SQL Injection 공격 방지를 위한 코드 변환 애플리케이션 설계 및 구현

하만석, 박수현
국민대학교 비즈니스IT전문대학원
e-mail : msha@kookmin.ac.kr, shpark21@kookmin.ac.kr

Design and Implementation of SQL Injection attack prevention code conversion application

Man-Seok Ha, Soo-Hyun Park
Graduate School Of Business IT, Kookmin University

요 약

인터넷의 보급에 따른 신속정확하고 편리한 정보처리의 장점에도 불구하고 최근 들어 급증하고 있는 보안 관련 사고들로 인하여 개인정보 및 기업정보의 관리에 대한 대책 마련이 시급한 가운데 있다. 그 중에서도 SQL 삽입 공격에 의한 악의적인 관리자 권한 획득 및 비정상적인 로그인 등으로 인하여 많은 피해가 발생하고 있다. 현재 SQL Injection에 관련된 대부분의 연구는 공격을 탐지하는 방법에 초점이 맞추어져 있다.

본 논문에서는 프로그램 코드를 분석하여 따옴표가 포함된 취약한 인라인 SQL 쿼리 구문을 찾아서 매개변수화된 쿼리로 변경하는 기능을 제공함으로써 근본적인 해결책을 찾고자 하였으며 Java, C#.net 등 다양한 언어를 지원하여 개발 업무에서의 활용성을 높이고자 하였다.

1. 서론

인터넷의 보급과 스마트폰의 대중화에 힘입어 데이터베이스의 사용량과 범위는 폭발적으로 증가하고 있다.

반면에 프로그램 개발자들은 바쁜 업무로 인하여 다양한 보안 공격 위협에 대한 충분한 고려를 하지 못하고 프로그램을 개발하는 경향이 있다.

신규 프로젝트 뿐 아니라 기존에 개발된 프로그램들을 유지보수하고 업그레이드하기 위해서도 역시 많은 비용과 인력이 투입되어야 하므로 취약한 부분에 대한 보완이 잘 이루어지지 못하는 경향이 있다.

SQL 삽입 공격으로 인해 그동안 많은 시스템들이 피해를 입은 바 있다. 현재 대부분의 시스템에서 SQL 삽입 공격에 대한 피해를 예방할 수 있도록 시스템을 보완했다.

하지만 아직도 여러 이유로 보완되지 못한채 방치되어 있는 시스템들이 많이 있다.

최근에 발생하고 있는 많은 보안사고들을 볼 때 아직도 SQL 삽입 공격과 같은 초보적이고 간단한 공격에도 대처하지 못하는 것은 IT강국으로서의 우리나라의 위상에도 걸맞지 않은 일이라고 생각된다.

그동안 SQL 삽입 공격에 대한 대부분의 연구는 대부분 공격을 탐지하여 대응하는 방향으로 초점이 맞추어져 있다.[1][2][3]

본 논문에서는 SQL 삽입 공격의 빌미를 제공하는 따옴표가 포함된 인라인 SQL 스크립트 코드를 분석하여 매개변수를 사용하는 코드로 변경함으로써 근본적인 문제를 해결하는 방식을 취하고자 한다.[4]

본 논문의 구성은 다음과 같다. 서론에서는 연구 배경 및 목적과 연구 범위 및 방법에 대하여 서술한다. 본론에서는 먼저 SQL 삽입 공격 및 방어 대책에 대한 이론적 배경에 관하여 살펴본 후 코드 변환 시스템을 구현하고자 한다.

설계 기법으로는 객체지향모델링 기법인 UML을 활용하여 시스템을 설계하고 이에 따라 코드 변환 소프트웨어를 구현하였다.

이후 본 논문과 기존 방식과의 비교 분석을 통하여 연구의 성과 및 향후 연구과제에 대하여 언급한다.

