1 접근 통제 기술

1 접근 통제 기술

- 접근 제어(接近 制御) 또는 액세스 제어는
  누군가가 무언가를 사용하는 것을 허가하거나
  거부하는 기능을 말하는 것으로,

  파일, 프린터, 레지스트리 키, 디렉터리 서비스
  개체 등에 대한 서비스, 사용자의 권한을 결정 함

  (ACM/ACL, MAC, DAC, RBAC, ABAC)

1 접근 통제 기술

- 물리접근
  - 개인의 물리 접근은 비용 청구, 인증 등에 따라 허용될 수도 있고 허용되지 않을 수도 있음
  - 물리 접근에 대해서는, 부지나 건조물, 방 등의 입구에서 허가된 사람에게만 입장을 제한하는 것을 가리킴, 이러한 물리적 접근 제어는, 경비원, 경호원, 접수원 등의 사람에 의해서나 열쇠, 자물쇠 등의 기계적 수단, 카드에 의한 접근 시스템 등 기술적인 수단에 의해서 이루어짐

1 접근 통제 기술

컴퓨터보안

- 컴퓨터 보안에서의 접근 제어는 사람이나 프로세스가 시스템이나 파일에 읽기, 쓰기, 실행 등의 접근 여부를 허가하거나 거부하는 기능을 말함
- 컴퓨터 보안에 의한 접근 제어에는 인증, 인가, 감사가 포함됨(AAA), 이러한 것들은 물리적 기기에 의한 수단으로 수명 측정, 금속 자물쇠, 디지털 서명(전자 서명 - 부인 방지), 암호화(스니핑), 사회 장벽, 인간 및 자동화 시스템에 의한 감시 등을 포함

1 접근 통제 기술

- 컴퓨터보안
  - 인가는 역할 기반 접근 제어(RBAC), 접근 제어 목록(ACL), XACML와 같은 정책 언어 등에 추가 됨
  - 접근 제어 시스템은 식별 및 인증, 인가, 책임(Accountability)의 기본 서비스를 제공(AAA), 식별 및 인증은 시스템에 로그온 할 수 있는 사람을 결정하고, 인가는 인증된 사람이 생기는 것을 결정 하며, 책임은 해당 사용자가 무엇을 하였는지는 밝힘(로그)

2 접근 통제의 예

[정부(군대)에서 사용하는 보안 수준]

보안 등급	설명
Top Secret(TS)	국가 수준에서 가장 높은 보안 등급이다. 공개되었을 때 국가 보안에 '극도의 치명적인 피해'를 입힐 수 있는 정보로, 우리나라에서는 '1급 비밀'에 해당한다.
Secret	공개되었을 때 국가 보안에 '심각한 피해'를 입힐 수 있는 정보로, 우리나라에서는 '2급 비밀'에 해당한다.
Confidential	공개되었을 때 국가 보안에 '피해'를 입힐 수 있는 정보로, 우리나라에서는 '3급 비밀'에 해당한다.
Restricted	공개되었을 때 '의도치 않은 영향'을 미칠 수 있는 정보로, 우리나라에서는 '대외비'에 해당한다. 일부 국가에서는 이 등급이 없는 경우도 있다.
Unclassified	분류되지 않는 정보 중에서 민감하지 않은 정보이지만, 공개되어도 아무 문제가 없다는 의미는 아니다.

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

2 접근 통제의 예

- DAC(Discretionary access control)
  - '임의적 접근 통제'
  - 정보 소유자가 정보의 보안 수준을 결정하고 그에 대한 접근 통제까지 설정하는 모델
  - 중앙 집중화된 정보 관리가 어려워 엄격한 정보에 대한 접근 통제 거의 불가능
  - 대표적인 예는 유닉스나 윈도우에서 수행하는 파일에 대한 접근 통제 설정 (유닉스 파일 권한의 예 : rwx rw- rw-)
  - 권한 위임(grant) 가능

