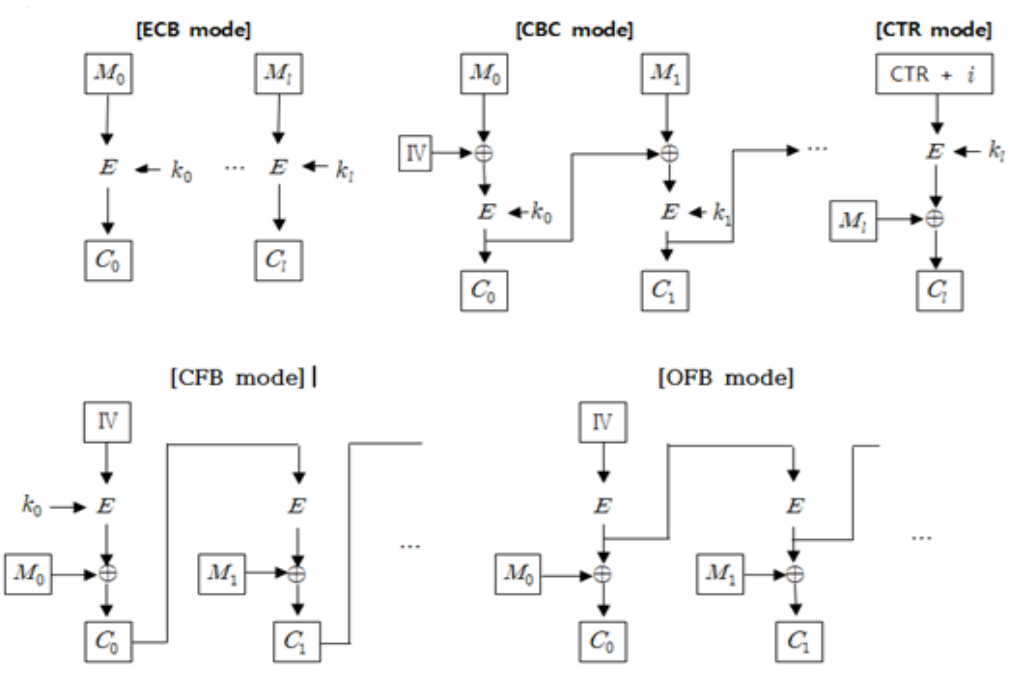
• Feistel 구조와 SPN 구조 알고리즘 분류

(1) Feistel 구조
- DES, RC5, SEED, LOKI, CAST, Blowfish, MISTY, Twofish, Mars 등

(2) SPN 구조
- Rijndael(AES), ARIA, CRYPTON. SAFER, SHARK, Square 등

[블록 암호 사용 방식]

이름	특징	암호모드
ECB모드 (Electric CodeBook)	· 가장 간단하고 빠르며, 병렬 처리 가능 · 평문 M의 크기가 블록 크기의 배수가 아니라면, padding이 필요. · $M_A = M_B$ 이면 $C_A = C_B$ 이다.	블록암호 (권장X)
CBC모드 (Cipher Block Chaining)	· 복호화 시 병렬처리가 가능하지만 암호화 시에는 병렬처리 불가능 · 첫 번째 평문 블록 암호화시 이전 암호문이 없으므로, 초기 벡터(IV, Initialization Vector)가 사용된다. · 암호화 할 때 하나의 평문 에러 발생 시, 이후의 모든 암호문에 에러가 발생한다. · 복호화 할 때 하나의 암호문 에러 발생 시, 두 개의 평문에 에러가 발생한다.	블록암호 (Practice Crypto-graphy 권장)
CFB모드 (Cipher FeedBack)	· 어떤 블록 암호도 스트림 암호(비동기식)로 바꿀 수 있다. → padding이 필요 없다. · 복호화 시에도 암호화 함수 사용. · 암호화, 복호화 시 블록 암호의 암호화 함수를 이용	스트림암호 (비동기식)
OFB모드 (Output Feedback)	· 암호문 C에서 비트 손실이 발생하면 그 다음에 오는 평문은 모두 에러가 발생하기 때문에 동기를 새로 맞추어야 한다. · IV가 바뀌면 암호문이 모두 바뀜 · 잡음이 있는 채널상의 스트림암호에 많이 사용한다. · 복호화 시에도 암호화 함수 사용.	스트림암호 (동기식)
CTR모드 (CounTeR)	· ECB모드처럼 독립적으로 암호화/복호화를 수행한다. · 복호화 시에도 암호화 함수 사용. · ATM 보안, IPsec 보안에 사용	스트림암호 (Practice Crypto-graphy 권장)



04. 비대칭키 암호 (출제빈도 3.6%)

[대칭키 암호의 키 배송 문제 해결 방법]

1. 키 사전 공유
  - 키 관리기관(Trusted Authority, TA)이 사전에 사용자들에게 비밀 경로를 통하여 키를 전달
  - 많은 키가 필요. (각 사용자별로 n-1개, TA는 $\frac{n(n-1)}{2}$ 개)

2. 온라인 키 분배
  - 암호 통신이 필요할 때마다 키 배포 센터(Key Distribution Center, KDC = TA)에서 키를 전달
  - KDC에서 통신하는 사용자의 키로 세션키 K를 암호화해서 전달. 각 사용자는 자신의 키로 복호화해서 세션키 K를 얻어 메시지/암호문을 암호화/복호화한다.
  - 암호 통신이 끝난 후, 사용한 세션키 K를 삭제
  - KDC는 n개의 키, 각 사용자들은 자신의 키 1개만 가진다.

3. Diffie-Hellman 키 교환
  - 유한체상의 이산대수 문제(Discrete Logarithm Problem, DLP)를 풀기 어렵다는 사실에 기반
  - 키 교환(계산) 절차 (송신자:A/수신자:B)
    1) A와 B는 매우 큰 소수 p와 q를 선택한다. (p, q : 공개 / x, y : 비밀 / R_1 , R_2 은 노출돼도 상관없음)
    2) A는 임의의 큰 수 x를 $0 \leq x \leq p-1$ 범위 내에서 선택하고, $R_1 = q^x \bmod p$ 를 계산.
    3) B는 임의의 큰 수 y를 $0 \leq y \leq p-1$ 범위 내에서 선택하고, $R_2 = q^y \bmod p$ 를 계산.
    4) A와 B는 서로에게 R_1 , R_2를 전달한다.
    5) A는 $K = (R_2)^x \bmod p$ 를 계산해서 K를 얻는다.
    6) B는 $K = (R_1)^y \bmod p$ 를 계산해서 K를 얻는다.
  - 공격 기법
    1) 이산대수 공격 : R_1 , R_2 를 도청자가 가로채 x, y를 구하게 되면 K를 알아 낼 수 있다.
    2) 중간자 공격(Man-In-The-Middle Attack, MITM) : 인증단계가 없어서 이 공격에 취약.
      → 전자서명과 공개키 인증서를 이용해서 막을 수 있다.
      → 국-대-국(STS, Station-To-Station) 프로토콜 : DH에 기반. 세션키를 만들기 위해 공개키 인증서를 이용한 전자서명을 사용하여 MITM 공격을 방지
    3) DoS 공격 : DH의 계산이 복잡한 것을 이용하여 공격 대상 서버에 IP 스푸핑 등을 통해 위조한 키 생성 요청을 동시에 다수 요청하여 키 생성 시간으로 인해 서버를 마비시키는 공격
      → 키 생성 요청자를 확인하기 위한 쿠키(cookie)를 사용함으로써 방지

4. 공개키 암호 사용으로 해결

[공개키 암호]

- **대칭키 암호** : 평문을 복잡한 형태로 변환하여 기밀성 유지
- **공개키 암호** : 수학적으로 해결하기 곤란한 문제를 기반으로 기밀성 유지
- 대칭키 암호 vs 공개키 암호에서 어떤 것이 더 안전하다고 말할 수 없다.

- 공개키 암호에서는 암호화키와 복호화키가 분리되어 있다.
- 두 개의 키는 서로 수학적 관계이기 때문에 각각 별개로 만들 수 없다.

[RSA(Rivest-Shamir-Adleman) 암호시스템]

- 공개키 암호 알고리즘, 세계적으로 널리 인정되는 표준이다.
- **인수분해 문제(Prime factorization)에 근거.**
- 암호화뿐만 아니라 전자서명의 용도로 사용된다.

- 암호화/복호화
  - e : 공개 / d : 비밀
  - 암호화 : $C = M^e \mod n$
  - 복호화 : $M = C^d \mod n$

- **키 생성 알고리즘** (공개 : e, n / 비밀 : $p, q, d, \Phi(n)$)
  1. 서로 다른 소수 p와 q를 선택
  2. $N = p \times q$ 계산
  3. $\Phi(N) = (p-1)(q-1)$ 계산
  4. $\Phi(N)$ 보다 작고, $\Phi(N)$과 서로소인 정수 e를 찾는다.
  5. $de \equiv 1 \pmod{\Phi(N)}$ 을 만족하는 정수 d를 구한다.

  → 공개키 : (e, n) / 비밀키 : (d, n)

- p, q, e, d 조건
  - p와 q는 거의 같은 크기이고 최소 512bits가 되어야 한다. -> N은 최소 1024bits
  - $p-1$과 $q-1$은 큰 소인수를 갖는다.
  - $p-1$과 $q-1$의 최대공약수는 작은 수이다.
  - d와 n은 거의 같은 크기이다.
- RSA 알고리즘의 안정성은 p와 q를 구해 내는 것에 달려있다.

- 최적 비대칭키 암호 패딩 (OAEP, Optimal Asymmetric Encryption Padding)
  - RSA에서 짧은 메시지는 짧은 메시지 공격에 의해 암호문을 위험에 빠뜨리게 된다.
  - 의미 없는 padding을 메시지에 붙여 공격 작업을 어렵게 한다.

- 공격 기법
  - 수학적 공격(소인수 분해 공격) : 두 개의 소수 곱을 인수분해 하려는 시도
  - 타이밍 공격(시간 공격) : 복호화 알고리즘 실행 시간에 의존
  - 선택 암호문 공격(CCA) : 임의의 데이터 송신 시 그것을 암호문으로 간주하고 회신해주는 것을 이용한 공격. OAEP로 방지할 수 있다.

[Rabin 암호시스템]

- 합성수 모듈러에 관하여 제곱근을 찾기 어렵다는 사실로부터 안정성을 얻는다. (인수분해 문제)
- 암호화 과정은 오직 한 번의 곱셈으로 매우 빨리 수행된다.
- Rabin 암호시스템의 복잡도는 큰 수 n을 두 개의 소수의 곱으로 소인수분해하는 수준의 복잡도와 같다.
- p와 q의 크기가 충분히 크다면 RSA만큼 안전하다.

[ElGamal 암호시스템]

- 이산대수 문제에 근거.
- 암호화과정에서 암호문의 길이는 평문의 약 2배가 됨.
- 때문에, 많은 메모리 공간이 필요, 전송 속도도 느려짐.

[타원 곡선 암호(ECC, Elliptic Curve Cryptosystem)]

- 타원 곡선(Elliptic Curve)이라는 이론에 근거 (타원곡선 군에서의 이산대수 문제)
- **RSA보다 키의 길이를 줄이면서도 동일한 성능**을 제공하여 전자상거래의 핵심 기술로 주목받고 있음
- H/W와 S/W로 구현하기가 용이
- 메모리와 처리능력이 제한된 응용분야(스마트카드, 무선통신 단말기 등)에 효율적

[하이브리드 암호시스템]

- 대칭키 암호의 기밀성 + 공개키 암호로 키 배송 문제 해결
- 공개키 암호 문제점 2가지
  1) 대칭키 암호에 비해 처리속도가 많이 느리다. → 하이브리드 암호시스템
  2) MITM 공격에 약하다. → 공개키에 대한 '인증'이 필요

[인수분해/이산대수 근거 암호시스템 분류]

- 인수분해 문제 : RSA, Rabin
- 이산대수 문제 : Diffie-Hellman, ElGamal, ECC, DSS

05. 해시함수와 응용 (출제빈도 1.6%)

[일방향 해시함수]

• 해시함수 : 임의의 길이 m을 갖는 메시지 M을 입력으로 해서 고정된 길이 n을 갖는 해시 값 또는 해시 코드라 불리는 값을 출력하는 함수

• 충돌이 존재. 해시 값 한 개에 여러 메시지를 가질 수 있다.

• 일방향 해시함수 = 메시지 다이제스트 함수 = 암호학적 해시 함수

• 출력되는 해시 값 = 메시지 다이제스트 = 핑거프린트

• **특징**

- 임의 길이의 메시지로부터 고정 길이의 해시 값을 계산
- 해시 값을 고속으로 계산할 수 있다.
- 일방향성을 갖는다. : 해시 값으로부터 메시지를 구할 수 없다.
- 메시지가 다르면 해시 값도 다르다.

• 충돌을 발견하는 것이 어려운 성질을 **충돌 내성(collision resistance)**이라고 부른다.

