공격코드 사례분석을 기반으로 한 SQL Injection에 대한 단계적 대응모델 연구★

김점구* · 노시춘*

요 약

SQL Injection 기법은 공개된지 수년이 지났지만 웹해킹 공격중 가장 위험한 공격으로 분류되어 있다. 웹 프로그래밍은 자료의 효율적인 저장 및 검색을 위해 DBMS를 필수적으로 사용하고 있다. 주로 PHP,JSP,ASP 등의 스크립트 언어를 이용하여 DBMS와 연동한다. 이러한 웹 어플리케이션에서 클라이언트의 잘못된 입력값을 검증하지 않으므로 비정상적인 SQL 쿼리가 발생할 수 있다. 이러한 비정상적 쿼리는 사용자 인증을 우회하거나 데이터베이스에 저장된 데이터를 노출시킬 수 있다. 공격자는 SQL Injection 취약점을 이용하여 아이디와 암호를 몰라도 웹기반 인증을 통과할 수 있고 데이터베이스에 저장된 데이터를 열람해 볼 수 있다. SQL Injection에 대한 대책으로 다수의 방법이 발표되었다. 그러나 어느 한 가지 방법에 의존할 경우 많은 보안 공백이 발생할 수 있다. 단계적 대응모델은 사고 예방적 측면에서 소스코드 작성 단계, 서버 운용단계, 데이터베이스 핸드링 단계, 사용자 입력값 검증 활용 단계 등 대책을 프레임워크로 구성하여 적용하는 방법이다. 이 대응모델 을 적용할 경우 운용과정을 통해 존재하는 SQL Injection의 공격가능성을 보다 효과적으로 차단이 가능하다.

A Study of Step-by-step Countermeasures Model through Analysis of SQL Injection Attacks Code

Jeom goo Kim* · SiChoon Noh*

ABSTRACT

SQL Injection techniques disclosed web hacking years passed, but these are classified the most dangerous attacks. Recent web programming data for efficient storage and retrieval using a DBMS is essential. Mainly PHP, JSP, ASP, and scripting language used to interact with the DBMS. In this web environments application does not validate the client's invalid entry may cause abnormal SQL query. These unusual queries to bypass user authentication or data that is stored in the database can be exposed. SQL Injection vulnerability environment, an attacker can pass the web-based authentication using username and password and data stored in the database. Measures against SQL Injection on has been announced as a number of methods. But if you rely on any one method of many security hole can occur. The proposal of four levels leverage is composed with the source code, operational phases, database, server management side and the user input validation. This is a way to apply the measures in terms of why the accident preventive steps for creating a phased step-by-step response nodel, through the process of management measures, if applied, there is the possibility of SQL Injection attacks can be

Key words : SQL Injection, Attacks Code, Countermeasures, Model, OWASP

접수일(2012년 2월 24일), 수정일(1차: 2012년 3월 11일), 게재확정일(2012년 3월 19일)

★ 본 연구는 2011년 산학협동재단 연구비 지원으로 이루어졌음.

* 남서울대학교 컴퓨터학과

1. 서 론

최근 웹해킹(web hacking)의 특징은 다른 해킹 기법들에 비해 목표가 되는 시스템의 환경에 따른 영향을 크게 받지 않으며 쉽게 사용할 수 있고 공격에 성공할 경우 위력이 크다는 점이다. 웹해킹 중에서 SQL Injection 공격은 어려운 공격기법도 있지만 고난도 기술이 아니어서 쉽게 배울 수 있는 방법이 많다. 특히 공격을 자동화해놓은 툴이 많고 SQL Injection 도구들은 사용하기 쉬운 GUI방식의 도구가 많다. 이 툴을 이용 시 공격 대상 서버의 시스템 명령까지 내릴 수 있으며 데이터 조회와 수정도 가능하여 심각한 문제를 야기한다. 이와 같은 환경에서 웹해킹의 과제인 SQL Injection 공격에 대처하기 위해 SQL Injection 공격코드 분석과 단계적 대응방법 모델을 제안한다. 연구순서는 웹해킹 동향, SQL Injection 공격코드 분석, 단계적 대응방법, 결론의 순서이다.