2 접근 통제의 예

- MAC(Mandatory access control)
  - '강제적 접근 통제' : 군사 환경에서 사용
  - Bell-LaPadula 모델, Biba 모델,
    Chinese-Wall 모델, Harrison-Ruzzo-Ulman 모델,
    Clark & Wilson 모델 등이 있음

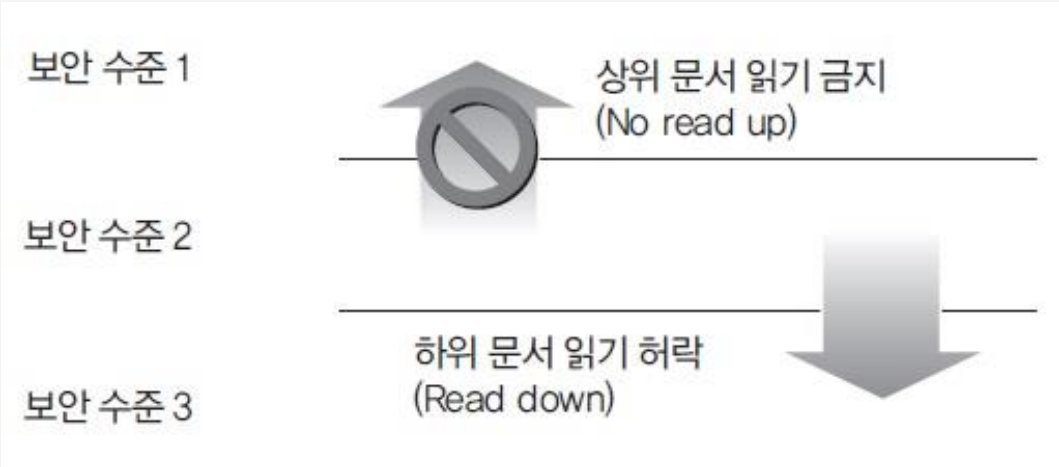
3 Bell-LaPadula 모델

- 정보의 기밀성에 따라 상하 관계가
  구별된 정보를 보호하기 위해 사용(기밀성)

- Property(Star property)
  : 자신의 권한보다 높은 보안 수준의 문서에는
  쓰기가 가능하지만 낮은 보안 수준의 문서의 경우
  쓰기 권한이 없음

3 Bell-LaPadula 모델

[Bell-LaPadula 모델의 읽기 권한 제한]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

3 Bell-LaPadula 모델

[Bell-LaPadula 모델의 쓰기 권한 제한]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

3 Bell-LaPadula 모델

- 미 국방부 지원 보안 모델로 보안 요소 중 기밀성 강조
- 최초의 수학적 모델로 강제적 정책에 의해
  접근 통제하는 모델
- 보안 정책은 정보가 높은 레벨에서
  낮은 레벨로 흐르는 것을 방지

3 Bell-LaPadula 모델

- BLP 속성
  - No Read Up
    : 보안 수준이 낮은 주체는 보안 수준이 높은 객체를 읽어서는 안 되는 정책, 주체는 객체와 동일한 등급이거나 객체보다 높은 등급일 때만 읽을 수 있음
  - No Write Down (star property)
    : 보안 수준이 높은 주체는 보안 수준이 낮은 객체에 기록해서는 안됨, 주체의 등급이 객체와 동일하거나 그 객체보다 낮아야 기록이 가능