• 랜덤 오라클 모델(Random Oracle Model)

- 해시함수에 대한 이상적인 수학적 모델
- 임의의 길이를 갖는 메시지에 오라클은 난수로 생성된 고정된 길이의 메시지 다이제스트를 생성, 제공
- 이미 다이제스트가 존재하는 메시지가 주어지면 오라클은 저장되어 있던 다이제스트 제공
- 새로운 메시지에 대해서는 이미 생성한 다이제스트와는 다른 다이제스트를 생성

• 공격 기법

- 무차별 공격 : 약한 충돌 내성을 깨고자 하는 공격
- 일치블록 연쇄공격 : 사전에 해시 값을 다양하게 생성 후 공격하고자 하는 메시지의 해시 값과 같은 것을 찾는 공격
- 중간자 연쇄공격 : 전체 해시 값이 아닌 해시 중간 결과에 대한 충돌 쌍을 찾는다.
- 고정점 연쇄공격 : 메시지 블록과 연쇄변수 쌍을 얻어 연쇄변수가 발생하는 특정한 점에 임의의 동등한 블록들을 메시지 중간에 삽입해도 전체 해시 값이 변하지 않는다.
- 차분 연쇄공격 : 압축함수의 입출력 차이를 조사하여 0의 충돌쌍을 찾아내는 공격

[해시함수의 보안 요구사항]

• **역상 저항성 (= 약 일방향성)**

- $y = h(M)$을 만족하는 **입력 값 M**을 찾는 것이 매우 힘들어야 된다.
- 안전성, 역함수 계산 방지

• **두 번째 역상 저항성 (= 약한 충돌 내성 = 강 일방향성)**

- $y = h(M) = h(M'), M \neq M'$ 을 만족하는 **다른 입력 값 M'**을 찾는 것이 매우 힘들어야 한다.
- 안전성, 역함수 계산 방지

• **충돌 저항성 (= 강한 충돌 내성 = 충돌 회피성)**

- $y = h(M) = h(M')$을 만족하는 **임의의 두 입력 값 M, M'**을 찾는 것이 매우 힘들어야 한다.
- 안전성, 내부부정 방지
- 충돌 저항성은 두 번째 역상 저항성을 보장

[키가 없는 해시함수]

• 메시지 다이제스트(Message Digest)
  - MD2 → MD4 → MD5
  - MD5는 메시지를 512bits로 된 블록으로 나누고 128bits 다이제스트를 출력한다. 라운드 수는 64(14 * 4R)
  - 128bits 메시지 다이제스트는 충돌 공격에 내성을 갖기에는 길이가 너무 짧다고 알려진다.
  - 내부 구조에 대한 약점이 발견되고, 생일 공격에 노출되었다.
• SHA(Secure Hash Algorithm)
  - MD4 해시함수에 기초해서 만들어짐. MD5보다 느리지만 조금 더 안전.
  - SHA-160 (SHA-1) : 메시지를 512bits 블록으로 나누고 160bits의 다이제스트를 출력한다. 라운드 수는 80(20 * 4R)
  - SHA-512 (SHA-2) : 메시지를 1024bits 블록으로 나누고 512bits의 다이제스트를 출력한다. 메시지의 길이는 2^{128}bits를 넘지 않는다.
• RIPEMD-160 : 메시지를 512bits 블록으로 나누고 160bits의 다이제스트를 출력한다.
• Tiger : 64bits 시스템에서 해시 함수를 수행하기 위해 설계. MD5, SHA-1보다 속도가 빠르다.
• HAVAL : 메시지를 1024bits 블록으로 나누고 128/160/192/224/256bits의 다이제스트를 출력한다.
• HAS160 : 메시지를 512bits 블록으로 나누고 160bits의 다이제스트를 출력. 한국형 전자서명 표준에 사용할 목적으로 개발됨.

• 변경 감지 코드(MDC, Modification Detection Code)
  - 메시지의 무결성을 보장하는 메시지 다이제스트
  - 수신자는 메시지로부터 새로운 MDC를 생성하고, 송신자에게 받은 MDC와 비교하여 해당 메시지가 변경되지 않았다는 것을 보장해준다.

[키를 사용하는 해시함수, MAC]

• 무결성을 확인하고, 메시지에 대한 인증을 하는 기술
• 대칭키 암호 사용. (MAC 알고리즘으로 블록 암호나 해시 함수에 기반을 둠)
• **대칭키 암호를 사용**함으로써, 키 배송 문제가 생긴다.

• 축소 MAC : 해시 과정이 두 단계로 이루어져 있음. (키도 두 번 사용)
• **HMAC** : 앞의 H는 해시함수를 의미하며, SHA-1과 같은 일방향 해시함수를 이용하여 MAC를 구성
• CBC-MAC : 블록 암호 모드인 CBC모드와 유사한 방법
• CMAC : CBC-MAC와 같은 종류의 데이터 인증과 무결성을 제공하지만, 조금 더 안전
• CCM(Counter with CBC-MAC) : CTR 모드와 CBC-MAC을 통합. AES 암호 알고리즘, CTR 모드, CBC-MAC 인증 알고리즘으로 구성
• GCM mode : CTR 모드에 인증 기능을 추가한 모드

• 재전송 공격
  - MAC 값을 도청해서 저장해둔 뒤 저장해둔 MAC 값을 반복해서 송신하는 공격
  - 보안 대책으로 순서 번호, 타임스탬프, 비표(nonce), 시도/응답이 있다.

• **MAC가 해결하지 못하는 문제**
  - 제 3자에 대한 증명 : 두 사람 사이에서는 상대방이 MAC를 계산하였다고 말할 수 있지만, 제 3자에게 MAC 값을 누가 계산했는지 증명할 방법이 없다. → 전자서명으로 증명 가능
  - 부인 방지 → 역시 전자서명으로 부인 방지가 가능해진다.

06. 전자서명과 PKI (출제빈도 4.0% / 매회 2~4문제 출제)

[전자서명 서비스]

- 전자서명의 형식
  - 공개키 암호방식을 이용한 공개키 서명 방식 : 누구나 검증 가능하여, 서명 생성 및 검증이 간편
  - 관용 암호방식을 이용한 중재 서명 방식 : 서명 생성과 검증을 제3자가 중재. 서명할 때마다 제3자의 참여가 있어야 함.

- **메시지 인증** : 수신자 B는 받은 메시지가 송신자 A로부터 왔다는 것을 알 수 있다.
- **메시지 무결성** : 메시지가 변경되면 서명도 변경된다. (해시함수, 공개키)
- **부인방지** : 신뢰받는 제 3자로부터 검증받을 수 있다.
- 공개키 알고리즘을 사용하면 기밀성을 보장할 수 있다.
- 암호화 시스템에서는 수신자의 개인키와 공개키가 사용되고, 전자서명에서는 송신자의 개인키와 공개키가 사용된다.

- 전자서명의 주요기능 : 위조 불가, 서명자 인증, 부인방지, 변경 불가, 재사용 불가
  - 위조 불가 : 합법적인 서명자만이 전자서명을 생성할 수 있어야 한다.
  - 서명자 인증 : 전자서명의 서명자를 누구든지 검증할 수 있어야 한다.
  - 부인방지 : 서명자는 서명행위 이후에 서명한 사실을 부인할 수 없어야 한다.
  - 변경 불가 : 서명한 문서의 내용을 변경할 수 없어야 한다.
  - 재사용 불가 : 전자문서의 서명을 다른 전자문서의 서명으로 사용할 수 없어야 한다.

[전자서명 구조]

- RSA 전자서명 구조
  - 전자서명 구조에서는 개인키와 공개키의 역할이 바뀐다.
  - 암호화에서는 수신자의 키를 이용했지만, 전자서명에서는 송신자의 개인키와 공개키를 이용한다.
- ElGamal 전자서명 구조 : ElGamal 암호시스템과 동일한 키를 사용하지만 알고리즘은 다르다.
- Schnorr 전자서명 구조 : ElGamal 기반이지만 서명의 크기를 줄인 새로운 구조
- DSS(Digital Signature Standard)
  - ElGamal 전자서명을 개량한 방식. (이산대수 문제)
  - 오직 전자서명 기능만을 제공하도록 설계되었고 서명과 검증에 소요되는 계산량을 줄임.
- 타원곡선 전자서명 구조(ECDSA, Elliptic Curve DSA) : 짧은 비트 길이로 인해 짧은 처리 시간에 짧은 서명 생성이 가능

[전자서명 방식]

- 복원형 전자서명
  - 기존 공개키 암호방식을 이용하여 별도의 전자서명 프로토콜이 필요 없음.
  - 메시지를 일정 크기 블록으로 나누어, 그 각각의 블록에 서명을 해야 하므로 시간 소요↑
  - 실제로는 사용되지 않는다.
- **부가형 전자서명**
  - 메시지를 해시하여 나온 다이제스트에 한 번의 서명 생성 과정만이 필요하므로 효율적이다.
  - 전송량이 조금 늘어나지만 실제로 많이 사용되는 방법이다.

[특수 전자서명]

• 공개키 방식을 이용한 전자 서명은 검증하는 키가 공개되어 있어서 서명의 검증을 누구나 할 수 있는 문제점이 있다.

• 부인방지 전자서명 : 자체 인증 방식을 배제시키고 서명 검증할 때, 서명자의 도움이 있어야 검증이 가능한 방식
• 의뢰 부인방지 서명 : 신뢰하는 제 3자가 서명을 검증해주는 방식
• 수신자 지정 서명 : 지정한 검증자만이 서명을 검증할 수 있고, 필요시 제 3자에게 그 서명이 서명자에 의해 자신에게 발행된 서명임을 증명할 수 있게 하는 방식
• 위임 서명 : 부재 중 자신을 대리해서 서명을 할 수 있는 방식.
• **은닉 서명(Blind Digital Signature)** : 서명문의 내용을 숨기는 방식. 익명성을 유지할 수 있음.
• 다중 서명 : 동일 전자문서에 여러 명이 서명하는 방식

[전자투표 시스템]

• 요구사항
- 완전성 : 투표 결과의 정확한 집계
- 익명성 : 투표 결과로부터 투표자 구별 불가
- 건전성(강건성) : 부정 투표자에 의한 선거 방해가 없어야 한다.
- 이중투표방지(재사용 불가) : 정당한 투표자는 단 1회만 투표 허용
- 정당성 : 투표에 영향을 미치는 것이 없어야 한다.
- 적임성 : 투표권 없는 자의 투표 행위 금지
- 검증 가능 : 투표 결과를 누구나 확인하여 검증해볼 수 있다.(투표 결과 위조 불가능)

• 방식 (PSEV - 키오스크 - REV)
- PSEV 방식 : 기존 선거 방식 + 전자
- 키오스크(kiosk) 방식 : PSEV 방식과 유사하지만 공공장소에 설치, 관리자 없음
- REV 방식 : 어디서든 투표 가능. 비밀투표 침해가능성 높음

[전자입찰 시스템]

• 요구사항
- 독립성 : 각 구성요소는 독자적인 자율성을 보장받아야 함
- 비밀성 : 네트워크상의 개별 정보는 누구에게도 노출되지 않아야 함
- 무결성 : 입찰 시 입찰자 자신의 정보를 확인 가능하게 하여, 누락 및 변조 여부 확인이 가능해야 함
- 공평성 : 입찰이 수행될 때 모든 정보는 공개되어야 함
- 안전성 : 각 입찰 참여자 간의 공모 방지

[전자서명으로 해결할 수 없는 문제]

• 서명 검증을 할 때 이용하는 공개키가 진짜 송신자의 공개키가 맞는지의 문제를 해결하기 위해 공개키 인증서가 고안되었다.
• 공개키 인증서는 공개키를 메시지로 간주하고 신뢰 가능한 제 3자에게 전자서명을 해서 받은 공개키이다. 이 기반 구조를 PKI(Public Key Infrastructure, 공개키 기반 구조)라고 한다.

[PKI 개념]

• 공개키 알고리즘을 위한 키 관리 구조

• 제공 서비스 : **기밀성, 무결성, 인증, 부인방지, 접근제어**

• 일방향 해시함수 → MAC → 전자서명 → PKI 의 흐름으로 이어진다.

[PKI 구성요소]

• 크게 인증기관, 검증기관, 등록기관, 저장소, 사용자로 구성되어 있다.

• 인증기관(CA, Certification Authority) : 인증정책 수립, 인증서 관리, 다른 CA와의 상호 인증

1) 정책승인기관(PAA, Policy Approving Authority)

- 루트 인증기관으로, PKI 전반에 사용되는 정책을 생성, 수립
- 하위 기관들의 정책 감사

2) 정책인증기관(PCA, Policy Certification Authority)

- PAA 하위 계층, 하위 CA, 사용자들의 정책 수립
- CA의 공개키를 인증하고 인증서, 인증서 폐지 목록 등을 관리

3) 인증기관(CA)

- PCA 하위 계층.
- 사용자/등록기관의 요청에 공개키 인증서를 발행·폐지
- 사용자에게 자신의 공개키와 상위 기관의 공개키를 전달

• 검증기관(VA, Validation Authority)

- 인증서의 유효성 여부 / 관련 거래의 유효성 여부 등을 확인시켜줌.
- 적절한 인증서 검증 기능이 없다면, 보안성이 떨어진다.
- CA에서 직접 운영가능.

• 등록기관(RA, Registration Authority)

- 사용자와 CA가 원거리에 있는 경우, 사용자와 CA 사이에 RA를 두어 사용자의 인증서 신청 시 CA 대신 사용자의 신분·소속 등을 확인
- 선택적 요소. RA가 없으면 CA가 RA의 기능을 대신 수행할 수 있다.

[PKI 형태]

• 계층 구조

- 최상위에 루트 CA가 존재하고, 그 밑으로 하위 CA가 계층적으로 존재하는 트리 구조.
- 상위 CA가 하위 CA에 인증서를 발행하고, 하위 CA는 상위 CA의 인증정책에 영향을 받는다.
- 루트 CA 간에 상호인증은 허용하지만, 하위 CA 간에 상호인증은 불가능.
- 정부 같은 관료조직에 적합, 협동업무 관계에는 부적합
- 인증경로 탐색이 용이하고, 모든 사용자가 최상위 CA의 공개키를 알고 있어서 검증이 용이
- 최상위 CA에 집중되는 오버헤드가 발생
- 최상위 CA의 비밀키 노출이 되면 피해가 막대하다.

• 네트워크 구조
  - 각각의 CA들이 자신의 인증정책에 따라 독립적으로 존재하는 형태
  - 모든 상호인증이 허용되면 상호인증의 수가 대폭 증가
  - 유연하고 인증경로 단순하여 실질적인 업무관계에 적합.
  - CA의 비밀키가 노출이 되도 피해가 적다.
  - 인증경로 탐색이 복잡하고, 정책 수립과 적용이 어려움

[인증서 표준 규격 X.509]

• 공개키 인증서 (PKC, Public Key Certificate)
  - 해당키가 특정인의 것이라는 것을 보증해주는 것
  - 개인정보, 소유자의 공개키가 들어있고, CA의 개인키로 전자서명 되어있다.