2. 본론

2.1 SQL 삽입 공격의 정의

SQL Injection은 어플리케이션의 사용자 입력 값에 SQL 코드를 삽입 또는 추가하고, 해당 SQL 구문을 가장 마지막의 SQL 서버에서 전달하여 해석 및 실행하는 과정에서 발생하는 공격이다. 모든 종류의 SQL 구문으로 만들어진 프로시저의 경우 잠재적으로 취약점을 갖고 있는데, 이는 SQL 자체 취약점과 함수에서 제공하는 다양한 코딩 옵션에 의한 취약점이다. SQL Injection의 가장 일반적인 형태는 파라미터 값에 SQL 명령을 삽입하고 실행하는 형태로 이루어진다. 간접 공격은 악의적인 목적의 코드를 테이블이나 메타데이터로 저장될 문자열을 감염(조작)시킨다. 해당 구문이 동적으로 SQL 명령으로 생성되면 악성코드가 실행되게 된다. 웹 어플리케이션에서 동적으로 SQL 구문 생성을 위해 전달하는 파라미터에 대해 검증이 실패했을 때 공격자는 SQL 구문 조작이 가능하다. 공격자가 SQL 구문을 조작 가능할 때 해당 구문은 어플리케이션 사용자 권한으로 실행될 것이다; SQL 서버에서 명령을 실행할 때, 명령에 의해 생성된 프로세스는 관련 컴포넌트(예를 들어 데이터베이스 서버, 어플리케이션서버, 웹 서버)와 똑같은 권한을 갖고 실행된다. 이는 종종 권한 상승을 부르기도 한다.[4][5]

2.2 SQL injection 방법

예를 들어 아래와 같은 Java 언어로 작성된 코드가 있다고 가정한다.

String sql = "select name from member where userid='"+userid+"' passwd='"+ passwd +"'";

사용자가 아이디에 test' or 1=1 - - 이라는 값을 입력하면 DB서버에 아래와 같은 명령어가 전달되게 된다.

String sql = "select name from member where userid='test' or 1=1 -- ' passwd='"+ passwd +"'";

위의 SQL 문장의 결과는 항상 true가 되므로 잘못된 계정임에도 불구하고 로그인이 되는 문제가 발생한다.

만약 사용자가 아이디에 admin' - - 이라는 값을 입력하면 DB서버에 아래와 같은 명령어가 전달되게 된다.

String sql = "select name from member where userid='admin' -- passwd='"+ passwd +"'";

만약 테이블에 admin이라는 관리자 계정이 존재한다면 관리자 권한으로 로그인이 된다. 따라서 중요한 정보가 노출되는 심각한 상황이 발생할 수 있다.[2][6][7][8]

2.3 매개변수화된 쿼리의 장점

따옴표가 포함된 인라인 SQL 스크립트와 달리 매개변수화된 쿼리의 장점은 다음과 같다.[4][9]

첫째, 가독성이 향상된다. 인라인 SQL 스크립트를 사용할 경우 SQL 문장을 작성할 때 따옴표를 잘못 입력하여 에러가 나는 경우가 많다. 하지만 매개변수화된 쿼리를 사용하면 따옴표를 자동으로 변환처리해 주기 때문에 오류가 덜 발생하고 가독성이 향상된다.

둘째, 쿼리의 실행속도가 개선된다. 매개변수화된 쿼리는 매번 다른 쿼리로 인식되어 DB서버에서 SQL 문장 해석, 실행계획 구성을 하게 되지만 매개변수화된 쿼리는 같은 패턴의 SQL 문장으로 인식되어 최초 한번만 SQL 문장 해석, 실행계획 구성을 한 후 두 번째부터는 생략하게 되어 처리속도가 빠르다.

셋째, 인라인 SQL 스트립트는 따옴표가 포함된 SQL injection에 매우 취약하지만 매개변수화된 쿼리는 따옴표를 자동으로 처리하므로 SQL injection을 근본적으로 방지할 수 있다.