3 Bell-LaPadula 모델

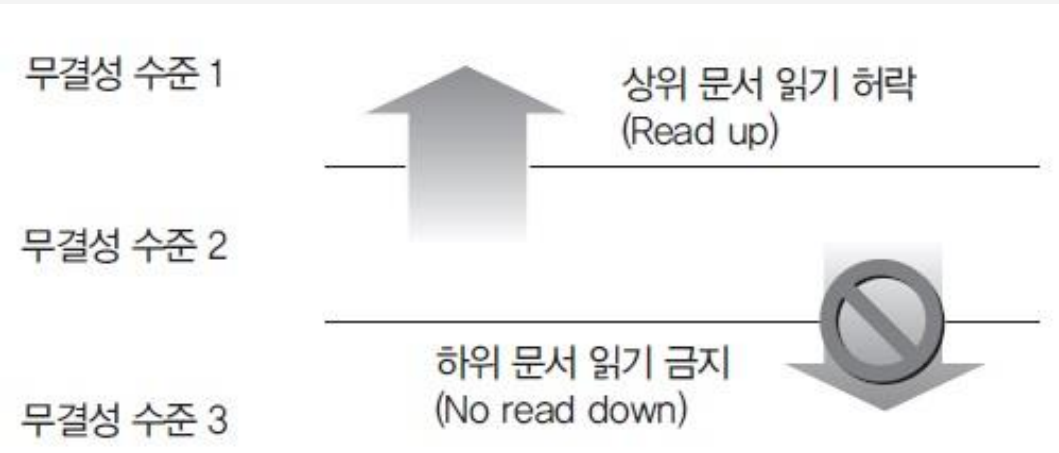
- 단점
  - 낮은 등급의 주체가 상위 등급의 객체를 보지 않은 상태에서 수정하여 덮어쓰기가 가능하므로 문서가 비인가자로부터 변경될 가능성이 있음

    (Blind Write 가능) → 무결성 파괴

4 Biba 모델

- 정보의 무결성을 높이는 것이 목적일 때 사용(무결성)

[Biba 모델의 읽기 권한 제한]



※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

4 Biba 모델

- 정보의 무결성을 높이는 것이 목적일 때 사용

[Biba 모델의 쓰기 권한 제한]

※ 출처 : 인터넷 해킹과 보안, 김경곤, 한빛아카데미, 2017

4 Biba 모델

- 무결성 모델
  - 인가되지 않은 사용자에 의해 데이터가 수정되는 것을 통제
  - 인가된 사용자라 할지라도 권한이 없는 데이터를 수정하는 것을 통제
  - 데이터는 내/외부적으로 일관성을 유지
- BLP의 단점을 보완한 무결성을 보장하는 최초의 모델
- 무결성의 3가지 목표 중 비인가자의 데이터 수정 방지 가능

4 Biba 모델

- 비바의 속성은 BLP의 반대 개념
- 비바 속성
  - No Read Down
    : 높은 등급의 주체는
    낮은 등급의 객체를 읽을 수 없음
  - No Write Up
    : 낮은 등급의 주체는
    상위 등급의 객체를 수정할 수 없음

5 클락-윌슨(Clark-Wilson) 모델

- 무결성 중심의 상업용 모델로 설계된 모델(상업용)
- 사용 응용 보안 요구 사항을 다루고 있음
- 금융/회계 데이터는 자산에 대한 정보를 취급하고 있어 변조 방지가 더욱 중요(무결성)
- 예측 가능하고 완전하게 처리되는 자료 처리 정책 필요
- 무결성을 위한 규칙 : 주체 → 프로그램 → 객체

6 만리장성(Brewer Nash, Chinese Wall) 모델

- 충돌을 야기하는 어떠한 정보의 흐름도
  차단해야 한다는 모델로 이익 충돌 회피를 위한 모델
- 직무 분리를 접근 통제에 반영한 개념이 적용(충돌)
- 금융 서비스 제공 회사가 이해 충돌의 발생을
  막기 위해 설계된 내부 규칙
- 적용 영역 : 투자, 금융, 파이낸셜, 로펌, 광고 분야

7 HRU 모델

- The HRU security model (Harrison, Ruzzo, Ullman model) is an operating system level computer security model which deals with the integrity of access rights in the system. It is an extension of the Graham-Denning model, based around the idea of a finite set of procedures being available to edit the access rights of a subject s on an object o. It is named after its three authors, Michael A. Harrison, Walter L. Ruzzo and Jeffrey D. Ullman.

7 HRU 모델

- Along with presenting the model, Harrison, Ruzzo and Ullman also discussed the possibilities and limitations of proving the safety of systems using an algorithm.