• X.509 v3 인증서 프로파일

요소	구분	설명
버전 (Version)	**필수**	X.509의 버전
일렬번호 (Serial Number)	**필수**	CA에 의해 인증서에 부여되는 유일한 번호. (발행자이름과 일렬번호로 인증서를 **유일**하게 구분할 수 있어야 한다.)
서명 알고리즘 식별자 (Signature Algorithm ID)	**필수**	CA가 인증서를 서명하기 위한 알고리즘과 알고리즘 식별자를 포함. 이 정보는 끝 부분 서명 필드에도 포함되어 있다.
발행자 이름 (Issuer name)	**필수**	인증서 발행자(보통 CA)의 이름
유효기간 (Validity period)	**필수**	인증서 유효기간의 시작날짜와 종료날짜
주체 이름 (Subject name)	**필수**	사용자(피발급자)의 이름 상위 CA가 하위 CA에게 인증서를 발행하는 경우 이 필드에는 CA의 이름이 된다.
주체의 공개키 정보 (Subject Public Key)	**필수**	사용자의 공개키, 관련 알고리즘
발행자 유일 식별자 (Issuer Unique ID)	선택 (v2)	발행자나 사용자의 이름이 중복되는 경우 이를 구별하기 위한 수단
주체 유일 식별자 (Subject Unique ID)	선택 (v2)	주체를 유일하게 구별하는 데 사용
확장 (Extensions)	선택 (v3)	발행자가 인증서에 추가적으로 사적인 정보를 넣을 수 있는 필드
서명 (Signature)	**필수**	인증서에 대한 서명 값이 들어있는 필드 나머지 전체 필드를 보호하는 역할을 한다. 다른 필드 전체를 해시해서 나온 해시 값을 CA의 개인키로 암호화한 값이 들어간다.

• X.509의 확장영역에는 키와 정책 정보, 사용자와 발행자 속성, 인증 경로 제약조건이 들어간다.

• X.509 인증서 폐지 목록
  - 인증서 폐지 목록(CRL, Certificate Revocation List)은 CA의 저장소 또는 디렉터리에 저장되어 신뢰 사용자가 언제든지 이 목록을 검색할 수 있어야 한다.
  - CRL 내의 폐기된 인증서들은 인증서 일렬번호에 의해서 확인할 수 있다.
  - 인증서 폐지 사유
    (1) 사용자의 개인키가 노출되었거나 훼손된 것으로 판단되는 경우
    (2) CA가 사용자를 더 이상 인증해줄 수 없을 경우
    (3) CA의 개인키가 노출되었거나 훼손된 것으로 판단되는 경우

• 온라인 인증서 상태 검증 프로토콜(**OCSP,** Online Certificate Status Protocol)
  - 실시간으로 인증서 상태를 확인할 수 있는 프로토콜
  - 백그라운드에서 자동으로 수행하며, CA에 의해 관리되고 있는 CRL을 검사
  - OCSP 클라이언트, OCSP 서버, 인증 서버로 구성

• 인증서 관리 프로토콜(CMP, Certificate Management Protocol) : PKI 환경에서 인증서 관리 서비스를 제공하기 위한 PKI 실체들(이용자, CA, RA 등) 간의 통신 프로토콜

07. 키, 난수 (출제빈도 0.3%)

• 세션키(Session key) : 통신 때마다 한 번만 사용되는 키
• C 마스터키(Master key) : 반복적으로 사용되는 키
• CEK(Contents Encrypting Key) : 사용자가 이용하는 콘텐츠를 암호화하는 키
• KEK(Key Encrypting Key) : 키를 암호화하는 키

• 솔트(Salt)
  - 키(KEK)를 만들 때 패스워드와 함께 일방향 해시함수에 입력된다.
  - 사전공격을 막기 위해 존재
  - KEK를 만들 때 솔트를 사용하면 KEK 후보가 솔트의 비트 길이만큼 늘어난다.

• 난수의 성질 : 무작위성 - 예측 불가능성 - 재현 불가능성
• 의사난수 생성기 : 난수(의사난수)를 생성하는 S/W

PART 03. 접근통제

08. 접근통제 개요 (출제빈도 0.7%)

- 접근제어 : 허가되지 않은 자원의 사용과 허가되지 않은 방법을 통한 자원 사용을 제어하는 것
- 주체(행위자), 접근(주체의 활동), 객체(제공자)
- 접근 통제 절차 3+1가지
  1) 식별 : 본인이 누구라는 것을 시스템에 밝히는 행위(ID, 계정번호, 메모리카드)
  2) 인증 : 주체의 신원을 검증하기 위한 증명(패스워드, 스마트카드, 생채인증)
  3) 인가 : 인증된 주체에게 접근을 허용하고 특정 업무를 수행할 권리를 부여(접근제어목록, 보안등급)
  4) 책임추적성 : 시스템에 인가된 주체가 시스템에 어떤 행위를 하고 있는지 기록
- 기본 원칙 : 직무 분리, 최소 권한 등

09. 사용자 인증 (출제빈도 4.8% / 매회 4~5문제 출제)

[지식 기반 인증(What you know)]

장점	다양한 분야에서 사용
	검증이 확실
	비용이 저렴함
	사용 편리
단점	소유자의 분실 가능성
	공격자의 추측 가능
	사회 공학적 공격에 취약
	사용자의 관리 부주의로 인한 노출

1) 패스워드(Password)

- 고정된 패스워드
  - 방법1 : ID/PW를 DB에 그대로 저장
  - 방법2 : 패스워드를 해시한 값을 저장
  - 방법3 : 패스워드에 솔트(Salt)를 붙여 해시한 값을 저장

- 일회용 패스워드(OTP, One-Time Password)
  - 오직 한번만 사용되는 패스워드이기 때문에 도청이나 도난이 무의미해짐
  - 방법1 : 사용자와 시스템이 PW 목록에 대해 합의하고, 그 패스워드들은 오직 한번만 사용한다.
  - 방법2 : 사용자와 시스템이 PW를 순차적으로 업데이트하기로 합의한다.
  - 방법3 : 사용자와 시스템이 해시함수를 이용하여 순차적으로 업데이트하기로 합의한다.

- 패스워드의 안정성과 문제점
  - 패스워드의 길이가 길어질수록, 패스워드의 사용기간이 짧을수록, 사용빈도가 낮을수록 패스워드를 추측하기 어렵다.
  - 크래킹 툴(NTCrack, John the ripper)과 같은 소프트웨어로 크랙하기 쉽다.

2) 시도-응답(Challenge-Response) 개인 식별 프로토콜

• 도청이 가능한 환경에서 안전

• 대칭형 암호와 공개키 암호에 기반

• 어떤 실체가 자신의 신분을 다른 실체에게 증명하기 위하여 자기 자신만이 소유하고 있는 어떤 비밀정보를 자신이 알고 있다는 사실을 간접적으로 보여주는 프로토콜

• 일방향 개인 식별 프로토콜 : 서버 또는 클라이언트 중에 어느 한 대상을 식별하는 프로토콜

• 상호 개인 식별 프로토콜 : 상호간에 대상을 식별하는 프로토콜

3) 영지식(zero knowledge) 개인 식별 프로토콜

• 시도-응답 개인 식별 프로토콜이나 패스워드를 이용한 개인 식별은 클라이언트와 서버가 비밀 정보를 동시에 공유하고 있어야 하기 때문에, 클라이언트가 서버를 신뢰하지 못하는 상황에서 유용

• 자신의 비밀정보를 제공하지 않고 자신의 정당한 신분만을 밝힘으로써 식별되는 유형의 프로토콜

4) I-PIN(Internet Personal Identification Number)

• 주민번호 대신에 ID와 PW로 대체하는 수단

• 이용효과

- 주민번호 유출예방
- 본인확인 강화

[소유 기반 인증(What you have)]

장점	일반적임
	입증된 기술(신용카드)
	생체 인식 방식보다 경제적
단점	물리적 파손
	소유물이 없을 경우 인증이 어려움
	복제 가능
	사회 공학적 공격
	자산 관리 기능 요구

1) 메모리 카드(토큰) : 정보를 저장할 수 있지만 정보를 처리할 수 없다.

2) 스마트카드 : 정보를 저장/처리할 수 있다.

3) 일회용 패스워드(OTP, One-Time Password)

• OTP용 프로그램에서 사용자 패스워드와 OTP 생성용 입력 값을 입력하면 암호 알고리즘을 사용해서 OTP를 생성하는 사용자 인증 방법.

• OTP는 단말장치로 주로 구현돼서 소유기반으로 분류한다.

• 사회 공학적 공격에 대처 가능하며 일정 시간마다 패스워드를 변경한다.

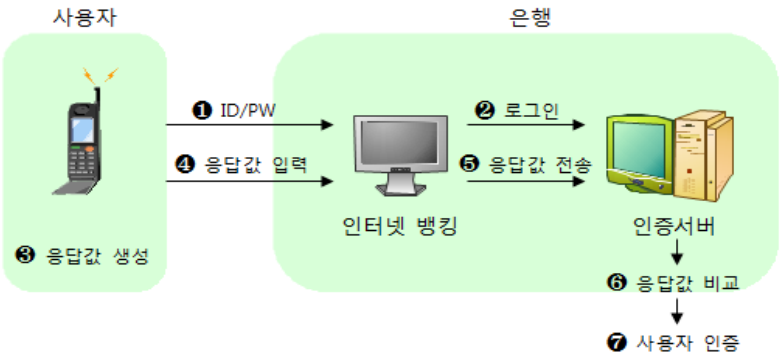
• 휴대폰을 통한 인증으로 사용자의 편리성 및 안정성을 확보

• 질의응답 방식

- 인증서버에서 질의 값을 전송하고 사용자가 질의 값을 입력해 응답 값을 생성해 응답 값을 전송하는 방식
- 구조가 간단하며 OTP 생성 매체와 인증서버 간 동기화가 필요 없다.
- 사용자가 질의 값을 직접 입력해야 하는 번거로움이 있고, 인증 서버에서는 같은 질의 값이 생성되지 않도록 관리해야 한다.

• 시간과 이벤트 동기화 방식

- 토큰장치와 인증서비스는 반드시 동일한 비밀키를 공유하여 암·복호화에 사용한다.
- 시간 동기화 방식 : 토큰장치와 비밀키에 나타나는 시간 값은 OTP를 생성하는데 사용
- 이벤트 동기화 방식 : 사용자가 토큰장치의 버튼을 누르면 다음 인증 값이 나타남.
- 이벤트 동기화 방식은 시간 동기화 방식에 비해 동기화되는 기준 값을 수동으로 조작할 필요가 적어 사용이 간편함.
- 질의응답 방식에 비해 호환성이 높음
- OTP 생성 매체와 인증서버의 시간 정보/계수기 값이 동기화되어 있어야 함.
- 동작 방식



• S/KEY 방식

- 유닉스 계열 OS에서 인증에 사용되는 해시 체인 기반의 방식
- 클라이언트에서 정한 임의의 비밀키를 서버에서 받아 이 비밀키를 첫 번째 값으로 해시 체인 방식으로 이전 결과값에 대한 해시 값을 구하는 작업을 n번 수행한 뒤, 이 n개의 OTP를 서버에 저장
- 전용 장치(H/W)가 필요 없이 S/W로 쉽게 구현될 수 있지만, 사용자마다 관련 S/W가 필요하고 정해진 횟수마다 시스템 재설정 필요

[개체(생물학적) 특성 기반 인증(What you are)]

장점	사용하기 쉬움
	분실, 손실, 도난 될 수 없다
	위조가 어렵다
	대여 불가능
단점	잘못 판단할 가능성 존재
	관리가 어려움
	인증을 위한 임계치 설정이 어렵다

• 생체인증 기술 평가항목

- 보편성, 유일성, 지속성, 획득성, 성능, 수용성, 반기만성

• 생체인증의 정확도
- FRR(False Rejection Rate, 오거부율), FAR(False Acceptance Rate, 오인식률)
- FRR : 인식되어야 할 사람이 얼마나 시스템에 의해서 인식이 되지 않았는지에 대한 값
- FAR : 인식되어서는 안 될 사람이 얼마나 시스템에 의해서 인식이 되는지에 대한 값
- 보안성 강화 : FRR↑ / FAR↓
- 사용자의 편의성 : FAR↑ / FRR↓
- CER(Crossover Error Rate)/EER(Equal Error Rate) : FRR과 FAR이 일치하는 지점으로, 수치가 낮을수록 정확하다.

[통합인증체계(SSO)]

• SSO(Single Sign On) : 한 번의 시스템 인증을 통하여 접근하고자 하는 다양한 시스템에 재인증 절차 없이 접근할 수 있도록 하는 통합 로그인 솔루션이다.
- 패스워드 분실/망각 가능성 감소
- 관리의 간편함, 보안수준의 향상

장점	사용자 편의성 증가, 보안성 강화
	중앙 집중 관리를 통한 효율적 관리(운영비용 감소)
단점	SSO서버 침해 시 모든 서버의 보안 침해 가능 (단일 실패 지점)
	자원별 권한관리 미비

• 엑스트라넷 접근 관리(EAM, Extranet Access Management) : 인트라넷, 엑스트라넷 및 일반 클라이언트/서버 환경에서 자원의 접근 인증과 이를 기반으로 자원에 대한 접근 권한을 부여/관리하는 통합 인증 관리 솔루션
• 식별/접근 관리(IAM, Identity and Access Management) : ID와 패스워드를 종합적으로 관리해주는 역할 기반의 계정 관리 솔루션. EAM을 확장/보완한 솔루션

[커버로스(Kerberos)]

• **인증 프로토콜**이자 **KDC(키 배포 센터)**이다.
• 개방된 네트워크 내에서 서비스 요구를 인증하기 위해 **대칭키 암호기법**에 바탕을 둔 티켓 기반 인증 프로토콜.
• 분산 환경을 위한 SSO이며, 기업 접근 통제를 위한 확장성, 투명성, 안정성, 보안을 제공

• 커버로스 구성요소
- KDC : 커버로스의 핵심, 키 분배 서버. 사용자의 패스워드는 비밀키로 변환된다.
- AS : 실질적으로 인증을 수행하는 KDC의 부분 서비스
- TGS : 티켓을 부여/분배하는 KDC의 부분 서비스
- 티켓 : 사용자에 대해 신원과 인증을 확인하는 토큰

• 커버로스 구성
1. 커버로스는 모든 사용자의 패스워드를 알고 있고, 중앙집중식 DB에 그 패스워드를 저장하고 있는 인증서버(AS)를 이용
2. AS는 각 서버와 유일한 비밀키를 공유
3. TGS는 AS에게 인증 받은 사용자에게 티켓을 발행
4. 사용자가 새 서비스를 요청할 때마다 자신을 인증하는 티켓을 이용하여 TGS에 접속, TGS가 해당 서비스에 대한 티켓 발행
5. 사용자는 서비스 승인 티켓을 보관하고 필요할 때마다 티켓을 사용하여 서버에 인증시킨다.