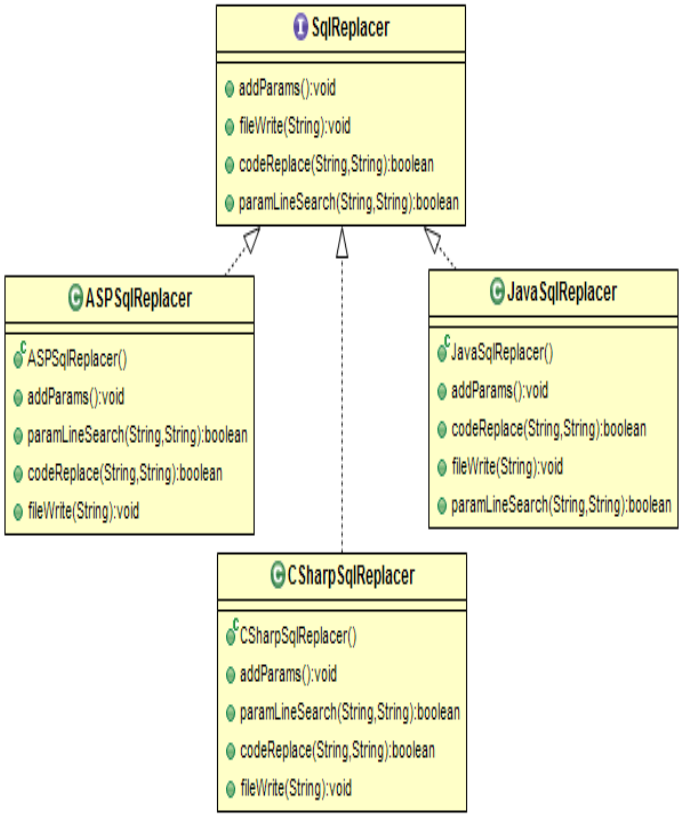
2.4 코드 변환 시스템 설계 및 구현

본 논문에서 구현하고자 하는 코드변환시스템은 다양한 프로그램 언어에 적용시킬 수 있으며 인라인 SQL 쿼리 구문을 매개변수화된 쿼리로 자동변환할 수 있도록 코드를 분석하여 변환하고자 한다.

기존 연구에서는 프로그램을 이용하여 취약한 부분을 알려주지만 코드변환은 개발자가 직접 수작업으로 할 수 있도록 도움을 주는 방식을 제안하고 있다.[4]

본 논문에서는 Java 프로그램을 이용하여 인라인 SQL 코드가 포함된 취약한 코드를 찾아내어 자동으로 코드를 변환하도록 처리하고자 하였다.

다양한 프로그래밍 언어를 지원하기 위해 공통적인 모듈을 인터페이스로 만들고 각각의 프로그램 언어별로 세부적으로 구현하는 방식으로 설계하였다.



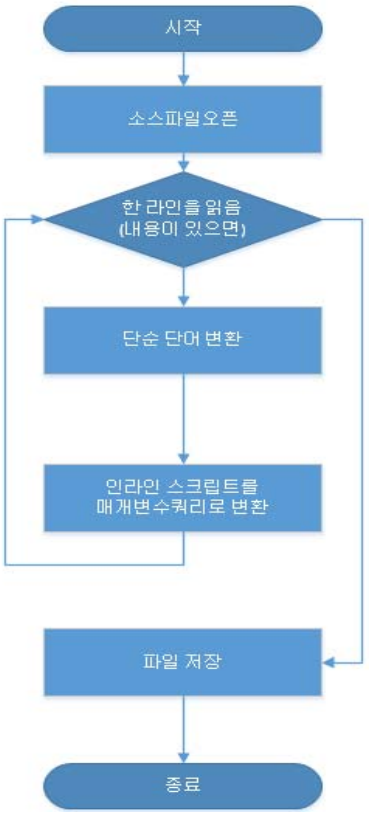
[그림 1] 클래스 다이어그램 설계

본 논문에서는 프로그램 코드의 다양한 패턴들을 모두 고려하여 처리하기 어려운 부분이 있으므로 아래와 같은 조건 하에서 테스트하였다.

첫째, 인라인 SQL 스크립트의 자료형은 기본 자료형 뿐 아니라 객체자료형도 모두 사용 가능하나 이 프로그램에서는 객체자료형을 제외한 기본자료형만으로 처리하였다.

둘째, 따옴표로 묶여진 인라인 SQL 스크립트를 한 라인에 한 개의 항목만 작성하도록 하였다.

코드 자동 변환 프로그램의 실행 과정은 [그림 2]와 같다.