8 관리적 접근 통제

- 정책 및 절차를 수립하여 조직에서 보유하고 있는 중요 정보가 무엇인지 식별하고 그에 따른 통제 방안을 수립하는 것(관리적)

- 내부 임직원들이 회사의 보안 정책을 준수하도록 하기 위한 직원 통제 활동 (직무 분리, 직무 교대, 감독 구조, 보안 인식 훈련 등)

9 논리적 접근 통제(기술적 통제)

- 보호해야 할 정보에 접근하는 것을 제한하기 위해 사용하는 하드웨어와 소프트웨어 도구(논리적)

- 가장 많이 사용하는 것은 IP 주소 또는 MAC 주소에 따른 접근 통제

- 인터페이스 통제
  : 필요한 기능만 사용할 수 있도록 메뉴나 셸의 형태로 인터페이스 제공 (데이터베이스 - 뷰)

10 물리적 접근 통제

- 네트워크 분리(망 분리 - VDI, 가상화), 경계선 보안, 전산실 통제, 데이터 백업 등

- 스위치나 라우터 등의 네트워크 장비에 임의로 접근할 수 없도록 잠금 장치 등을 설치해놓아야 함 (보안 장비는 안전한가?)

2 입력 값 검증

1 SQL 인젝션 공격 코드 입력값 검증

- SQL 인젝션 공격 취약점을 제거하려면
  Prepared Statement 객체를 이용하여
  데이터베이스에 컴파일된 쿼리문을 전달하는
  방식이 현재 가장 안전(SQL 인젝션 공격)

1 SQL 인젝션 공격 코드 입력값 검증

안전한 코드의 예(자바)(쿼리 체크함)

```
try{
  String tableName = props.getProperty("jdbc.tableName");
  String name = props.getProperty("jdbc.name");
  String query = "SELECT * FROM" + tableName + "WHERE Name =" + name;
  stmt = con.prepareStatement(query);
  rs = stmt.executeQuery();
  ⋮
}
catch (SQLException sqle) { }
finally {}
```

1 SQL 인젝션 공격 코드 입력값 검증

안전한 코드의 예(자바)

```
try{
  String tableName = props.getProperty("jdbc.tableName");
  String name = props.getProperty("jdbc.name");
  String query = "SELECT * FROM ? WHERE Name = ?";
  stmt = con.prepareStatement(query);
  stmt.setString(1, tableName);
  stmt.setString(2, name);
  rs = stmt.executeQuery();
  ⋮
}
catch (SQLException sqle) { }
finally {}
```

2 크로스 사이트 스크립팅 공격 코드 입력값 검증

- 외부에서 스크립트 코드를 입력하지 못하도록 방어하는 것이 가장 효과적(XSS 공격)

- 일반적으로 스크립트를 입력하기 위한 문자열에는 <, >, &, “ 등이 있는데 이러한 것을 &lt, &gt, &amp, &quot 로 치환하는 것이 보안상 유리

2 크로스 사이트 스크립팅 공격 코드 입력값 검증

- 안전하지 않은 코드의 예(자바)(스크립트 가능)

```
<h1> XSS Vulnerability Sample </h1>
<%
   String name = request.getParaemeter("name");
%>
<p> NAME: <% =name %> </p>
```

2 크로스 사이트 스크립팅 공격 코드 입력값 검증

- 안전한 코드의 예(자바)

```
<%
  String name = request.getParameter("name");
  if (name != null)
  {
    name = name.replaceAll("<", "&lt;");
    name = name.replaceAll(">", "&gt;");
    name = name.replaceAll("&", "&amp;");
    name = name.replaceAll("₩"", "&quot;");
  }
  else { return; }
%>
```

3 악성 파일 업로드 공격 입력값 검증

- 필요한 파일의 확장자만 업로드를 허용
- 업로드 되는 파일을 저장할 때는 파일명과 확장자를 추측할 수 없도록 문자열 값을 변형하여 저장하는 것이 안전(원격 실행 가능)