• 커버로스의 취약성
- KDC는 단일 실패 지점이 될 수 있으므로, 이중화 구성이 필요
- 비밀키는 사용자의 워크 스테이션에 임시로 저장되어, 공격자에게 탈취될 수 있다.
- 패스워드 추측 공격에 취약

• 커버로스 버전4 vs 버전5
- 버전4에서는 DES 알고리즘을 사용해야 했지만 버전5에서는 모든 대칭키 알고리즘을 사용할 수 있다.
- 버전4에서는 인터넷 프로토콜 주소를 사용해야 했지만 버전5에서는 어떤 유형의 네트워크 주소를 사용할 수 있다.
- 버전4에서는 티켓 유효기간의 한계가 있었지만 버전5에서는 유효기간의 한계가 없어 수명이 더 길고 갱신도 가능하다.

[세사미(SESAME)]
- 커버로스의 기능을 확장하고 약점을 보완하기 위해 개발된 SSO 기술
- 커버로스는 대칭키 기반이지만 세사미는 비대칭 및 대칭키 암호화 기술에 기반한다.
- 커버로스에 티켓이 있으면, 세사미에는 PAC(Privileged Attribute Certificate)가 있다.

10. 접근통제 보안 모델 (출제빈도 2.4%)

[강제적 접근통제(MAC, Mandatory Access Control)]
• **각 주체와 객체의 보안등급을 비교**하는 것에 기반
• 관리자만이 접근제어의 규칙을 설정, 변경할 수 있다. (**중앙집중형** 보안관리)
• 사용자의 의도와는 관계없이 의무적으로 접근을 제어한다.
• 모든 MAC 모델은 벨라파듈라(BLP) 모델을 근간으로 하고 있다.
• 구성 : 주체 등급(사용자, 주체의 보안레벨) / 객체 등급(데이터, 객체의 보안레벨)

장점	매우 엄격한 보안, 중앙집중식 관리
	모든 객체에 대한 관리가 용이
단점	구현, 운영이 복잡
	상업적인 환경에 부적합

• 적용사례 : 방화벽

[임의적 접근통제(DAC, Discretionary Access Control)]
• 접근하고자 하는 **주체의 신분에 따라 접근권한을 부여**하는 방법
• **객체의 소유자가 접근여부를 결정**한다. (분산형 보안관리)
• 하나의 주체마다 객체에 대한 접근 권한을 부여해야 함.

장점	구현이 쉽다.
	권한 변경이 유연함.
단점	데이터(객체)의 의미에 대해 지식이 없음
	신분(ID) 도용 시 통제 방법이 없음
	트로이 목마 공격에 취약

• 적용사례 : ACL

• 접근제어 행렬(Access Control Matrix)
  - 주체를 행, 객체를 열로 구성하고 해당 셀에 주체가 객체에 수행할 수 있는 접근 권한을 기록하여 관리
  - 효과적인 권한 부여 정책을 정의 가능하지만 주체와 객체의 수가 많아질 경우 행렬의 크기가 커져 관리가 어렵다.
• 자격 목록(Capability List, Capability Tickets, Capability Table)
  - 주체의 관점에서 객체에게 권한을 부여
  - 콘텐츠의 보안성이 보장받지 못하는 분산환경에서 사용
  - 커버로스가 자격목록의 한 예이다.
• 접근제어 목록(ACLs, Access Control Lists)
  - 객체의 관점에서 주체에게 권한을 부여

[역할기반 접근통제(RBAC, Role-Based Access Control)]

• **주체와 객체 사이에 역할을 두어 역할에 따라 접근통제**
• MAC와 DAC의 단점을 보완한 기법이다. (Non-DAC)
• 주체, 객체에 접근권한을 할당하는 것이 아닌 역할에 접근권한을 할당한다.
• 정책을 자주 변동하지 않고도 정책관리를 용이하게 할 수 있다.
• 장점
  - 관리자에게 편리한 관리능력 제공
  - DAC에 비해 유연성은 떨어지나, 일관성 있는 접근제어가 용이
  - 최소권한의 원칙, 직무분리의 원칙, 데이터의 추상화

[벨라파둘라 모델(BLP, Bell-LaPadula Confidentiality Model)]

• 기밀성을 강조한 MAC 모델
• 보안 규칙

	읽기	쓰기	읽기/쓰기
상급 보안 계층	X	O	X
할당된 보안 계층	O	O	O
하급 보안 계층	O	X	X

  - 단순 보안 속성(simple security property, ss-property) : 상향 읽기 X / 하향 읽기 O
  - 성형(*) 보안 속성(*-property) : 상향 쓰기 O / 하향 쓰기 X
  - 특수 성형(*) 속성 : 동일 레벨에서만 읽기/쓰기 가능

• 보안 단계가 높은 정보를 훔치는 트로이 목마 공격이 불가능하다.
• 문제점
  - 지나치게 기밀성에 치중하여 **무결성·가용성은 고려하지 않음**
  - **은닉 채널**을 다루지 않음 (→ 기밀성이 훼손될 가능성이 있음)

[비바 무결성 모델(Biba Integrity Model)]

• BLP를 보완한 무결성 모델(MAC)
• 무결성의 목표 중 비인가자에 의한 부적절한 변조방지만을 목적으로 한다.
• 보안 규칙

	읽기	쓰기	서비스 요청
상급 보안 계층	O	X	X
할당된 보안 계층	O	O	O
하급 보안 계층	X	O	O

- 단순 무결성 속성 : 상향 읽기 O / 하향 읽기 X
- 성형(*) 무결성 속성 : 상향 쓰기 X / 하향 쓰기 O
- 호출 속성 : 상향 호출 X / 하향 호출 O

[클락-윌슨 무결성 모델(Clark-Wilson Integrity Model)]

• 조금 더 정교하고 실제적인 무결성 모델
• 무결성의 3가지 목표(비인가자/허가된 사용자에 의한 부적절한 변조방지, 내부/외부 일치성 유지)를 제공
• 상업용으로 설계됨.
• 사용자는 프로그램을 통새서만 객체에 접근할 수 있다.

[만리장성 모델(Chinese Wall Model)]

• 정보 흐름 모델을 기반으로 주체와 객체 사이에서 이해 충돌을 야기하는 방식으로 정보가 흐르지 않도록 함
• MAC, DAC 모델

11. 접근통제 보안위협 및 대응책 (출제빈도 0.3%)

[패스워드 크래커]

• 사전 공격(Dictionary Attack) : 패스워드 사전 파일을 이용하여 접속 계정을 알아내는 해킹 방법
• 무차별 공격(Brute-force Attack) : 성공할 때까지 가능한 모든 조합의 경우의 수를 시도해 공격하는 방법
• 레인보우 테이블을 이용한 공격 : 일정 수의 패스워드와 해시(hash)로 이루어진 체인(Chain)을을 무수히 만들어 놓은 테이블

[ICT 기반 사회공학 공격 기법]

• 사회공학(Social Engineering) : 신뢰할 수 있는 개인이나 조직을 사칭하여 공격대상의 민감한 정보를 빼내는 공격 기법
• 피싱(Phishing) : 공격 대상에게 E-mail 을 발송하여 Phisher 들이 운영하는 위조된 사이트로 이동시킨 후, 개인 정보를 요구하는 공격 기법
• 파밍(Pharming) : 정상적인 사이트로의 접속 요청을 위조 사이트로 방향을 바꾸어 개인정보를 탈취하는 기법
• 스미싱(SMishing) : SMS+Phishing. SMS 을 통해 사용자를 속여 악성 소프트웨어 설치를 유도하고, 해당 소프트웨어를 통해 개인정보를 탈취하거나 금전을 요구하는 기법

PART 04. 시스템 보안

12. 운영체제 개요 (출제빈도 1.3% / 최근 6회 출제문제 없음)

[운영체제 개념]

- 컴퓨터 시스템의 각종 자원을 효율적으로 관리하고 운영하게 하여 사용자에게 편의성 제공
- 목적
  - 처리량의 향상
  - 반환시간의 최소화
  - 응답 시간의 최소화
  - 사용가능도 향상
  - 신뢰도 향상

[운영체제의 이중연산 모드]

- OS의 적절한 동작을 보장하기 위해 OS 코드의 실행과 사용자 정의 코드의 실행을 구분할 수 있어야 한다.
- 사용자 모드 / 커널 모드(슈퍼바이저 모드, 특권 모드)
- 사용자 모드에서는 제한적인 명령어만 사용 가능하고, 관리자 모드에서는 모든 명령어 사용 가능하다.
- 사용자 Application은 시스템 호출(System Call)을 통해 OS로부터 서비스를 요청한다.

[운영체제 구조]

- 커널(Kernel) : OS의 핵심부분으로 주기억장치에 상주하며 프로세스와 파일을 관리

- 모놀리식(monolithic) 구조
  - 시스템 관리에 필요한 모든 기능이 커널 레벨에서 동작하는 방식
  - 시스템 호출 인터페이스나 커널 안에서 통신하는 경우 오버헤드가 거의 없다.
  - 구현하기 어렵고, 유지보수가 힘듦. (커널 코드 전체를 재컴파일해야 한다.)

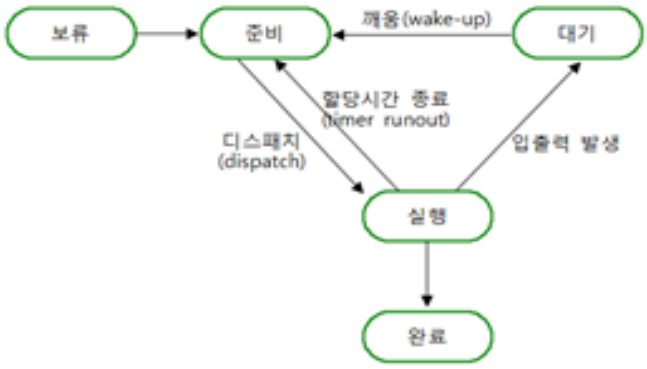
- 마이크로 커널(Microkernels) 구조
  - 가장 기본이 되는 서비스만을 커널에 포함시켜 다른 기능들은 유저 레벨에서 각각의 프로세스로 구현하는 방식
  - 소스코드의 크기가 작고, 기능 수정 필요시 해당 기능이 들어있는 프로세스만 재컴파일하면 된다.

- 계층적 접근(Layered Approach)
  - 계층 1 : 프로세서 관리
  - 계층 2 : 기억장치(메모리) 관리
  - 계층 3 : 프로세스 관리
  - 계층 4 : 주변장치 관리
  - 계층 5 : 파일과 데이터 관리

13. 운영체제 주요 구성기술 (출제빈도 1.8%)

[프로세스(Process)]

- 프로세스 : 실행 중인 프로그램. 프로세스 제어 블록(PCB, Process Control Block)을 가진 프로그램
- 스레드(Thread) : 프로세스 하위 개념이며, 각각의 스레드는 독립적인 제어흐름을 가지고 자신만의 스택과 레지스터를 가진다.
- PCB(Process Control Block)
  - 프로세스를 관리하는데 필요한 모든 정보를 유지하는 자료구조 테이블
  - 포함된 정보 : 프로세스 상태, 프로그램 카운터, CPU 레지스터, CPU 스케줄링 정보, 메모리 관리 정보, 회계 정보, I/) 상태 정보
- 상태전이도



[CPU 스케줄링]

1) FCFS(First Come First Service)
  - CPU를 먼저 요청한 프로세스가 CPU를 먼저 할당 받는 방식 (비선점)

2) SJF(Shortest Job First)
  - 작업수행시간(버스트시간)이 짧은 작업을 우선적으로 처리
  - 비선점 방식이며, 버스트시간이 긴 작업은 시간이 많이 걸림

3) HRN(Highest Response ratio Nest)
  - 대기시간이 긴 프로세스의 우선순위를 높여 기아상태를 해결
  - 비선점 방식

4) RR(Round Robin)
  - FCFS 방식에 시간 할당량 개념을 도입한 방식
  - 수행중인 프로세스가 할당된 시간을 초과하면 다음 프로세스가 선점하여 작업을 수행한다.
  - 시간할당량↑↑ - FCFS 방식과 동일해질 수 있다.
  - 시간할당량↓↓ - 많은 문맥교환으로 오버헤드 증가

5) SRT(Shortest Remaining Time)
  - SJF 방식과 동일하게 준비 큐에 있는 프로세스들 중 가장 짧은 버스트시간의 프로세스를 먼저 수행한다.
  - 선점 방식

6) 다단계 큐(MLQ, Multi-Level Queue)
- 서로 다른 작업을 각각의 큐에서 처리. (선점)
- 각각의 큐는 다른 스케줄링 알고리즘을 가질 수 있다.

7) 다단계 피드백 큐(MFQ, Multi-level Feedback Queue)
- 프로세스가 들어오면 높은 우선순위를 할당해 단계 1에서 수행하고, 시간할당량을 초과하면 다음 낮은 우선순위의 큐에서 수행. (선점)
- 가장 낮은 우선순위의 큐에서는 RR 방식으로 작업 처리
- CPU와 I/O장치의 효율을 높일 수 있다.