[그림 2] 코드 자동 변환 프로그램의 실행 과정

코드 자동 변환 프로그램을 이용하여 변경 전의 코드를 분석하여 변경 후의 코드를 아래와 같이 생성하였다.

```
string sql = "select name from member
                where userid='"+userid+"' ";
sql +=" and passwd='"+passwd+"'";
SqlCommand cmd
  = new SqlCommand(sql, conn);
SqlDataReader dr = cmd.ExecuteReader();
```

[표1] 변경전의 인라인 SQL 스크립트 코드(C#.net)

```
string sql = "select name from member
                where userid=@userid ";
sql += " and passwd=@passwd";
SqlCommand cmd
  = new SqlCommand(sql, conn);
cmd.Parameter.AddWithValue(@userid,userid);
cmd.Parameter.AddWithValue(@passwd,passwd);
```

[표2] 변경 후의 코드(C#.net)

```
StringBuilder sql = new StringBuilder();
sql.append("select name from member");
sql.append(" where userid='" + userid + "' ");
sql.append(" and passwd='" + passwd + "' ");
stmt = conn.createStatement();
rs = stmt.executeQuery(sql.toString());
```

[표3] 변경 전의 인라인 SQL 스크립트 코드(Java)

```
StringBuilder sql = new StringBuilder();
sql.append("select name from member");
sql.append(" where userid=? ");
sql.append(" and passwd=? ");
stmt = conn.prepareStatement(sql.toString());
stmt.setString(1,  userid  );
stmt.setString(2,  passwd  );
```

[표4] 변경 후의 코드(Java)

3. 결론

본 논문에서는 SQL 삽입 공격에 대한 방어 대책의 일환으로 기존에 개발된 프로그램 코드를 분석하여 SQL 삽입 공격에 취약한 인라인 SQL 쿼리구문을 매개변수화된 쿼리로 변환함으로써 근본적인 해결을 시도하였다. 매개변수화된 쿼리는 따옴표 및 특수문자 등을 자동으로 변환처리하며 DBMS의 부하가 감소되고 처리속도가 빠른 장점을 가지고 잇다.

본 논문에서는 전세계적으로 많이 사용되고 있는 Java와 C#.net 프로그램 소스를 분석할 수 있도록 하였다. 차후 ASP, PHP, VB.net 등의 다양한 언어도 지원할 수 있도록 기능을 추가할 예정이다.

본 논문에서 제시한 코드 변환시스템은 다음과 같은 한계점 및 향후 연구과제를 갖는다.

첫째, 프로그램 코드의 다양한 패턴과 변수들에 대해 완전하게 자동화처리를 하지는 못했다.

둘째, Java, C#.net 이외의 다른 언어에서도 활용될 수 있도록 기능을 추가할 예정이다.

셋째, 최근 많이 사용되고 있는 스마트폰에 내장된 SQLite를 DB로 사용하는 경우에 대한 고려가 필요하다.

본 연구는 해양수산부의 지원으로 수행하고 있는 "수중 광역 이동통신 시스템 개발" 사업 결과의 일부임을 밝히며 지원에 감사드립니다.

참고문헌

[1] M.Amutha Prabakar, M.KarthiKeyan, K. Marimuthu, "AN EFFICIENT TECHNIQUE FOR PREVENTING SQL INJECTION ATTACK USING PATTERN", IEEE Computer Socity, 2013.5

[2] 김점구, 노시춘, "공격코드 사례분석을 기반으로 한 SQL Injection에 대한 단계적 대응모델 연구", 정보보안논문지, 2012.3

[3] 최성호, "Hacking Tool을 이용한 SQL Injection 공격에 대한 방어 방법", 서강대학교 정보통신대학원 석사논문, 2010

[4] 박주화, "자바 기반 웹 응용프로그램의 SQL Injection 취약점 대응방안에 대한 연구", 성균관대학교 정보통신대학원 석사학위논문, 2010

[5] JUSTIN CLARKE, "철통보안 SQL injection", Elsevier Inc. 2009

[6] Lwin Khin Shar, Hee Beng Kuan Tan, "Defeating SQL Injection", IEEE Computer Socity, 2013.3

[7] Debabrata Kar, Suvasini Panigrahi, "Prevention of SQL Injection Attack Using Query Transformation and Hashing", IEEE Computer Socity, 2012.3

[8] 이문주, "Mass SQL Injection에 대한 침입 탐지 및 대응 시스템 연구", 중앙대학교 정보대학원 석사학위논문, 2011

[9] 김병권, "SQL Injection 웹 취약점의 공격과 방어", 청주대학교 대학원 석사학위논문, 2010

[10] 방지호, 하란, "안전한 소프트웨어 개발을 위한 정적 분석 도구 시험코드 개발", 한국통신학회논문지, 2013.5

[11] 장희선, "Web Vulnerability Scanner를 이용한 취약점 분석", 융합보안 논문지, 2012.9