3 악성 파일 업로드 공격 입력값 검증

안전하지 않은 코드의 예(자바)(확장자 체크 안함)

```
public void upload(HttpServletRequest request) throws ServletException
{
  MultipartHttpServletRequest mRequest = (MultipartHttpServletRequest)
request;
  String next = (String) mRequest.getFileName().next();
  MultipartFile file = mRequest.getFile(next);

  // MultipartFile로부터 file을 얻음
  String fileName = file.getOriginalFilename();

  // upload 파일에 대한 확장자 체크를 하지 않음
  File uploadDir = new File("/app/webapp/data/upload/notice");
  String uploadFilePath = uploadDir.getAbsolutePath()+"/"+fileName;

/* 이하 file upload 루틴 */
```

3 악성 파일 업로드 공격 입력값 검증

안전한 코드의 예(자바)

```
⋮
public void upload(HttpServletRequest request) throws ServletException
{
  MultipartHttpServletRequest mRequest = (MultipartHttpServletRequest) request;
  String next = (String) mRequest.getFileName().next();
  MultipartFile file = mRequest.getFile(next);
  if(file = = null) reture;
  //화이트리스트 방식으로 업로드 파일의 확장자를 체크
  if(fileName != null)
  {
    if(fileName.endsWith(".doc") || fileName.endsWith(".hwp")
    || fileName.endsWith(".pdf") || fileName.endsWith(".xls") )
    {
      / * file 업로드 루틴 */
      // 저장 시 파일명을 추측하지 못하도록 변형하여 저장
    ⋮
    }
    else throw new ServletException("에러");
  }
/* 이하 file upload 루틴 */
⋮
```

4 서버 측과 클라이언트 측 필터링

- 입력값 검증을 위한 코드를 JSP, 자바, ASP와 같은 서버 측 프로그래밍 언어에 포함할지, 자바스크립트와 같은 클라이언트 측 프로그래밍 언어에 포함할지에 따라 보안 수준이 달라짐(server-side script vs. client-side script)

- 클라이언트 측 필터링의 경우 공격자가 Burp Suite와 같은 웹 인터셉트 프로그램(프록시)으로 무력화할 수 있기 때문에 웹 애플리케이션에서 사용자 입력값 검증 코드를 반드시 JSP, 자바, ASP와 같은 서버 측 프로그래밍 언어 내에서 구현해야 함

5 화이트리스트 방식

- 필요한 특정 입력값만 받아들이고
  그 외의 모든 값은 필터링

- 특정 파일의 확장자(.hwp, .doc, .pdf, .xls)가
  아닌 모든 값을 차단(악성 파일을 pdf로 만들면?)

6 블랙리스트 방식

- 악성 패턴에 한해서만 필터링
- 대부분의 입력값은 허용하고
  일부 특정 입력값만 차단할 때 사용
  (내용을 보는 것이 아니라면 얼마든지 속일 수 있음)

7 입력값 검증

- 컴퓨터 보안에서, 개발자가 안전하다고 완벽히 확신하는, 알려진 좋은 입력이 있음, 또한 나쁜 문자들로 알려진 것들도 존재함(코드 인젝션을 유발할 수 있는) 이것에 기반하여 어떻게 입력이 관리되어야 하는지에 대한 두 가지의 다른 접근법이 존재

7 입력값 검증

- 화이트리스트 (좋다고 알려진 것들)
  : 화이트리스트는 좋은 입력으로 알려진 것들의 리스트,
  이것은 기본적으로 "A, B 그리고 C는 좋다"고 말하는
  리스트임

- 블랙리스트 (안 좋다고 알려진 것들)
  : 블랙리스트는 안좋은 입력으로 알려진 것들의 리스트,
  이것은 기본적으로 “A, B 그리고 C는 좋지 않다”고
  말하는 리스트임

7 입력값 검증

- 보안 전문가들은 블랙리스트가 우연히 나쁜 입력을 안전하게 다룰 수 있으므로 화이트리스트를 선호하는 경향이 있으나, 어떤 경우에 화이트리스트 솔루션은 쉽게 구현되지 않을 수 있음(화이트에 대한 정의)

- 필터 입력 : 화이트리스트 필터들(예시)