[교착상태(Deadlock)]

• 둘 이상의 서로 다른 프로세스가 자신이 요구한 자원을 할당 받아 점유하고 있으면서 상호간에 상대방 프로세스에 할당되어 있는 자원을 요구하는 경우에 발생

• 교착상태 발생하기 위한 4가지 조건
- 상호배제 : 프로세스가 자원을 배타적으로 점유
- 점유와 대기 : 프로세스는 하나의 자원을 점유하고 있으면서, 다른 자원을 할당 받기 위해 대기
- 비선점
- 환형 대기 : 프로세스와 자원의 할당/요청 관계가 원형을 이룸

[메모리 관리]

• 반입(Fetch) 정책 : 주기억장치에 적재할 다음 프로그램이나 자료를 언제 가져올 것인가를 결정하는 문제
- 요구 반입(Demand fetch) 정책 : 어떤 프로그램이나 자료가 참조되는 시점에 주기억장치에 적재
- 예상 반입(Anticipatory fetch) 정책 : 앞으로 요구될 가능성이 큰 프로그램이나 자료를 미리 주기억장치에 적재

• 배치(Placement) 정책 : 새로 반입된 자료나 프로그램을 주기억장치의 어디에 위치시킬 것인가를 결정하는 문제
- 최초 적합 : 첫 번째 사용 가능한 가용 공간을 할당
- 최적 적합 : 사용 가능한 공간들 중에서 가장 작은 것을 할당
- 최악 적합 : 가장 큰 가용 공간을 할당

• 교체(Replacement) 정책 : 새로 들어온 프로그램이 들어갈 장소를 확보하기 위해 어떤 프로그램을 주기억장치에서 제거할 것인가를 결정하는 문제

1) FIFO(First In First Out) : 메모리에 가장 먼저 들어온 페이지부터 교체한다.
2) 최적(Optimal) 교체(OPT) : 가장 낮은 페이지 부재율을 가진 알고리즘
3) LRU(Least Recently Used)
- 가장 널리 사용되는 방법으로, 가장 오랫동안 사용되지 않은 페이지를 교체하는 기법
- 지역성에 의존하며, 불러왔던 시간을 기록하기 위한 오버헤드 발생하며 구현하기 어렵다.
4) LRU 근사 페이지 교체 :: SCR(Second Chance Replacement)
- 참조 Bit를 두어 FIFO의 단점인 오랫동안 주기억장치에 있으면서 자주 쓰이던 페이지가 교체되는 것을 막기 위한 기법.
- 참조 bit가 0이면 교체 / 1이면 0으로 변경 후 다시 큐에 적재
5) LRU 근사 페이지 교체 :: NUR(Not Used Recently)
- LRU의 단점인 시간 오버헤드를 적게 하는 기법
- 참조 Bit와 변경 Bit를 사용

6) LFU(Least Frequently Used)
- 참조 횟수가 가장 적은 페이지를 교체하는 기법
- 최근에 적재된 페이지가 교체될 가능성이 있다.

7) MFU(Most Frequently Used)
- 참조 횟수가 가장 큰 페이지를 교체하는 기법
- LFU와 MFU는 잘 사용되지 않음.

• 쓰레싱(Thrashing)
- 너무 자주 페이지 교체가 일어나는 현상.
- 페이지 교체↑ → 페이지 부재 빈번 → 처리시간 < 교체시간

• 페이지 부재 빈도(PFF, Page-Fault Frequency)

[가상기억장치]

• 페이징(Paging)
- 기억장치의 영역을 동일한 크기로 나누어 적재하는 방법
- **내부 단편화 발생**

→ 80M 적재 후 20M(내부 단편화 공간)의 공간이 남은 모습

• 세그먼테이션(Segmentation)
- 기억장치의 영역을 다양한 크기의 논리적인 단위로 나누어 적재하는 방법
- **외부 단편화 발생**

→ 100M를 적재하고 싶지만 여유 공간이 부족해서 적재를 못하는 모습

• 페이지 사상(mapping) 방법
- 연관 사상(associative mapping) : 위치 지정이 아닌 내용 지정으로 주기억장치보다 접근시간이 빠른 연관기억장치에 페이지 사상표 전체를 넣는 방법으로 가장 빠르고 융통성 있는 구조.
- 직접/연관 사상 혼용 방법 : 지역성의 원리를 이용
- 직접 사상 : 주기억장치에 페이지 사상표가 위치하며, 두 번의 기억장치 접근이 필요

[디스크 스케줄링 기법]

• FCFS 기법
- 작업 요청 순서대로 서비스 받음
- 구현이 쉽고 작업 우선순위에 대해 공평하지만, 응답시간이 길어질 수 있다.

• SSTF(Shortest Seek Time First) 기법
- 현 위치에서 탐색거리가 가장 짧은 요청이 서비스를 받는다.
- 평균 응답시간이 짧아지지만 응답시간의 편차가 심함. (기아현상 발생 가능)

• SCAN 기법
- 진행방향상의 가까운 요청부터 서비스. 부하가 적을 때 우수
- 가장 바깥쪽 실린더에 도달하면 진행방향을 바꿈
- SSTF에 비해 응답시간의 편차가 적고, 차별대우를 줄였다.
- 바깥쪽 트랙이 가운데 트랙보다 차별대우를 받는다.

• C-SCAN 기법
- 한쪽 방향으로만 진행하면서 요청을 서비스하며 끝에 도착하면 처음부터 진행.
- 부하가 많을 때 우수
- 안쪽, 바깥쪽 트랙의 차별대우를 완전히 제거
- 응답시간의 편차가 적음

• N-step SCAN 기법
- SCAN 알고리즘에서 방향 전환 시 먼저 요구한 N개의 요청만 서비스
- 헤더가 있는 실린더에 요청이 집중될 때 발생하는 무한대기를 제거

• SLTF(Shortest Latency Time First) : 회전시간 최적화 스케줄링

14. 보안 운영체제 (출제빈도 0.5%)

[보안 운영체제 개요]

• 보안 운영체제 : 기존의 운영체제 내에 보안기능을 통합시킨 보안 커널을 추가로 이식한 운영체제
• 파일 보호 기법 : 파일의 이름 명명, 패스워드, 암호화

[보안 커널(Secure Kernel)]

• 신뢰 컴퓨팅 기반(TCB, Trusted Computing Base) : OS, H/W, F/W, S/W 등이 포함된 하나의 컴퓨터 시스템 내의 모든 보호 메커니즘
• 보안 커널은 TCB의 핵심이다.

• 참조 모니터(Reference Monitor) : 주체의 객체에 대한 모든 접근통제를 담당하는 추상 머신
• 참조 모니터, 보안 커널의 3가지 특징 : 격리성(분리, 부정조작 방지), 검증가능성(크기가 작음), 완전성(우회불가)
• '참조 모니터'는 추상적인 개념으로써, 이를 실제로 구현한 것이 '보안 커널'이고, 이 보안 커널로 구성된 것이 'TCB'이다.

[신뢰 플랫폼 모듈(TPM, Trusted Platform Module)]

• 신뢰 컴퓨팅을 위한 H/W, S/W 방법에서 핵심이 되는 하드웨어 모듈
• 특징
- 신뢰 컴퓨팅(TC, Trusted Computing)의 가장 하위에 위치하며, 훼손 방지가 필수적이기 때문에 하드웨어칩으로 구현하는 것이 일반적이지만 S/W로 구현하기도 한다.
- 물리적인 공격(도난)에도 정보 노출이 힘들다.
- TPM의 신뢰 관련 연산 : 암호화 키의 생성과 저장, 패스워드 저장, 무결성 검증, 디지털 인증서 관련 신뢰 연산의 제공

15. 클라이언트 보안 (출제빈도 1.6%)

[악성 소프트웨어]

• 악성 소프트웨어의 분류

- 독립형/기생형 : 다른 프로그램 없이 독립적으로 존재하지 못하는 악성 소프트웨어를 기생형으로 분류한다. 이런 기생형 악성 소프트웨어에는 바이러스, 논리폭탄, 백도어가 있다. 독립형으로는 웜, 좀비 프로그램 등이 있다.
- 바이러스형/비-바이러스형 : 자기 복제 여부로 분류하며, 바이러스형에는 바이러스, 웜이 있으며, 비-바이러스형에는 트로이목마와 백도어 등이 있다.

[바이러스(Virus)]

• 자기 자신 또는 자신의 변형을 복사하여 컴퓨터 작동에 피해를 주는 명령어의 집합
• 파일이나 부트섹터 등 감염대상을 필요로 한다. (기생형)

• 세대별 분류 : 원시형 - 암호화 - 은폐형 - 갑옷형 - 매크로
• 매크로 바이러스(Macro Virus)
- 플랫폼과 무관하게 사용되고, 쉽게 전파된다.
- 문서를 감염시키고 코드의 실행부분은 감염시키지 않는다.
- 일반적으로 사용자가 실행 파일보다 주의를 덜 하기 때문에 피해가 더 크다.

• 안티 바이러스 필터링 방법
- signature scanning : 특정 바이러스만이 가진 유일한 형태의 signature를 찾아내는 방법
- behavioral virus scanning : 바이러스가 수행 중에 어떤 행동을 보이는지 추적하는 방법.

• 바이러스 예방책
- 신뢰있는 업체에서 구입한 상업용 S/W를 사용한다.
- 바이러스 스캐너를 통해 정기적인 검사를 하고, 주기적으로 업데이트를 한다.
- Windows Script Host, Active X, VBScript, JavaScript는 비활성화 한다.
- 사용에 심하게 방해되지 않는 수준에서 보안 정책을 수립/적용하고 교육한다.

[웜(Worm)]

• 자기 자신을 복제하여 네트워크를 통해 스스로 확산시킨다.
• 다른 시스템에 직접적인 영향을 미치지 않는다. (↔ 트로이목마)
• 다른 프로그램에 기생하지 않는다. (↔ 바이러스)

• 웜의 종류
- MASS Mailer형 웜 : 대량 메일 발송을 통해 확산되는 웜. 감염된 시스템이 많으면 SMTP 서버의 네트워크 트래픽이 증가
- 시스템 공격형 웜 : OS 고유 취약점을 이용해 내부 정보를 파괴하거나, 컴퓨터를 사용할 수 없는 상태로 만들거나, 백도어를 설치하는 웜.
- 네트워크 공격형 웜 : 특정 네트워크나 시스템에 대해 서비스 거부(DoS) 공격을 수행한다.

[트로이목마(Trojan horse)]

- 자신의 실체를 드러내지 않으면서 마치 다른 프로그램의 한 유형인 것처럼 가장하여 활동하는 프로그램.
- **자기복제 X (↔웜, 바이러스), 다른 파일 감염 X**
- 사용자 정보를 유출, 파괴하거나 DoS 공격, 원격 조정, 키로거(Keylogger)에 이용
- 트로이목마의 일반적인 기능은 원격 조정, 패스워드 가로채기, 키보드 입력 가로채기(키로거), 시스템 파일 파괴가 있다.

  * Back Orifice : MS 관리자용 도구이며, 해커 그룹에서 개발한 백도어 프로그램이다.

[기타 악성 소프트웨어]

- 스파이웨어(Spyware)
  - 설치된 시스템의 정보를 주기적으로 원격지의 특정한 서버에 보냄
  - 광고나 마케팅을 목적으로 배포하면 애드웨어(Adware)라고도 한다.
  - 유용한 S/W를 제공하면서 목적을 달성
- Exploit : 하나 혹은 여러 개의 취약점을 공격하기 위한 스크립트 혹은 프로그램
- 루트킷(Rootkit) : 컴퓨터 시스템에 침입 후 관리자(root) 수준의 접근 허락을 얻기 위해 사용하는 해킹 도구 모음
- 조크(Joke) : 실제 바이러스는 아니지만 감염자에게 심리적인 위협이나 불안을 조장하는 프로그램
- Hoax : 남을 속이거나 장난을 목적으로 퍼뜨리는 가짜 바이러스

[웹브라우저 보안]

- 쿠키(Cookie) 개념
  - 사용자들이 웹 사이트를 편리하게 이용할 수 있도록 하기 위한 목적으로 개발
  - Client(웹 브라우저)에 저장된다. * 세션(Session) : 서버 측에 저장되는 정보. 인증과 같은 보안관련 정보는 세션에 저장
  - 쿠키를 이용하여 사용자의 정보를 수집할 수 있다.
  - 쿠키는 텍스트 파일이기 때문에 실행되지 않으며, 바이러스를 전파할 수 없다.
  - 쿠키는 쿠키 안에 저장된 도메인 이름을 갖고 있는 사이트에서만 유효하기 때문에, 다른 웹사이트에서 읽기가 불가능하다.

- 쿠키의 구조
  - 4개의 속성과 하나의 데이터를 가지는 구조체이다.

```
Set-Cookie: name=value; expires=[Date]; domain=[Domain]; path=[Path]; [secure]
```

  * 헤더 Set-Cookie : Server → Client
    Cookie : Client → Server

  - 유효기간(expire) : 기본적으로는 브라우저 종료될 때까지 사용할 수 있지만(임시 쿠키 / 세션 쿠키), 유효 기간을 지정하면 지정 기간까지 쿠키 데이터를 읽고 쓸 수 있다. (영구 쿠키 / 저장된 쿠키)
  - Path : 기본적으로 쿠키는 쿠키 데이터를 생성한 웹 페이지에서만 그 데이터를 읽을 수 있지만 이 항목을 지정해주면 해당 Path 이하에서는 그 쿠키 데이터를 공유할 수 있게 된다.
  - Domain : Path 항목이 한 개의 사이트 단위로 지정해 준다면, Domain 항목은 도메인 단위로 지정할 수 있다.
  - Secure : 쿠키 데이터의 전송 방법 지정. HTTPS 등을 사용할 수 있다. HTTPS 통신(전송구간 암호화)일 경우에 지정

- 쿠키 보안 취약점
  - XSS(Cross-Site Scripting) : 자바스크립트가 사용자의 컴퓨터에서 실행된다는 점을 이용한 공격
  - 스니핑(Sniffing) : 네트워크를 통해 전송되는 암호화되지 않은 쿠키를 탈취할 수 있다.

