

과정명	
02차시	애플리케이션 보안 운영

<1> 애플리케이션 보안 운영

[1] 애플리케이션

- 애플리케이션은 상용 소프트웨어 솔루션, 웹 애플리케이션을 포함하여 SI 구축 방식으로 개발된 모든 응용 프로그램을 말함
- 애플리케이션은 공격 유형에 사전 대비하지 않으면 애플리케이션 보안 취약점이 되므로 사전적인 예방 조치가 필요함

[2] 애플리케이션 공격 유형

(1) XSS 공격

- 공격자가 웹 서버의 애플리케이션에 XSS 취약점을 이용하여 공격용 악성 스크립트를 서버측 또는 URL에 미리 삽입해 놓고 이를 피공격자의 웹 브라우저로 전달받는 데이터에 포함시켜 개인 브라우저에서 실행되게 함으로써 공격하는 기법
- 공격자가 의도적으로 브라우저에서 실행될 수 있는 악성 스크립트를 웹 서버에 입력 또는 출력 시 위험한 문자를 중성화시키지 않고 처리하는 애플리케이션의 개발 과정에서 발생
- 일반적으로 자바스크립트에서 발생하지만, VB 스크립트, ActiveX 등 클라이언트에서 실행되는 동적 데이터를 생성하는 모든 언어에서 발생이 가능
- 비교적 쉽게 공격할 수 있고 웹 애플리케이션 개발 시 제거되지 않아 매우 광범위하게 분포

- 저장 XSS 공격, 반사 XSS 공격, DOM 기반 XSS 공격으로 분류됨

1) 저장 XSS 공격

- 웹 애플리케이션 취약점이 있는 웹 서버에 악성 스크립트를 영구적으로 저장해 놓는 방법
- 공격자가 웹 서버에 공격용 악성 스크립트를 미리 입력해 놓으면, 웹 사이트 사용자가 악성 스크립트가 삽입되어 있는 페이지를 읽는 순간 방문자의 브라우저를 공격하는 방식

2) 반사 XSS 공격

- 웹 애플리케이션의 지정된 변수를 이용할 때 발생하는 취약점을 이용하는 것으로 검색 결과, 에러 메시지 등 서버가 외부에서 입력 받은 값을 받아 브라우저에게 응답할 때 전송하는 과정에서 입력되는 변수의 위험한 문자를 사용자에게 그대로 돌려주면서 발생
- 공격자가 악성 URL을 피해자에게 전송하고 사용자가 클릭하도록 유도하여 URL을 클릭하면 클라이언트를 공격하는 방식

3) DOM 기반 XSS 공격

- 피해자의 브라우저가 HTML 페이지를 구문 분석할 때마다 공격 스크립트가 DOM 생성의 일부로 실행되면서 공격
- 페이지 자체는 변하지 않으나, 페이지에 포함되어 있는 브라우저측 코드가 DOM 환경에서 악성코드로 실행
- 저장이나 반사 XSS 공격과는 다르게 서버와 관계없이 브라우저에서 발생
- DOM 환경에서 악성 URL을 피해자에게 전송하여 사용자의 브라우저를 공격

(2) SQL Injection 공격

- 사용자 입력 값이나 URL 요청 등에 포함되는 파라미터에 악의적인 시스템 명령 또는 SQL 구문을 삽입하여 공격하는 기법

- 주로 로그인, 데이터 베이스 열람, 데이터 베이스 시스템 명령 권한 획득 등의 공격 시행
- 인증우회와 데이터 노출의 두 가지 방법이 있음

1) 인증우회

- 로그인 페이지를 타겟으로 행해지는 공격
- SQL 쿼리문의 TRUE/FALSE 의 논리적 연산 오류를 이용하여 로그인 인증 쿼리문이 무조건 TRUE 결과값이 나오게 하여 인증을 무력화 시키는 공격 기법
- 결과를 무조건 참으로 만들 수 있는 쿼리라면 무엇이든 그 패턴이 될 수 있음

2) 데이터 노출

- 데이터 절취를 목적으로 하는 방식
- 웹 사이트는 사용자의 입력 혹은 행위에 따라 URL 상의 파라미터로 특정 값을 전달하고 이를 데이터의 중요한 정보를 이용해 TRUE/FALSE로 알아냄

- 데이터 노출 방식으로는 Error based, Union based, Blind based, Time based가 있음

① Error based

- 데이터 베이스의 에러메시지를 기반으로 한 공격
- group by와 having 구문을 사용하여 에러를 일부러 유발
- 주로 MSSQL 데이터 베이스, Oracle, MYSQL에서 통함