2. 웹해킹 동향

웹(web)을 이용한 서비스 업무의 폭발적 증가 추세와 더불어 웹해킹을 통한 시스템의 데이터 탈취 또는 해킹사고가 급증하고 있다. Web상에서 작동하는 프로그램을 일반적으로 CGI (Common Gateway Interface)라 하는데 웹 게시판이나 쇼핑몰 등이 있으며 웹해킹 시 대부분 CGI를 목표로 한다. 인트라넷시스템에서는 방화벽을 이용하여 취약한 포트를 차단하지만 80번 포트는 방화벽이 차단하지 못하는 웹사이트이다. 이는 HTTP를 사용하는 웹사이트에서 해킹공격이 구조적으로 가능하게 되어있음을 의미한다. 해킹공격이 과거에는 OS나 응용프로그램 취약점을 노렸으나 최근에는 웹 프로그램 코드의 취약점을 이용하고 있다. 국제웹보안표준기구 OWASP는 해킹동향을 바탕으로 위험성이 높은 애플리케이션 취약점 중에서 TOP10을 3년 주기로 발표하고 있다. 2010년도 TOP10을 보면 Crss Site Scripting과 Injection Flaws가 1, 2위 순위이다. Crss Site Scripting은 특정 스크립트를 작성한 후 사용자가 데이터를 입력하면 스크립트가 시작되어 사용자의 쿠키값을 가로챈다, 가로챈 쿠키값을 웹 프록시 등을 이용하여 재전송 하고 공격자는 열람자의 정보로 로그인이 가능해진다. 2010 OWASP TOP10은 <표1>과 같다[1].

<표1> OWASP Top Ten Overview (2010)

A1: Injection
A2: Cross-Site Scripting (XSS)
A3: Broken Authentication and Session Management
A4: Insecure Direct Object References
A5: Cross-Site Request Forgery (CSRF)
A6: Security Misconfiguration
A7: Insecure Cryptographic Storage
A8: Failure to Restrict URL Access
A9: Insufficient Transport Layer Protection
A10: Unvalidated Redirects and Forwards

3. SQL Injection 메커니즘 분석

3.1 시스템 아키텍처 환경

SQL Injection을 유발시키는 웹시스템 구조는 다양하며 본 연구에 인용한 소프트웨어 구조는 윈도우 운영체제 환경에서 스크립트 코드(Script Code) 언어를 MS-SQL에 연동하여 DBMS를 핸들링하는 방법을 전제로 했다. 이 구조를 분석대상으로 선택한 것은 RDBMS를 연동하는 방법으로 일반적으로 가장 많이 사용하는 사례이기 때문이다.

<표2> SQL Injection 관련 소프트웨어 구조

기능	소프트웨어	연동아키텍처
운영체제	윈도우운영체제	WinSock
애플리케이션	COBOL,C, C++, Java, PL/I	스크립트 언어
Script Code	PHP, JSP, ASP	MS-SQL
DB핸들링	MS-SQL	RDBMS
클래스 라이브러리	MFC	라이브러리
웹서버	IIS, Apache	클라이언트

3.2 웹 프로그램 코드형식

웹시스템 환경에서 어떤 DB에 입력값을 입력하거나 삭제하려면 웹 프로그램을 사용해야 한다. 웹프로

그램은 ASP, JSP, PHP 등 스크립트 코드가 사용된다. 그중에서 ASP 프로그래밍 기본형식과 ASP를 이용한 MS-SQL에 데이터 입력 소스작성 기본형식을 설명하면 다음과 같다[3].

o ASP 프로그래밍 기본형식

ASP는 웹 시스템 구축에 사용되는 프로그램으로서 MS계열의 텍스트기반 스크립트 코드 이자 자동적 페이지 기법이다. 즉 사용자마다 페이지를 다르게 하여 사용자별 페이지를 보여주는 방식이다. 인터넷상에서 데이터를 저장하고 홈페이지에서 출력하는 형태라면 어떤 형태든 사용 가능하다. ASP 코드의 일반형식은 다음과 같다.

input1.asp

FORM문을 사용하여 (ID, PW)를 입력받고, '로그인' 버튼을 클릭 시 input2.asp를 호출한다. '취소' 버튼을 클릭 시 입력내용이 초기화 된다.

```
<TABLE width="320" border="2">
<FORM NAME="input1" METHOD=POST ACTION
="input2.asp">
```

input2.asp

input1.asp에서 입력받는 (ID, PW)를 화면에 출력한다. FORM문을 사용하여 입력값을 입력 받는다. '계산' 버튼 클릭 시 process.asp를 호출하고 '취소' 버튼 클릭 시 입력이 초기화 된다.

o ASP와 MS-SQL 연동 기본형식

SQL쿼리문은 데이터베이스 구축, 데이터 구축, 사용자 쿼리 명령어 3개영역에서 7가지 종류의 명령문으로 구성된다.