7 입력값 검증

- 예시
  - 모든 문자들이 A-Za-z 여야 한다는 입력 필터는 유닉스 애플리케이션을 셸 인젝션으로부터 보호하기 위해 사용 됨
  - 공격자들은 셸 인젝션을 시도하기 위해 입력 ; ls -l / 를 공급(파일 목록 확인)
  - 필터는 입력에 적용
  - 문자열 ; - / 는 화이트리스트에 없기 때문에 필터에 의해 던져 짐

7 입력값 검증

- 예시
  - 문자열 lsl 는 화이트리스트에 존재하기 때문에 필터에 의해 받아들여 짐
  - 오직 안전한 입력만 남게 되기 때문에 익스플로잇 시도는 실패함(취약점 공격)

7 입력값 검증

- 필터 입력 : 블랙리스트 필터들(예시)

- 만약 [:;.-/]이 포함된 문자열이 좋지 않다고 알려졌다면 ; ls -l / 는 받아들여 지지만, 원래 입력은 ls l 로 대체됨(;-/ 는 던져진다) 이 전략은 여러 문제들을 갖음

- 이것은 알려지지 않은 위협을 보호할 수 없음. 개발자가 고려하지 않은 다른 “안좋은” 입력들이 존재할 것(무엇이 블랙인가?)

7 입력값 검증

- 이것은 미래의 위협을 보호할 수 없음

  입력들이 현재는 안전하지만 사용된 언어의
  변화에 따라 미래에는 위험할 수 있는 표현을
  포함하고 있을 수 있음

  예를 들면, 유닉스 명령줄 보안 필터는
  C 셸에 대한 공격을 막기 위해 고안되었지만,
  만약 소프트웨어가 배시를 사용하는 환경으로
  바뀐다면 위험해질 것(셸의 종류 - C 셸, 배시 등)

3 전자우편 보안

1 전자우편 보안 개요

- 전자우편을 안전하고 편리하게 사용하기 위해서 전자우편에 대한 보안 기능은 필수적인 것
- 전자우편의 취약성

도청

- 네트워크상에서 메시지의 내용을 도청하여 악의적인 목적으로 유용 가능함

메시지 위·변조

- 공격자는 전자우편의 내용을 위조하거나 변조하여 수신자가 메시지의 내용을 오인하도록 할 수 있음

1 전자우편 보안 개요

전자우편의 취약성

송신자 변조

- 공격자는 노출을 피하기 위해 메시지의 송신자를 타인 명의로 변조한 후 악의적인 메시지를 송신할 수 있음

송신부인

- 송신자는 메시지 전송 사실을 부인할 수 있음

수신부인

- 수신자는 메시지를 수신하고도 이를 부인할 수 있음

1 전자우편 보안 개요

전자우편의 취약성

재전송

- 공격자는 송신자 변조 후에 동일한 메시지를 단시간에 반복 전송함으로 수신자의 메일 서버를 공격할 수 있음

1 전자우편 보안 개요

- 전자우편 보안

비밀성(confidentiality)(기밀성)

- 도청, 메시지 위·변조 등의 공격을 차단하기 위하여 메시지의 내용을 암호화하는 기능으로서 공개키/대칭키 암호 기술이 사용됨

메시지 무결성(integrity)

- 메시지 위·변조 여부를 확인하는 기능으로서 메시지 다이제스트(digest)(해시) 기술이 사용됨

1 전자우편 보안 개요

- 전자우편 보안

인증(authentication)