② Union based

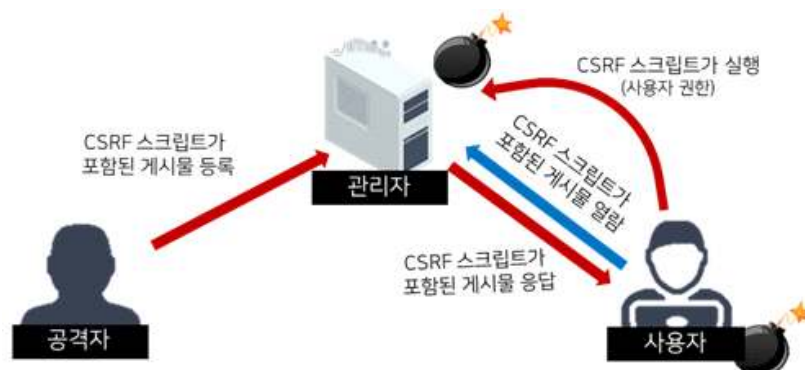
- 두 개의 쿼리문에 대한 결과를 통합해서 하나의 테이블로 보여주게 함
- 정상적인 쿼리문에 사용하면, 원하는 쿼리문 실행 가능
- Union 인젝션 성공을 위해서는 Union 테이블 컬럼 수와 데이터 형이 동일해야함

③ Time based

- 서버로부터 TRUE/FALSE의 응답을 통해서 데이터 베이스 정보 유추

(3) CSRF 공격

- 특정 웹사이트에 대하여 사용자가 인지하지 못한 상황에서 사용자의 의도와는 무관하게 공격자가 의도한 수정, 삭제, 등록 등의 행위를 요청하게 하는 공격
- 웹 애플리케이션이 사용자로부터 받은 요청에 대하여 사용자가 의도한 대로 작성되고 전송된 것인지 확인하지 않는 경우에 발생
- 해당 사용자가 관리자인 경우 사용자 권한 관리, 게시물 삭제, 사용자 등록 등 관리자 권한으로만 수행 가능한 기능을 공격자의 의도대로 실행 가능



[3] 애플리케이션 보안 프로토콜

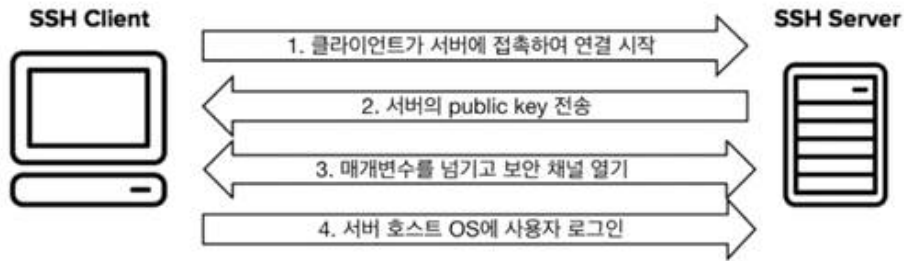
- 애플리케이션 보안 프로토콜은 OSI 7 Layer로 계층을 구분할 때 응용 계층에 해당됨

(1) SSH(Secure Shell)

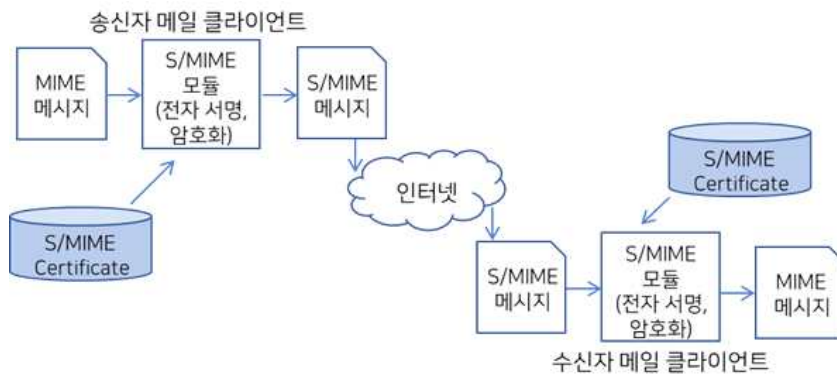
- 정보 통신망을 통하여 원격 단말기에 접속하여 명령을 실행하거나 파일 전송 또는

복사를 할 수 있도록 해주는 응용 프로그램이나 프로토콜

- 텔넷, rsh, rlogin 을 대체하여 네트워크에서 안전한 통신 수단을 제공하고 암호화 기법을 사용하여 통신 내용이 노출되더라도 기밀성을 유지