16. 윈도우 서버 보안 (출제빈도 3.3% / 최근 3회간 5문제 이상 출제)

[윈도우 파일시스템]

• 파일시스템 종류

	FAT16	FAT32	NTFS
용량	저용량(최대 2GB)	고용량(2GB~2TB)	대용량
암호화·압축	X	X	O
클러스터	한 클러스터에 1632KB	한 클러스터에 4KB	가변 클러스터(512B ~ 4KB)
장점	· 호환성 우수(대부분의 MS OS에서 호환) · 저용량 볼륨에 최적화		· 강력한 보안기능, 감사 기능 · 디스크의 효율적 사용 · 대용량 볼륨, 긴 파일이름
단점	· 보안 취약 · 대용량 볼륨에 비효율적		· NT계열외의 OS에는 호환 X · 저용량 볼륨에서 FAT보다 속도 저하됨

• NTFS 디스크 구조

MBR	VBR	MFT	시스템 파일	파일 영역

- MBR(Master Boot Recode) : 파티션 생성 시 물리적 디스크의 첫 번째 섹터에 위치한 512bytes 크기의 영역. 부팅 시 BIOS에 의해 POST 과정을 마친 후 이 영역의 부트 코드를 호출하게 된다.
- VBR(Volume Boot Recode) : 부팅을 위한 기계어 코드와 볼륨 및 클러스터의 크기, MFT의 시작 주소 등의 설정 정보를 저장
- MFT(Master File Table) : 모든 파일에 대한 데이터를 제외한 파일 이름, 크기, 생성 시간 등의 모든 정보가 저장
- 시스템 파일 : 복구하는데 사용할 파일들이 저장
- 파일 영역 : 각 파일에 대한 실제 데이터가 저장

• 부팅 순서

- 윈도우 XP 이전
  (1) POST(Power On Self Test) 수행
  (2) 기본 부팅 관련 설정사항 로드 : CMOS의 설정사항을 읽어 옴.
  (3) MBR 로드 : 부팅 매체에 대한 기본 파일시스템 정보를 읽어 옴.
  (4) NTLDR(NT Loader) 실행
  (5) ntoskrnl.exe 실행 : ntoskrnl.exe는 HAL.DLL을 읽어 옴.
- 윈도우 Vista 이후
  (1)~(3) 동일
  (4) 윈도우 부트 서브 시스템 실행
  (5) 윈도우 OS 로더(Winload.exe) 실행 : 각종 장치 드라이브를 로드하고, ntoskrnl.exe를 실행

[윈도우 계정]

• 설치할 때 기본적으로 생성되는 계정

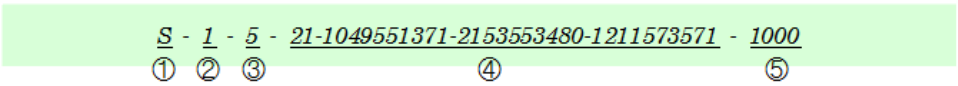
- Administrator : 관리자 권한의 계정. 사용자가 사용 가능한 계정 중 가장 강력한 권한을 가진다.
- SYSTEM : 시스템에서 최고 권한을 가진 계정. 로컬에서 Administrator보다 상위 권한이며 사용자는 이 계정으로 시스템에 로그인할 수 없다.
- Guest : 제한적인 권한을 가진 계정.

• 설치할 때 기본적으로 생성되는 그룹

- Administrators : 도메인 자원이나 로컬 컴퓨터에 대한 모든 권한이 있다.
- Account Operators : 계정을 관리하는 그룹
- Backup Operators : 시스템 백업을 위해서 모든 시스템의 파일과 디렉터리에 접근할 수 있다.
- Guests : 도메인 사용 권한이 제한된 그룹. 시스템 설정 변경 권한이 없다.
- Print Operators : 도메인 프린터에 접근할 수 있다.
- Users : 도메인과 로컬 컴퓨터를 일반적으로 사용하는 그룹. 개개인에 할당된 사용자 환경을 직접 만들 수 있지만, 설정할 수 있는 항목에는 한계가 있다.
- Power Users : Users 그룹 권한에서 디렉터리나 네트워크 공유, 공용 프로그램 그룹 생성, 시계 설정 권한을 추가로 가진다.
- Server Operation : 도메인의 서버를 관리할 수 있는 권한을 가진 그룹

• SID(Security Identifier)

- Unix/Linux 시스템의 UID와 비슷한 윈도우 개념
- 'Whoami /SID' 명령어나 'Whoami /USER' 명령어를 통해 SID를 확인할 수 있다.



① S : SID를 의미

② 1 : revision number

③ 5 : ID authority value

④ 21-1049551371-2153553480-1211573571 : 시스템의 고유한 숫자.

⑤ 1000 : 관리자는 500번 / Guest는 501번 / 일반 사용자는 1000번 이상의 숫자를 갖는다.

[윈도우 인증]

• LSA (Local Security Authority)

- 모든 계정의 로그인에 대한 검증, 접근 권한 검사
- SRM이 생성한 로그를 기록, 보안 서브시스템이라 불리기도 함

• SAM (Security Account Manager)

- 사용자/그룹 계정 정보에 대한 DB를 관리 (경로 : %systemroot%/system32/config/sam)
- 로그인 입력 정보와 SAM DB 정보를 비교해 인증 여부 결정

• SRM (Security Reference Monitor)

- 사용자에게 SID를 부여
- SID에 기반하여 파일/디렉터리에 대한 접근 권한을 결정
- 감사 메시지 생성

[공유 자료 관리]

• 파일과 폴더의 접근 권한

- 제공되는 권한 : 모든 권한, 수정, 읽기 및 실행, 폴더 내용 보기, 읽기, 쓰기
- 특정 사용자가 여러 그룹에 속한 경우 특정 파일/디렉터리에 대한 접근 권한은 누적된다.
- 파일에 대한 접근 권한이 디렉터리에 대한 접근 권한에 우선한다.
- 허용 권한보다 거부 권한이 우선한다.

• 기본 윈도우 공유 폴더

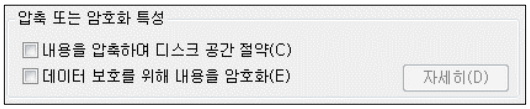
- C$, D$ 등 : 드라이브에 대한 관리목적 공유 폴더
- ADMIN$: 윈도우 설치 폴더에 접근하는 관리목적 공유 폴더
- IPC$(Inter Process Communication) : 네트워크 등에서 프로세스 간의 통신을 위해서 사용하는 통로

• 공유 폴더 관리

- net share 명령어로 숨겨진 공유 폴더를 확인할 수 있다. (숨겨진 공유 폴더 = 공유 이름 끝에 '$'가 붙은 폴더)
- C$, D$, ADMIN$ 공유가 숨겨졌더라도 매우 잘 알려져 있어 침입자가 네트워크에 액세스할 수 있는 위험이 있다. 대책방법으로 이런 공유를 비활성화 하는 방법이 있다. (net share 공유이름 /delete)
- net share 명령으로 공유 비활성시, OS 재부팅 시 다시 생성되기 때문에 레지스트리를 통해 완벽하게 제거할 수 있다. (HKLM)
- IPC$는 정상적인 방법으로 제거할 수 없으며, 제거할 경우 특정 서비스가 실행되지 않을 수 있다.

[암호 기능]

• EFS(Encrypting File System) : 파일/폴더를 암호화할 수 있는 기능



• BitLocker : 볼륨 단위(파티션 드라이브)의 데이터 암호화 기능

[레지스트리]

• 레지스트리는 윈도우 부팅 시 하이브 파일에서 값을 읽어 들여 구성된다. 하이브 파일에서 직접 읽어 들여 구성되는 키를 Master Key라 하고 Master Key로부터 값을 가져와서 재구성하는 키를 Derived Key라 한다.

- Master Key : HKLM(HKEY_LOCAL_MACHINE), HKU(HKEY_USERS)
- Derived Key : HKCU(HKEY_CURRENT_USER), HKCC(HKEY_CURRENT_CONFIG), HKCR(HKEY_CLASSES_ROOT)

• HKEY_CLASSES_ROOT (HKCR)

- 시스템에 등록된 파일 확장자와 그것을 열 때 사용할 어플리케이션에 대한 매핑 정보를 저장
- COM(Component Object Model) 오브젝트 등록 정보를 저장하고 있다.

• HKEY_CURRENT_USER (HKCU)

- 현재 시스템에 로그인하고 있는 사용자와 관련된 시스템 정보를 저장
- HKCU 키에서 설정한 내용이 HKU보다 우선권을 갖게 됨. (HKCU 키 값이 변경되면, HKU 키의 해당되는 키의 값도 변경됨)

• HKEY_LOCAL_MACHINE (HKLM)

- 컴퓨터에 설치된 하드웨어와 하드웨어를 구동시키는 데 필요한 드라이버나 설정 사항에 관련된 정보를 저장
- HKLM₩HARDWARE : 메모리에 휘발성으로 존재. 부팅 시 감지된 모든 하드웨어와 그 하드웨어 장치의 드라이버 매핑 정보가 저장
- HKLM₩SAM : 사용자와 패스워드, 소속 그룹, 도메인 정보 등 로컬 계정 정보와 그룹 정보를 저장. 시스템 계정만 접근 가능
- HKLM₩SECURITY : 시스템 범위의 보안 정책과 사용자 권리 할당 정보를 저장. 시스템 계정만 접근 가능
- HKLM₩SOFTWARE : 시스템 범위의 소프트웨어 목록과 그 환경 설정 정보(이름, 경로, 라이선스 정보 등)를 저장
- HKLM₩SYSTEM : 시스템이 부팅될 때 필요한 시스템 범위의 환경 설정 정보(시작시킬 서비스 목록 등)를 저장

• HKEY_USERS (HKU)

- 시스템에 있는 모든 계정과 그룹에 관한 정보를 저장
- 만약 시스템의 사용자가 한 명이라면 HKCU 설정 내용과 일치
- HKCU에 저장된 정보 전체와 데스크톱 설정, 네트워크 연결 등의 정보가 저장되어 있으며, user.dat에 그 내용을 저장

• HKEY_CURRENT_CONFIG (HKCC)

- 시스템이 시작할 때 사용하는 하드웨어 프로파일 정보를 저장
- HKLM에 서브로 존재하는 config의 내용(디스플레이와 프린터에 관한 설정 정보)만을 담고 있다.

• 레지스트리 공격/백업

- 레지스트리 백업 : 윈도우의 레지스트리 정보는 windows 폴더에 USER.DAT, SYSTEM.DAT 파일에 저장된다. 윈도우의 모든 시스템 정보를 백업하기 위해서는 USER.DAT, SYSTEM.DAT, SYSTEM.INI, WIN.INI 파일을 백업해야 한다.
- 부팅 시 악성코드 실행 공격 : 재부팅 시 악성 프로그램을 구동시키기 위해 레지스트리를 변조하게 된다.
  * HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩Run : 개별 사용자 지속
  * HKCU₩Software₩Microsoft₩Windows₩CurrentVersion₩RunOnce : 개별 사용자 일회성
  * HKLM₩Software₩Microsoft₩Windows₩CurrentVersion₩Run : 전체 사용자 지속
  * HKLM₩Software₩Microsoft₩Windows₩CurrentVersion₩RunOnce : 전체 사용자 일회성
- 특정 확장자 실행 시 악성코드 실행 공격 : 특정 확장자 실행 시 원하는 악성 프로그램을 실행시키게 할 수 있다.
  * HKCR 의 내용을 변경함으로써 수행

17. UNIX/Linux 서버 보안 (출제빈도 3.6%)

[유닉스 시스템]

- 대화식 운영체제 : Shell(셸)
- 멀티태스킹, 멀티유저, 호환성/이식성 우수
- 계층적 파일시스템 : 파일 관리 우수
- 뛰어난 통신기능, 다양한 기능의 유틸과 개발환경 제공

[파일시스템 구조]

- 부트 블록(Boot Block), 슈퍼 블록(Super Block), i-node 리스트, 데이터 블록(Data Block)으로 구성
- i-node(index node)
  - 파일에 대한 정보(파일의 허가권, 소유권, 그룹, 최근 수정된 시간)와 이 파일에 할당된 데이터 블록의 주소를 저장하는 필드들로 구성된 약 120bytes의 고정된 크기의 구조체
  - MAC Time 정보 : M (마지막 수정 시간), A (마지막 접근 시간), C (속성 마지막 변경 시간)
- Super Block : 파일시스템의 정보를 유지하는 자료구조. 사용하지 않고 있는 i-node와 디스크 블록의 위치정보도 가지고 있음

[명령어 :: 디렉터리, 권한 관리]

- 디렉터리 정보 출력

```
ls [-option] [File_name | Directory_name]
```

  - [-a], [-l]
  - [-R] : 하위 디렉터리에 있는 내용까지 보여준다.
  - [-F] : 어떤 종류의 파일인지 알려준다.
  - [-i] : 디렉터리 및 파일에 지정된 i-node 번호를 보여준다.

- 접근권한

권한	파일	디렉터리
읽기(r) [4]	파일을 읽거나 복사할 수 있다.	ls 명령어로 디렉터리 목록을 볼 수 있다.
쓰기(w) [2]	파일을 수정, 이동, 삭제할 수 있다. (디렉터리에 쓰기권한이 있을 시)	파일을 생성하거나 삭제할 수 있다.
실행(x) [1]	파일을 실행할 수 있다.	파일을 디렉터리로 이동하거나 복사할 수 있다. (cd 명령어 사용가능)

  - 생성 디폴트 값은 파일은 666, 디렉터리는 777이다.
  - 디렉터리가 777인 이유는 디렉터리에 실행 권한(x)이 없으면, 디렉터리 안으로 들어갈 수 없기 때문이다.

- 파일 권한 관리

```
chmod [-R] [permission] [File_name | Directory_name]
```

  - [-R] : 하위 디렉터리와 파일의 권한까지 변경하는 옵션

- 소유자/그룹 변경 : chown 명령어 / chgrp 명령어

[프로세스]

• PID 0 – swapper : 부팅 시간 동안 OS에 의해 생성

• PID 1 – init

• PID 2 – pagedaemon : swapper와 함께 커널 모드에서 영구적으로 실행되는 커널 프로세스

• 고아 프로세스 : 부모 프로세스가 자식 프로세스보다 먼저 종료된 경우 발생. PID가 1번인 init 프로세스가 고아 프로세스의 부모 프로세스 역할을 하게 된다.

• 좀비 프로세스 : 자식 프로세스가 종료되었지만 부모 프로세스가 이를 확인하지 못하는 경우 자식 프로세스는 부모 프로세스가 확인할 때까지 좀비 프로세스로 프로세스 테이블에 남아 있게 된다.

[Run Level]

Run Level	내용
0	PROM(Programmable Read-Only Memory) 모드 (BIOS 수준에서의 작업) / (LINUX) 시스템 종료
S, s	시스템 단일 사용자 모드, 로컬 파일시스템이 마운트되지 않은 상태
1	시스템 단일 사용자 모드, 로컬 파일시스템이 마운트된 상태
2	NFS를 지원하지 않는 다중 사용자 모드 (NFS 클라이언트 모드)
3	네트워킹을 지원하는 다중 사용자 모드 (NFS 서버 모드), UNIX 기본 Run Level
4	사용자 정의 레벨
5	시스템 종료 / (LINUX) X윈도우 환경으로 실행되는 다중 사용자 모드
6	시스템 재부팅

[명령어 :: 파일시스템 관리]

• 파일시스템 연결

```
mount [-option] [device] [mount_point]
```

- [-a] : /etc/fstab 파일에 정의된 모든 파일시스템을 마운트(mount)한다.
- mount 명령어만 사용 시 현재 시스템에 마운트된 정보를 출력한다.