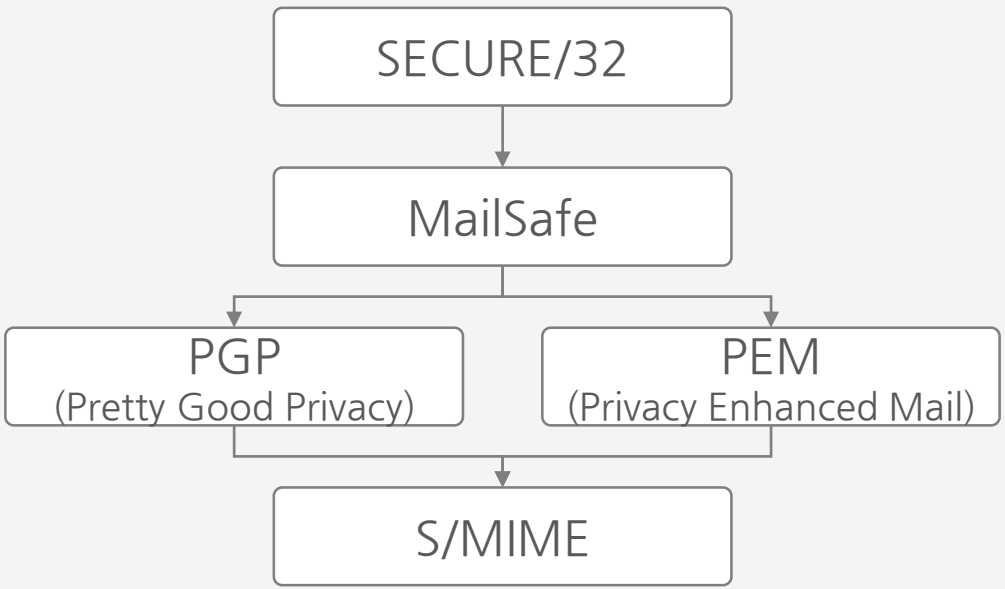
- 송신자 변조 등의 공격을 차단하기 위하여 송신자의 신원을 보증하는 기능으로서 전자서명 기술이 사용됨

부인 방지(non-repudiation)

- 송·수신 부인을 방지하는 기능으로서 전자서명 기술이 사용됨

1 전자우편 보안 개요

전자우편 보안의 역사



1 전자우편 보안 개요

전자우편 보안의 역사

- SECURE/32
  - 1980년대 초 발표
  - 최초의 전자우편 보안 시스템
  - 공개키 암호 기술을 기반으로 개발

- Mail-Safe
  - 1986년도 RSA Data Security사에 의해 개발
  - DES를 기반
  - 이후 개발된 PEM과 PGP의 모태가 됨

1 전자우편 보안 개요

전자우편 보안의 역사

- 1990년대 이후

  - PEM(Privacy Enhanced Mail), PGP(Pretty Good Privacy) 등이 개발

- S/MIME(Secure Multi-Purpose Internet Mail Extensions)

  - 최근까지 가장 대중적으로 사용되고 있는 전자우편 보안 시스템

2 PGP

- PGP(Pretty Good Privacy) 개요
  - 전자우편의 안전성을 위해 1991년 미국의 Phil Zimmermann에 의해 개발된 전자우편 보안 시스템
  - 비밀성, 인증, 무결성, 부인 방지, 압축 등의 기능을 지원

2 PGP

- PGP의 보안 기능
  - IDEA: 대칭키, CAST-128: 대칭키, RIPEMD-160: 해시 함수, Radix 64: Base 64

보안 기능	암호 알고리즘
메시지 암호화	IDEA, 3DES, CAST-128, RSA
전자서명(무결성, 인증, 부인 방지)	RSA, DSS/Diffe-Hellman, SHA-1, MD5, RIPEMD-160
압축	ZIP
이메일 호환성	Radix 64

3 PEM

- PEM(Privacy-Enhanced Mail) 개요
  - 인터넷에서 SMTP(Simple Mail Transfer Protocol)를 사용하는 기존의 전자우편 시스템의 보안상의 취약성을 보완하기 위한 것
  - 인터넷상에서 안전한 전자우편을 제공하기 위해 제안된 인터넷 표준안을 만드는 기술위원회(IETF: Internet Engineering Task Force) 표준안(draft)
  - Internet Activities Board에 의해 공식적인 표준으로 채택되지는 않았음

3 PEM

- PEM(Privacy-Enhanced Mail) 개요
  - RFC822 전자우편 규격과 함께 동작하도록 설계
  - 암호화, 인증, 키 관리 기능이 있음
  - 공개키와 비밀키(대칭키) 암호 시스템을 모두 지원