(2) S/MIME



- 전자 우편의 내용을 보호하기 위한 수단
- 기존에 널리 사용되는 SMTP를 대체하며 전자 우편 내용 보호를 위해 디지털 서명과 메시지 암호화의 기능을 제공
- S/MIME이 제공하는 보안 서비스에는 메시지에 대한 기밀성, 무결성, 사용자 인증, 송신 사실 부인 장비가 포함됨
- S/MIME 보안 서비스

기능	기본 동작
디지털 서명	<ul style="list-style-type: none"> SHA-256 메시지 해시 코드를 생성하고 이 메시지 다이제스트를 송신자 개인키를 가지고 RSA로 암호화하고 메시지에 포함
메시지 암호화	<ul style="list-style-type: none"> 송신자가 생성한 일회용 세션키를 이용해 CBC 모드의 AES-128로 메시지를 암호화하고, 세션키를 수신자의 공개키를 이용하여 RSA로 암호화하고 메시지에 포함
압축	<ul style="list-style-type: none"> 메시지를 압축하여 저장하거나 전송
이메일 호환성	<ul style="list-style-type: none"> 이메일 응용이 투명하도록 암호화된 메시지를 Base64로 변화하여 ASCII 스트링으로 바꿈

<2> 애플리케이션 보안 운영 개선

[1] 애플리케이션 보안 취약점

- 개발 단계에서 발생할 수 있는 개발자의 실수, 소프트웨어의 구조적인 결함 등에 의한 보안 취약점
- 보안 취약점 코드로는 CVE, CWE, CWE/SANS Top25, OWASP Top 10 등이 있음

(1) CVE(Common Vulnerabilities and Exposure)

- 알려진 보안 취약점에 대한 정보를 효과적으로 공유하고 대처하기 위하여 미국 MITRE 재단이 제공하고 있는 보안 취약점 목록 체계
- 'CVE-0000-0000' 형식으로 표현되며 2번째 4자릿수는 취약점이 등록된 해이고, 3번째 4자릿수는 등록 번호를 나타냄. 2015년부터는 자릿수에 제한을 두지 않고 필요에 따라 늘려 쓸 수도 있음

(2) CWE(Common Weakness Enumeration)

- 일반적인 취약점을 열거하여 소프트웨어 취약점 유형을 정형화한 목록을 소프트웨어 개발자 및 보안 관계자에게 제공하기 위한 것
- CVE와 동일하게 MITRE에서 제공

(3) CWE/SANS Top25

- CWE에서 심각한 소프트웨어 취약성을 유발시킬 수 있는 치명적인 약점을 정리한 목록
- 소프트웨어 개발 단계에서 흔히 발생하는 오류를 사전에 방지함으로써 취약점을 사전에 예방할 수 있도록 프로그래밍, 설계, 아키텍처의 오류에 대한 자료를 제공

(4) OWASP Top10

- MITRE, PCI DSS, DISA, FTC 등 다양한 기관이 참여하여 애플리케이션 보안 위험 상위 10개를 선정하여 발표
- 홈페이지를 통하여 애플리케이션 보안 위험뿐만 아니라 보안 도구와 표준, 안전한 코드 개발 및 보안성 검토, 라이브러리 등을 제공

- 애플리케이션 보안 취약점 유형

유형	보안 취약점 설명	예시
입력 데이터 검증 및 표현	♦ 프로그램 입력 값에 대한 검증 누락 또는 부적절한 검증, 데이터의 잘못된 형식 지정, 일관되지 않은 언어셋 사용 등으로 인하여 발생	SQL 인젝션, 크로스사이트 스크립트
보안 기능	♦ 보안 기능(인증, 접근 제어, 기밀성, 암호화, 권한 관리 등)을 부적절하게 구현 시 발생	부적절한 인가, 취약한 암호화 알고리즘 사용
시간 및 상태	♦ 동시 또는 거의 동시 수행을 지원하는 병렬 시스템, 하나 이상의 프로세스가 동작되는 환경에서 시간 및 상태를 부적절하게 관리하여 발생	제어문을 사용하지 않는 재귀 함수
에러 처리	♦ 에러를 처리하지 않거나, 불충분하게 처리하여 에러 정보에 중요 정보가 포함될 때 발생할 수 있는 취약점으로서, 시간 및 상태를 부적절하게 관리하여 발생	오류 메시지를 통한 정보 노출, 오류 상황 대응 부재
코드 오류	♦ 타입 변환 오류, 자원(메모리 등)의 부적절한 반환 등과 같이 개발자가 범할 수 있는 코딩 오류로 인하여 유발	널 포인터 역 참조, 부적절한 자원 해제
캡슐화	♦ 중요한 데이터 또는 기능성을 불충분하게 캡슐화하거나 잘못 사용함으로써 발생하는 보안 취약점으로 정보 노출, 권한 문제 발생	잘못된 세션에 의한 데이터 정보 노출
API 오용	♦ 의도된 사용에 반하는 방법으로 API를 사용하거나, 보안에 취약한 API를 사용하여 발생	DNS lookup에 의존한 보안 결정