• 파일시스템 연결 해제

```
umount [-option] [device | mount_point]
```

- [-a] : 마운트된 모든 파일시스템을 언마운트한다.
- [-f] : 해당 파일시스템을 사용하는 프로세스를 강제 종료하고 파일시스템을 언마운트한다.

• 하드디스크 사용량 (Disk Usage)

```
du [-option] [directory_name]
```

- 디렉터리의 하드디스크 사용량을 확인하는데 사용
- [-a] : 디렉터리뿐만 아니라 하위의 파일에 대한 정보도 출력
- [-s] : 현재 디렉터리가 차지하는 총 용량만 출력
- [-k] : 사용량을 KB 단위로 환산하여 출력

• 파일시스템 용량 정보 (Disk Free)

```
df [-option] [File_system | File]
```

- 파일시스템의 전체/사용가능 공간에 대한 정보를 출력
- [-h] : 사용자가 보기 편한 단위로 출력

[명령어 :: 프로세스 스케줄 관리]

• 정기적 스케줄 관리 (cron)

```
crontab [-e | -l | -r] [login_name]
```

- crontab 파일에는 정기적으로 처리해야 하는 작업 목록이 정의되어 있다.
- [-e] : crontab 파일 편집
- [-l] : crontab 파일 출력
- [-r] : crontab 파일 삭제
- 예시

```
# crontab -l
00 01 19 * * ls /var/log
// 매월 19일 01시 00분에 ls /var/log 명령어를 실행한다.
00 */2 * * * test.sh
// 매일 2시간 간격으로 test.sh를 실행한다.
00 08,20 * * 0 backup.sh
// 매주 일요일 08시, 20시마다 backup.sh를 실행한다.
```

• 일시적 스케줄 관리 (at)

- cron 데몬과 달리 at 명령은 정해진 시간에 한 번만 실행하고 소멸된다.

[시스템 보안]

• /etc/passwd 파일

- 파일 형식

[user_account] : [user_password] : [UID] : [GID] : [comment] : [home_directory] : [login_shell]

root : x : 0 : 0 : root : /root : /bin/bash

1) user_account : 사용자 계정
2) user_password : /etc/shadow 파일에 암호화되어 저장되어 있다.
3) UID : User ID. 보통 100번 이하는 시스템이 사용, 0번은 시스템 관리자를 나타낸다.
4) GID : Group ID
5) comment
6) home_directory : 로그인 성공 후에 사용자가 위치할 홈 디렉터리의 절대경로
7) login_shell : 로그인 셸의 절대경로. 로그인이 불필요한 계정의 셸은 /sbin/nologin, /bin/false으로 지정한다.

• /etc/shadow 파일

- 파일 형식

[user_id] : [encryption_pw] : [last_change] : [minilife] : [maxlife] : [warn] : [inactive] : [expires]

root : 6KrDCwLqA$Zde56t--itU8P/ : 17356 : 0 : 50 : 7 : 10 : 20000 :

1) user_id : 사용자 계정

2) encryption_pw : 일방향 해시 알고리즘을 이용해 암호화한 패스워드

- 형식 : $ id $ salt $ encrypted_password
- id : 적용된 일방향 해시 알고리즘 (1 : MD5 / 5 : SHA-256 / 6 : SHA-512 등)

3) last_change : 마지막으로 패스워드를 변경한 날(1970.01.01.부터 지난 일수로 표시)

4) minilife : 최소 패스워드 변경 일수(패스워드를 변경할 수 없는 기간)

5) maxlife : 최대 패스워드 변경 일수(패스워드 변경 없이 사용할 수 있는 일수)

6) warn : 경고 일수(maxlife 필드에 지정한 일수가 얼마 남지 않았음을 알림)

7) inactive : 최대 비활성 일수

8) expires : 계정이 만료되는 날(1970.01.01.부터 지난 일수로 표시)

• 파일 접근권한 (umask)

- 시스템 관리자는 /etc/profile에 umask를 설정하여 전체 사용자에게 동일한 umask 값을 적용할 수 있다.
- 개별 사용자에 대한 설정은 $HOME/.profile에 설정한다.

• 권한 상승(SetUID, SetGID)

- SetUID 비트를 가진 프로그램을 실행했을 때 사용자의 UID와 EUID(Effective UID)가 잠시 일치하지 않는 상태가 발생
- 만약에 test.exe라는 파일의 소유자가 root이고 SetUID 비트가 적용되어져 있다면, 일반 사용자 user가 test.exe 파일을 실행하는 동안 root의 권한을 가지게 된다.
- SetUID 비트가 설정된 파일은 공격 대상이 될 수 있으므로, SetUID 비트가 설정된 파일을 주기적으로 확인할 필요가 있다.
  * 관련 명령어 : `# find -perm 4000 -print`

• 디렉터리 접근권한(Sticky-bit)

- sticky-bit를 이용하여 디렉터리에 특별한 접근권한을 부여할 수 있다. (공유모드)
- sticky-bit가 설정된 디렉터리는 시스템에 있는 모든 사용자가 파일이나 하위 디렉터리를 생성할 수 있지만, 삭제는 소유주나 root에게만 허용된다. (/tmp같은 디렉터리에서 주로 사용)

[네트워크 보안]

• 슈퍼 서버 (inetd 데몬)

- inetd 데몬은 N개의 개별 서버를 하나로 통합하여 클라이언트로부터 서비스 요청이 올 때마다 해당 서비스와 관련된 실행 모듈을 실행해준다.
- 시스템에서 불필요한 서비스를 제한하기 위해 점검해야 될 것은 /etc/inetd.conf 파일이다.
- /etc/inetd.conf 파일을 변경하였다면 inetd 데몬을 다시 가동해야 반영된다. (refresh -s inetd 또는 /usr/sbin/inetd restart)
- inetd.conf 파일 구성

```
telnet    stream    tcp6    nowait    root    /usr/sbin/in.telnetd    in.telnetd
```

-> 왼쪽부터 서비스 이름, 소켓 타입(TCP-stream/UDP-dgram), 프로토콜, 대기 설정(TCP-nowait/UDP-wait), 로그인 이름, 서버(실행할 파일의 절대 경로), 인자(데몬 실행 시의 명령어)

• 접근통제 (TCP Wrapper)
- 서비스 요청 과정 : inetd 데몬 → tcpd 데몬 → 접근 권한 검사 → 해당 데몬에 연결
- inted 데몬이 관리하지 않는 standalone 데몬은 inetd 데몬이 통제할 수 없다. (모든 프로토콜에 대한 접근제어는 불가능)
- /etc/hosts.allow → /etc/hosts.deny 순으로 정책이 적용
- 설정 후에는 /usr/sfw/sbin/tcpdchk를 통해 설정 내용을 점검해볼 수 있다.

• PAM(Pluggable Authentication Modules, 장착형 인증 모듈)
- 리눅스 배포판에서 사용자 인증의 핵심
- 각 프로그램은 인증이 필요한 부분에 PAM 라이브러리를 호출한다.

[파일시스템 종류]

• ext2
- 2GB의 단일 파일 크기, 4TB의 디스크 사용 가능
- 파일시스템 복구기능 fsck(file system check) 제공. 복구시간↑, 복구시간 동안 시스템 사용 X

• ext3
- ext2와 호환
- 저널링(Journaling) 기술 제공. 빠르고 안정적인 복구기능

• ext4
- 성능 향상, ext2, ext3과 호환
- 16TB 이상의 파일 크기 지원, 1EB 이상의 볼륨 사용 가능
- 단편화 억제 기능 향상, 지연된 할당 파일시스템 기능 지원

[UNIX/Linux 서버 취약점 분석, 평가]

• root 계정 원격 접속 제한
- 원격 접속 시 root 계정으로 직접 로그인할 수 없도록 설정

• 패스워드 정책 점검
- 패스워드 최소 길이 : (Linux) /etc/login.defs 파일에 PASS_MIN_LEN 항목 설정
- 패스워드 최대 사용 기간 : (Linux) /etc/login.defs 파일에 PASS_MAX_DAYS 항목 설정
- 패스워드 최소 사용 기간 : (Linux) /etc/login.defs 파일에 PASS_MIN_DAYS 항목 설정

• PATH 변수 확인
- root 계정의 PATH 환경변수에 "."(현재 디렉터리)가 포함되어 있으면, 현재 디렉토리에 위치해 있는 공격자에 의해 생성된 파일이 실행될 수 있다.
- root 계정의 환경변수 설정 파일(/etc/profile)에서 현재 디렉터리를 나타내는 "."을 마지막에 위치시키거나 삭제한다.

• 파일 및 디렉터리 소유자 설정
- 소유자가 존재하지 않는 파일 및 디렉터리가 존재하는 지 확인하고 관리해야 한다.
- (Linux) # find / -nouser -print

18. 서버 보안 관리 (출제빈도 3.3%)

[서버관리자 업무]

• 시스템 관리자 계정으로 작업 시 직접 root 계정으로 로그인하는 것보다는 일반 사용자로 로그인한 후 su 명령을 이용해 root로 계정을 바꿔서 관리자 계정으로 변경하는 것이 좋다.

• 시스템 시작/종료
- shutdown -r now : 지금 즉시 시스템 재부팅
- shutdown -h now : 지금 즉시 시스템 종료
- half : 강제 종료
- reboot : 강제 재부팅

• 사용자 계정 관리
- /etc/passwd 파일, /etc/shadow 파일 관리
- 원격 접근권한 제거 : /etc/passwd 파일에서 /bin/bash 와 같은 셸 삭제
- 계정 사용기간 설정 : /etc/shadow 파일에서 사용기간 또는 만료일 설정

• 네트워크 관련 명령어

명령어	설명
ifconfig	통신 디바이스(NIC) 상태를 보여준다.
netstat -an	현 시스템에서 사용되는 통신 서비스의 상태를 보여준다.
top	시스템 자원(CPU, memory 등)의 사용현황을 보여준다.
ps -elf(-aux)	현 시스템에서 수행 중인 프로그램과 데몬 상태를 보여준다.
snoop	패킷을 캡처하여 분석한다.
nslookup	도메인에 대한 IP 정보 및 도메인 네임과 관련된 여러 검색이 가능
inetd	네트워크 슈퍼데몬인 inetd를 실행한다.

[윈도우 로그 분석]

• 이벤트(Event)라는 중앙 집중화된 로그를 수집하여 저장
• 중앙 집중화되어 있기 때문에 관리가 편하지만, 로그에 대한 보안 수준이 낮다.
• 현재 로그인한 사용자 확인 명령어, 툴 : net session 명령어, psloggedon 도구

• 윈도우 시스템 이벤트 로그 종류
- 응용 프로그램 로그(AppEvent.Evt) : **응용 프로그램이 기록**한 다양한 이벤트가 저장됨. 저장되는 이벤트는 소프트웨어 개발자에 의해 결정된다.
- 보안 로그(SecEvent.Evt) : 로그인 시도, 파일 생성·열람·삭제 등의 **리소스 사용**에 관련된 이벤트를 기록한다.
- 시스템 로그(SysEvent.Evt) : **Windows 시스템 구성요소**가 기록하는 이벤트. 구성요소의 오류를 이벤트에 기록한다.
- 추가적으로 디렉터리 서비스 로그, 파일 복제 서비스 로그, DNS 서버 로그 등이 있다.
- 이벤트 로그 파일은 %Windows%₩system32₩config 폴더에 위치
- 이벤트 로그 파일은 바이너리 형식이기 때문에, 별도의 프로그램(이벤트 뷰어 등)으로 열어야 한다.

• 감사 정책
- 어떤 로그를 남길지를 정의한 규칙
- 개체 액세스 감사 (감사 없음) : 특정 파일이나 디렉터리, 프린터 등과 같은 객체에 대한 접근 시도, 속성 변경 등을 탐지
- 계정 관리 감사 (실패) : 사용자/그룹의 추가/변경/활성화/비활성화, 계정 패스워드 변경 등을 감사
- 계정 로그인 이벤트 감사 (성공, 실패) : 도메인 계정의 로그인에 대한 사항을 로그에 남김
- 권한 사용 감사 (실패) : 권한 설정 변경이나 관리자 권한이 필요한 작업을 수행할 때 로그에 남김
- 로그인 이벤트 감사 (성공, 실패) : 로컬 계정 접근 시 생성되는 이벤트를 감사
- 디렉터리 서비스 액세스 감사 (실패) : 액티브 디렉터리 개체에 접근하는 사용자에 대한 감사 로그
- 정책 변경 감사 (성공, 실패) : 사용자 권한 할당 정책, 감사 정책 또는 신뢰 정책의 변경과 관련된 사항을 로그에 남김
- 프로세스 추적 감사 (감사 없음) : 프로세스를 시작하거나 중지할 때 해당 이벤트 발생
- 시스템 이벤트 감사 (감사 없음) : 시스템의 시동과 종료, 보안 로그 삭제 등 시스템의 주요한 사항에 대한 이벤트를 남김

• 로그 정책 설정
- 윈도우에서 로그 정책이 대부분의 정보를 로깅하지 않게 기본으로 설정되어 있다. 따라서 적절한 로깅 설정이 필요하다.
- 윈도우는 유닉스의 비해 로깅하는데 시스템 자원이 많이 소모되므로 모든 정보를 로깅하도록 설정하는 것은 바람직하지 않다.

[유닉스/리눅스 로그 분석]

• 유닉스는 로그를 여러 곳에 산발적으로 저장하기 때문에 모두 파악하고 관리하기 어렵지만, 다양한 로그들의 분석으로 공격자를 추적할 수 있게 된다.