3 PEM

전자 우편보안 PEM, PGP 비교

PEM	PGP
IETF에 의해 만들어진 전자우편 관련표준	Philip Zimmerman
익명의 메시지를 허용치 않음	익명의 메시지를 허용
중앙 집중화된 키 인증	분산화된 키 인증
구현이 어려움	구현이 용이함
높은 보안성(군사용, 은행시스템)	비교적 낮은 보안성
이론 중심	현실세계 중심
많이 사용되지 않음	많이 사용

4 S/MIME

- S/MIME(Secure/Multipurpose Internet Mail Extensions) 개요
  - 안전한 전자메일 전송을 위한 산업체 표준 규약
  - 기존 MIME 형식의 전자메일 서비스에 암호 및 보안 서비스가 추가된 구조
  - S/MIME에서 제공하는 암호 및 보안 기능
    - 인증, 부인 불가, 메시지 무결성, 메시지 비밀성 등

4 S/MIME

- S/MIME의 암호 알고리즘
  - S/MIME에서는 메시지의 무결성, 부인불가, 암호화, 인증 등의 기능 제공을 위하여 다양한 암호 알고리즘이 사용됨

알고리즘 분류	송신자		수신자	
	필수	권고	필수	권고
축약	SHA-1 (Secure Hash Algorithm-1)	MD5 (Message Digest 5 Algorithm)	SHA-1 (Secure Hash Algorithm-1)	MD5 (Message Digest 5 Algorithm)
서명	DSS (Digital Signature Standard)	RSA (Rivest, Shamir Adleman Algorithm)	DSS (Digital Signature Standard)	RSA (Rivest, Shamir Adleman Algorithm)
암호화	DH (Diffie-Hellman Algorithm)	RSA (Rivest, Shamir Adleman Algorithm)	DH (Diffie-Hellman Algorithm)	RSA (Rivest, Shamir Adleman Algorithm)

4 S/MIME

- S/MIME의 암호 알고리즘

SHA-1(Secure Hash Algorithm-1)

- SHA는 미국 NIST(National Institute of Standards and Technology)에 의해 개발된 해쉬 알고리즘인 SHS(Secure Hash Standard)내에 정의된 해쉬 알고리즘으로서 데이터 무결성 기능 제공(입력 제한)

MD5(Message Digest 5 Algorithm)

- 길이 제한이 없는 입력 데이터로부터 128 비트 해쉬 코드를 생성시키는 해쉬 알고리즘으로서 데이터 무결성 기능을 제공(입력 제한 없음)

4 S/MIME

- S/MIME의 암호 알고리즘

DSS(Digital Signature Standard)

- 1991년 개발된 미국 NIST의 전자서명 표준
- SHA를 사용하는 DSA (Digital Signature Algorithm)를 포함

RSA(Rivest, Shamir Adleman Algorithm)

- 1977년 Ron Rivest, Adi Shamir, Leonard Adleman에 의해 개발된 암호 알고리즘(공개키)으로서 전자서명, 암호화, 키 교환 기능을 제공

4 S/MIME

- S/MIME의 암호 알고리즘

DH(Diffie-Hellman Algorithm)

  - 공개키 암호 기술의 시발로서 이산 대수 문제를 기반으로 한 알고리즘

4 S/MIME

- S/MIME 메시지
  - 인터넷 텍스트 메일 규격에 의한 헤더(header)와 MIME 엔티티(entity)로 구성
  - RFC 822 헤더(header)
    - 전자메일 전송에 필요한 송신자, 수신자, 전송 시각 등의 정보로 구성
  - MIME header
    - MIME 버전, 콘텐츠 형식, 인코딩 방식, 콘텐트 식별자, 콘텐트 설명 등의 필드를 포함

4 S/MIME

- S/MIME 메시지
  - CMS object (Cryptographic Message Syntax)
    - 서명과 암호화 관련 데이터 필드를 포함
  - MIME body
    - 문자열, 파일 등 다양한 형식의 데이터를 포함

4 S/MIME

S/MIME 메시지의 구조