[2] 애플리케이션 보안 솔루션

(1) 네트워크 보안

1) 방화벽

- 서로 다른 네트워크를 지나는 패킷을 허용, 거부하는 하드웨어 또는 소프트웨어

2) 침입 차단 시스템(IPS)

- 외부 네트워크에서 내부 네트워크로 유입되는 패킷 중에서 악의적인 패킷을 찾아 제어

3) 가상 사설망(VPN)

- 네트워크 전용 회선을 그룹 또는 조직에서 내부 사용자를 드러내지 않고 공유하기 위하여 쓰이는 사설 통신망

(2) 접근 통제

1) 네트워크 접근 통제(NAC)

- 내부망의 호스트가 네트워크를 접근할 때 보안 정책에 따라 네트워크 접속을 제어하고 통제

- 2) 시스템 접근 제어
 - 시스템 또는 서버에 접근 권한, 접근 경로 및 실시간 감시
- (3) 암호화
 - 1) DB 암호화
 - 데이터베이스에 저장된 데이터의 기밀성을 유지하기 위하여 암호화 및 복호화를 수행
 - 2) 자료 유출 방지(DLP)
 - 내부 정보의 유출을 차단하고 개인 정보를 보호
- 암호화 방식
 - 1) API 방식
 - 플러그인 방식에 비해 암호·복호화 속도가 빠름
 - 색인 검색이 불가능하고 권한 통제가 약함
 - 2) 플러그인 방식
 - 색인 검색을 지원하고 권한 통제 기능이 강력함
 - DB 서버의 자원을 공유하여 부하가 발생하고, 배치 작업 시에 암호화 성능이 저하됨
 - 3) 혼합 방식
 - 상황에 따라 플러그인 방식과 API 방식을 병행하여 사용할 수 있는 방식으로 병행 적용을 위한 기준과 관리 기준이 필요

<3> 애플리케이션 보안 신규 위협 대응

[1] 애플리케이션 보안 위협 식별

- 의도적인 공격과 내제되어 있는 보안 취약점으로 구분

(1) 악의적인 공격

- 내·외부 사용자가 정보 시스템의 파괴, 중요 정보의 누출 등과 같이 악의적인 의도를 가지고 행하는 사이버 테러
 - 1) 해킹
 - 악성 프로그램, 백 도어 프로그램 등 여러 가지 공격 기법을 이용하거나 보안 취약점을 이용하여 정보 시스템을 공격하는 기법
 - 2) 피싱
 - 전자우편, 메신저 등을 이용해 공격 대상이 신뢰하는 대상으로 위장하는 사회 공학적 공격 기법
 - 3) 스팸
 - 불특정 다수에게 전자 우편, 메시지 등으로 광고성 정보를 송신하는 것으로 특정 사이트로 접속을 유도하여 악성 코드를 감염시킴
 - 4) DoS/DDoS
 - 좀비 PC를 이용하여 공격 대상 정보 시스템의 자원을 고갈시켜 서비스 불가 상태로 만드는 공격 기법

(2) 애플리케이션 보안 위협 식별 과정

- 애플리케이션 보안 위협 식별 과정은 크게 애플리케이션 취약점 파악, 악의적인 공격 위험 파악, 보안 관련 법 및 규정 확인 순으로 이루어짐
 - 1) 운영 애플리케이션 현황 파악
 - 운용 중인 애플리케이션 보안 솔루션, 정보 시스템 운영 체계, 애플리케이션 현황 파악
 - 보안 취약점은 특정 솔루션, 운영 체계, 애플리케이션 소스 코드에 사용된 언어, 스크립트에 따라 알려져 있는 취약점이 다르므로 자산 목록, 아키텍처 정의서 등을 이용하여 파악
 - 2) 점검 체크리스트 작성