로그	설명
utmp	· 현재 시스템에 로그인한 사용자의 상태를 출력 · 바이너리 형태. w, who, users, finger 등의 명령어로 확인
wtmp	· 사용자의 로그인/로그아웃, 시스템 종료/부팅 정보를 저장 · 바이너리 형태. last 명령어로 확인
sulog	· su 명령어(계정 변경)에 대한 로그. 텍스트 형태로 저장
lastlog	· 각 사용자의 최근(마지막) 로그인 시각
acct/pacct	· 시스템에 로그인한 모든 사용자가 수행한 프로그램에 대한 정보를 저장 · 시스템 자원을 비교적 많이 소모하여, 기본적으로 동작하지는 않는다. · 바이너리 형태. lastcomm 명령어로 확인
history	· 사용자별로 실행한 명령을 기록하는 로그. 텍스트 형태로 저장
btmp(Linux) loginlog(Unix)	· 실패한 로그인 시도에 대한 로그 · (Unix) 5회 이상 실패한 로그인 시도에 대한 로그
dmesg	· 부팅될 때 출력되는 모든 메시지를 기록
secure	· 원격 로그인 정보를 기록 · TCP_WRAPPER(xinetd)의 접속제어에 관한 로그도 남음
messages	· 시스템 운영에 대한 전반적인 메시지를 저장

[유닉스/리눅스 시스템 로그(/etc/syslog.conf)]

• 유닉스에서는 syslog에 의해 로그를 생성하고 관리한다.

• 데몬 프로세스인 syslogd는 OS에 의해 자동으로 시작하는데, 환경설정 파일 /etc/syslog.conf를 읽어서 어떤 로그를 어디에 남길지 결정하게 된다.

• 최근 리눅스는 기존 syslog를 개선한 rsyslog를 사용한다.

• /etc/syslog.conf 파일

```
facility.priority; facility.priority          action(logfile-location)
```

- facility(서비스)에 대하여 priority의 경우에 해당하는 상황이 발생 시 action에 해당하는 행위를 한다.(보통 로그 파일에 기록)
- facility(서비스) : *(모든 서비스), auth, authpriv(xinetd, telnet 등), cron, ftp, kern, lpr, mail, ntp, security, syslog 등
- priority(메시지 우선순위)

emerg	시스템 불능 상태
alert	즉각 조치 필요
crit	심각한 오류
err	일반적인 오류
warning	경고 메시지
notice	안내 메시지
info	일반 메시지
debug	디버깅 메시지
none	메시지 남기지 않음

- action : 특정 파일에 저장하거나, 터미널/콘솔, 특정 유저에게 보내지게 지정할 수 있다.

- /etc/syslog.conf 설정 예제

```
kern.*                                  /dev/console
```

→ kernel에 관련된 모든 로그를 콘솔에 출력

```
*.info;mail.none;auth.non               /var/log/messages
```

→ mail, auth 서비스에 대한 로그를 제외하고 모든 서비스에 대한 info 레벨 이상의 메시지를 /var/log/messages에 남긴다.

```
*.err                                   @user
```

→ 모든 서비스에 대한 err 레벨의 메시지를 user라는 사용자의 스크린으로 메시지를 보낸다.

[유닉스/리눅스 로그 관리]

• 텍스트 파일 형태의 로그를 실시간으로 모니터링하려면 'tail -f [로그 경로]' 명령을 이용한다.

• 로그 순환(logrotate)

- 시스템 로그 파일에 대해 순환, 압축 등을 해주는 리눅스 시스템 로그파일 관리기이다.

[IIS 웹 서버 로그]

• IIS(Internet Information Services) 웹 서버에서 로그는 기본 W3C 형식으로 남도록 설정되어 있다.

• W3C 주요 로그 필드

- date, time : 사용자가 페이지에 접속한 날짜와 시간
- c-ip : 웹 페이지에 접속한 사용자 IP
- cs-username : 웹 페이지에 접속한 사용자 계정
- s-computername : 웹 서버 이름
- s-ip, s-port : 웹 서버 IP, 웹 서버 포트
- cs-method, cs-url-stem, cs-url-query : HTTP 메소드, 요청 페이지, 요청 파라미터
- sc-status : 응답 코드

• 로그 예제

```
2019-02-26 01:38:38 192.168.33.3 GET /Part_one/Web08/web08.asp?id=admin&pw=1 80 -
        192.100.58.3 Mozilla/5.0+(Windows+NT+10.0;Win64;x64) 200 0 0 127
```

- `2019-02-26 01:38:38` : date, time
- `192.168.33.3` : s-ip
- `GET /Part_one/Web08/web08.asp?id=admin&pw=1` : cs-method, cs-url-stem, cs-url-query
- `80` : s-port
- `192.168.58.3 Mozilla/5.0+(Windows+NT+10.0;Win64;x64)` : c-ip, 클라이언트가 접속한 웹 브라우저 정보
- `200 0 0 127` : sc-status, 서버에서 클라이언트로 전송한 데이터의 크기, 클라이언트에서 서버로 전송한 데이터의 크기, 처리 소요 시간

[Apache 웹 서버 로그]

• access log : 클라이언트의 요청에 의해 웹 서버가 응답한 내용에 대한 로그

• error log : 클라이언트의 요청에 의해 웹 서버에 오류가 발생한 내용에 대한 로그

• 로그 예제(/etc/httpd/logs/access_log)

```
192.100.58.3 - - [26/FEB/2019:01:38:38 +0900] "GET /HTTP/1.1" 200 0 "-"
                "Mozilla/5.0+(Windows+NT+10.0;Win64;x64)"
```

[크래킹 S/W]

- John the Ripper : 패스워드 점검도구로 유명한 프로그램
- pwdump : 윈도우에서 패스워드를 덤프하기 위한 프로그램
- L0phtCrack : 패스워드 취약점 점검도구
- ipccrack : 패스워드를 원격지에서 추측하여 취약점을 점검하는 프로그램
- chntpw : 물리적으로 접근이 가능한 시스템에서 패스워드를 리셋시키는 프로그램
- ERD Commander : 윈도우에서 패스워드를 복구하는 프로그램
- 키로거(Keylogger) : 키보드로 입력한 정보를 로그로 남겨 실기간으로/정해진 시간에 공격자에게 전송하도록 하는 프로그램

[서버보안용 S/W]

• 취약점 분석 도구

- SATAN : 시스템 보안상의 약점을 찾아 보완할 수 있는 네트워크 분석용 보안 관리 도구
- SARA : SATAN 기반. UNIX에서 동작하고 HTML 형식의 보고서 기능
- SAINT : UNIX에서 동작하고 HTML 형식의 보고서 기능. 원격에서 취약점을 점검하는 기능 포함
- COPS : UNIX에서 동작하고 시스템 내부에 존재하는 취약점 점검. 취약한 PW를 점검.
- Nessus : 대부분의 OS에서 동작하고 클라이언트-서버 구조로 클라이언트의 취약점을 점검하는 기능, GUI 형태로 인터페이스가 편리하고 다양한 포맷으로 결과를 저장할 수 있다.
- nmap : 포트 스캐닝 도구

• 스캔 탐지 도구

- mscan : 메인 전체를 스캔하여 최근 많이 이용되는 주요 취약점을 한 번에 스캔할 수 있다.
- sscan : 네트워크를 통해 취약점 점검을 수행할 수 있는 도구
- portsentry : 실시간으로 포트 스캔을 탐지하고 대응하기 위한 프로그램. 공격 호스트를 경유하여 오는 모든 트래픽을 자동 재구성하는 기능이 있다.

• 무결성 점검 도구: tripwire, MD5, Fcheck, AIDE 등

• 침입탐지 및 방화벽

- Snort : 실시간 트래픽 분석, IP 네트워크에서의 패킷 처리를 담당하는 IDS
- IPchain / IPtable : 패킷 필터링 방화벽

19. 각종 시스템 보안위협 및 대응책 (출제빈도 3.0%)

[버퍼 오버플로우 공격]

• 프로세스 메모리 구조

- Text 영역 : 프로그램 코드와 상수가 정의. 읽기만 가능한 메모리 영역이다.
- Data 영역 : 전역 변수와 정적 변수가 저장되어 있는 영역
- Heap 영역 : 프로그래머의 필요에 따라 동적 메모리 호출에 의해 할당되는 영역
- Stack 영역 : 함수 인자 값, 함수 내의 지역 변수, 함수의 반환 주소 등이 저장되는 영역

• 스택 버퍼 오버플로우 : 보통 SetUID가 설정된 root 권한의 프로그램을 공격 대상으로 하고, 스택에 정해진 버퍼보다 큰 공격 코드를 삽입하여 반환주소를 변경함으로써 임의의 공격 코드를 root 권한으로 실행하도록 하는 공격 기법

• 힙 버퍼 오버플로우 : 힙에 할당된 공간이 함수에 대한 포인터를 포함하고 있는 경우, 공격자가 이 주소를 변경하여 겹쳐 쓴 버퍼에 있는 셸 코드를 가리키도록 하는 공격 기법

• 대응 방법

- 대응 방법은 크게 '컴파일 시간 방어', '실행 시간 방어'로 나눌 수 있다.

(1) 컴파일 시간 방어 : 컴파일할 때 검사하여 오버 플로우를 방지하거나 발견하는 방어 방법

- 고급 수준의 프로그래밍 언어 사용 : Java, ADA, Python 같은 언어는 컴파일러가 범위 검사를 강제로 수행하는 코드를 자동으로 추가한다.
- 안전한 함수 사용 : 입력 값을 검사하는 함수를 사용한다. (strncat(), strncpy(), fgets, snprintf() 등)
- 안전한 라이브러리의 사용
- 스택 보호 메커니즘(Stack Guard) : 함수의 진입과 종료 코드를 조사하여 함수의 스택 프레임에 대해 손상이 있는지 검사하는 방법. "canary"라는 값을 이용하여 탐지한다. 효과적인 방법이나 기존 프로그램들을 모두 새롭게 컴파일해야 한다.
- 스택 쉴드(Stack Shield) : 함수 시작 시 복귀 주소(RET)를 Global RET라는 특수 스택에 저장해 두었다가 함수 종료 시 저장된 값과 스택의 RET 값을 비교해 탐지하는 방법

(2) 실행 시간 방어 : 재컴파일 없이 기존 프로그램에 대해 방어할 수 있는 방법

- 주소 공간의 임의 추출(ASLR, Address Space Layout Randomization) : 각 프로세스 안의 스택이 임의의 다른 곳에 위치하도록 하는 것
- 실행가능 주소 공간의 보호(Non-Executable Stack) : 스택과 힙 영역을 실행 불능으로 만드는 방법

  * UNIX(Solaris 2.7 이상)에서 /etc/system 파일에 "set noexec_user_stack=1", "set noexec_user_stack_log=1" 설정으로 실행 불능 스택으로 만들 수 있다.

[포맷 스트링 공격]

• 포맷 스트링을 인자로 하는 함수의 취약점을 이용한 공격

• 외부로부터 입력된 값을 검증하지 않고 입출력 함수의 포맷 스트링을 그대로 사용하는 경우 발생

• 위협 요소

- 프로그램의 파괴 : 프로세스를 죽게 만들어 다른 공격을 수월하게 만들 수 있다.
- 프로세스 메모리 보기 : 시스템 내의 유용한 정보를 수집할 수 있다.
- 임의의 메모리 덮어쓰기 : 어떤 프로세스의 명령 통제권을 장악할 수 있다.

[레이스 컨디션 공격]

• 둘 이상의 프로세스나 스레드가 공유자원에 동시에 접근할 때 접근하는 순서에 따라 비정상적인 결과가 발생하는 조건/상황

• 실행되는 프로세스가 임시파일을 만드는 경우, 악의적인 프로그램을 통해 그 프로세스의 실행 중에 끼어들어 임시파일을 목적파일로 연결(심볼릭 링크)하여 악의적인 행위를 할 수 있다.

• 레이스 컨디션 공격의 대상 : 소유자가 root, SetUID 비트 설정, 임시파일을 생성하는 파일

• 생성되는 임시 파일의 이름을 알고 있어야 한다. -> 리눅스의 lsof 명령어를 통해 알아낼 수 있음.

• 대응 방법

- 임시파일에 접근하기 전에 임시파일에 대한 심볼릭 링크 설정 여부와 권한에 대한 검사 과정 추가
- 가능하면 임시파일을 생성하지 않는다.
- umask를 최소 022 정도로 유지하여, 임시파일이 공격자에 의해 삭제되지 않도록 한다.

[백도어(back door)]

• 시스템의 보안이 제거된 비밀통로

• OS나 프로그램 등에 접근할 때 정상적인 인증 과정을 거치지 않도록 하는 통로

• 개발과정에서 개발자에 의해서 만들어진 백도어를 특별히 트랩도어(Trap door)라고 부르기도 한다.

• 대표적으로 백 오리피스(back orifice)가 있다.

[시스템 자원 고갈 공격]

• 가용 디스크 자원 고갈 공격 : 디스크의 용량을 채우는 공격

• 가용 메모리 자원 고갈 공격 : malloc() 함수를 통해 메모리 할당을 반복적으로 하여 메모리를 고갈시키는 공격

• 가용 프로세스 자원 고갈 공격 : fork() 함수를 통해 가용 프로세스를 채우는 공격

• 프로세스 죽이기 공격 : 공격자가 root 권한을 획득한 상태에서 스크립트를 통해 프로세스를 죽이는 기법

[리버스 엔지니어링 공격]

• 공격대상 시스템 또는 프로그램을 리버스 엔지니어링을 통해 분석을 하여 취약점을 찾을 수 있으며, 이 취약점에 대한 공격 코드를 생성해낼 수 있다.

• 대응 방법

- 소스코드 난독화 : 프로그램 소스코드를 알아보기 힘들게 만드는 기술
- 바이너리 난독화 : 리버스 엔지니어링을 통해 분석하기 어렵게 변조하는 기술

[기타 시스템 보안 위협]

• 루트킷(Rootkit) : 공격자가 언제든지 시스템에 관리자 권한으로 접근할 수 있도록 비밀통로를 지속적으로 유지시켜주는 일련의 프로그램 집합

• GNU Bash 취약점(ShellShock) : 취약한 버전의 bash는 환경변수의 함수 선언문 뒤에 임의의 명령어를 삽입할 경우 환경변수에 설정된 함수 선언 시 함수 선언의 끝을 인지하지 못하고 삽입한 명령어까지 실행하는 취약점

• 논리폭탄(logic bomb) : 특정한 사건이 발생할 때 프로그램이나 일련의 코드를 실행하는 것