- 애플리케이션 현황에 따라 기본적인 점검 체크리스트 작성
- 주요 정보 시스템의 운영 체계, 보안 솔루션에 대해 알려진 취약점을 찾아 점검 체크리스트 작성
- 3) 정보시스템 구성 및 아키텍처 현황 파악
 - 네트워크 망 분리, DMZ 구성 등 파악
- 4) 위험 노출 대상 식별
 - 네트워크 구성에 따라 외부의 공격으로부터 1차적으로 보호되고 있는 영역과 공격 위험에 노출된 영역 식별
- 5) 취약점 점검
 - 포트 스캔, 보안 취약점 분석 도구를 사용하여 공격 위험에 노출된 취약점은 없는지 점검
- 6) 보안 관련 법 및 규정 확인
 - 정보 보안 관련 법 및 관련 규정을 따라 반드시 지켜야 하는 보안 준거 사항의 관련 조항을 명시하여 작성

[2] 애플리케이션 보안 아키텍처 개선

- 소프트웨어 아키텍처는 소프트웨어 구성 요소와 요소간의 관계를 정의한 것으로 소프트웨어 설계의 지침이자 원칙
- 소프트웨어 아키텍처에는 5가지의 구조가 존재함
- (1) 저장소 구조
 - 중앙의 저장소를 각 서브 시스템들이 독립적으로 이용하고 통신하는 구조
- (2) MVC 구조
 - 모델(정보 저장), 뷰(표현), 컨트롤러(제어 기능)로 구성
- (3) 클라이언트/서버(C/S) 구조
 - 사용자가 이용하는 클라이언트가 서버와 통신하여 데이터를 제어하는 구조
- (4) 계층 구조
 - OSI 7 Layer와 같이 각 계층이 정해진 인터페이스에 따라 통신하는 구조
- (5) 파이프 필터 구조
 - 각 서브 시스템이 입력 값을 처리하고 결과 값을 다음 서브 시스템으로 보내는 작업이 반복되는 구조
 - 각 서브 시스템 : 필터, 데이터 전송 관계 : 파이프
- 정보 시스템 아키텍처는 핵심 요구 사항과 고려 사항에 따라 아키텍처 설계 원칙을 수립하고, 이를 구현할 수 있는 구체적인 설계 요구 사항을 도출하여 설계함
- 아키텍처 설계를 위한 요구사항에는 업무, 기능, 보안을 위한 요구사항 등이 있음
- 아키텍처 고려 사항으로는 보안 관련 법 제도 준수, 개인 정보 보호 관련 법 제도 준수, 개발 시 소프트웨어 보안 개발 준수 등 점검해야 할 항목들이 포함됨

- 애플리케이션 개발 보안은 소프트웨어 개발 단계에서 보안 취약점을 사전에 제거하고, 개발 각 단계에서 수행하는 보안 활동을 통하여 안전한 소프트웨어를 개발하기 위한 개발체계임

구분	내용
대상	◆ 정보 시스템 감리 대상 정보화 사업
범위	◆ 설계 단계 산출물 및 소스 코드(상용 소프트웨어 제외)
기준	◆ 설계 단계 보안 설계 기준 총 20개 항목 ◆ 구현 단계 보안 약점 제거 기준 총 47개 항목
기타	◆ 감리 법인이 진단 도구 사용 시 국정원장이 인증한 도구 사용 ◆ 감리 법인은 소프트웨어 보안 약점 진단 시 진단원을 우선적으로 배치

- 소프트웨어 개발 보안 방법론에서는 요구사항 분석, 설계, 구현, 테스트 및 유지 보수 각 단계별로 추가적인 보안 활동을 정의하고 있음

(1) 요구사항 분석

- 보안 항목 요구사항을 식별하는 보안 활동 수행
- 시스템으로 관리되어야 하는 보안 항목들을 식별하고 각 정보의 보안 등급과 법 제도에 따른 관리 중요성 점검

(2) 설계

- 시스템 분석을 통하여 보안 위협 요소를 도출하는 위협 모델링, 보안 통제 기준 설정 등의 개발 보안 가이드가 제시하는 작업을 기존 개발 프로세스에 추가적으로 수행
- 최대한 많은 위협 요소를 도출하여 해당 위협들이 충분히 제거될 수 있도록 시스템 설계

(3) 구현

- 표준 코딩 정의서 또는 소프트웨어 개발 보안 가이드를 준수하여 개발
- 단위 테스트에서 충분히 보안 취약점을 제거할 수 있도록 함
- 인스펙션, 코드 리뷰 등을 통하여 소스 코드 수준에서 안정성이 보장되도록 함

(4) 테스트

- 설계 단계에서 도출된 위협들이 구현 단계에서 해당 취약점이 제거된 소프트웨어로 개발되었는지를 검증하는 작업 수행

(5) 유지 보수

- 개발 단계에서 취약점 제거를 위해 노력하였음에도 발생 가능한 보안 사고에 대한 관리, 사고 대응 및 주기적인 보안 업데이트 수행