<표3> 기본적인 SQL문

구분	SQL문의 종류			합계
사용 용도	데이터베이스 구축	데이터 구축	사용자 쿼리	3
SQL 문	CREATE, ALTER, DROP	INSERT, UPDATE, DELETE	SELECT	7
합계	3	3	1	

ASP와 MS-SQL 연동 시 여러 방법이 있으며 ODBC, OLEDB, ADODB 로도 가능하다. DB연동 후 SQL 쿼리문을 불러오며 연결한 데이터베이스에서 해당 쿼리문으로 데이터를 가져오게 된다. ASP를 이용한 MS-SQL 연동 기본형식은 다음과 같다.

```
Set dbcon=Server.CreateObject("ADODB.Connection")
dbcon.Open strConnect
Set rs=Server.CreateObject ("ADODB. RecordSet")
SQL = " SELECT * "
SQL = SQL & " FROM xxxxx "
set rs = dbcon.Execute(SQL)
```

ASP를 이용한 MS-SQL 연동후 쿼리문을 작성할 경우 연결한 데이터베이스로부터 해당 쿼리문으로 데이터를 가져오게 된다. 데이터 입력 소스작성 사용 시 rs("name")를 사용하는데 rs는 불러온 값을 보관하는 기능이다.

```
sql="sql 쿼리문 작성"
set rs=server.CreateObject("adodb.recordset")
rs.Open SQL,db
```

3.3 SQL 인증공격 과정

스크립트 코드로 구성된 웹 애플리케이션에서 사용자로부터 SQL문을 입력받는 부분, 즉 데이터베이스와 연동되는 부분은 크게 로그인, 검색, 게시판으로 나눌 수 있다. 어느 사이트에서든 로그인을 하려면 아이디와 비밀번호를 입력해야 한다. 웹 애플리케이션 코드상에서 사용자 인증처리 모듈이 입력값을 검사하지 않을 때 공격자는 비정상적 SQL Query를 삽입 할 수 있고 이를 이용해 사용 중인 데이터베이스에 폐해를 준다. SQL Injection을 악용하는 공격자는 아이디와 비밀번호를 정상적 문자가 아닌 특정 SQL문을 넣어 공격한다. 아이디나 비밀번호 부분에 다른 SQL문이 삽입되면 그대로 데이터베이스로 전송되어 공격자가 원하는 비정상적 결과가 초래된다[4].

3.4 SQL Injection 코드 사례

SQL Injection 인증우회의 코드 사례를 들면 공격자는 비정상적인 SQL 질의를 웹서버의 로그인 입력

부분에 유추된 로그인 계정인 'abc'와 패스워드로 ''kor'=' 로 값을 입력한다. 실제처리 SQL 질의문 사례는 다음 형식이다[4]

```
SELECT* FROM members WHERE id='abc' AND passwd'' kor ''=''
```

kor 이하의 SQL 질의문이 항상 올바른 값으로 인식되므로 공격자는 패스워드를 모르는 상태에서도 관리자로 로그인이 가능하게 된다. SQL Query 상에서 사용자 인증 우회, MS SQL의 ;을 이용한 복수명령 실행, MS-SQL에서 윈도우 명령어 실행, ASP 코드상에서의 인증우회 사례를 보면 다음과 같다[5][6][7].

o SQL Query 상에서 사용자 인증 우회

패스워드를 알지 못하더라도 패스워드 체크부분을 주석처리 함으로서 아이디만을 가지고 로그인이 가능하다[7].

로그인을 위해 사용되는 SQL Query

```
SELECT * FROM member_table
WHERE user_id='abc' AND user_pw='6789';
```

--을 이용한 패스워드 인증 우회 (--은 주석을 뜻함)

```
SELECT * FROM member_table
WHERE user_id='abc'--' AND user_pw='6789';
```

o MS-SQL의 ;을 이용한 복수명령 실행

MS-SQL의 ; 문자는 Linux 및 Unix의 ; 과 동일하며 동시에 여러 명령어 실행이 가능하다.

- 데이터베이스 삭제

```
SELECT * FROM member_table
WHERE user_id='test' ; DROP DATABASE member_database --'
AND user_pw='6789';
```

- 관리자 패스워드 변경

```
SELECT * FROM member_table
WHERE user_id='test'
; UPDATE member SET user_pw='test'
WHERE user_id='abc'--' AND user_pw='6789';
```

o MS-SQL에서 윈도우 명령어 실행

MS-SQL의 내장 프로시저를 이용한 윈도우 명령어를 실행한다.

- dir 명령어 실행

```
SELECT * FROM member_table
WHERE user_id='test'
; Exec master..xp_cmdshell 'dir' --'
AND user_pw=PASSWORD('1234');
```

- net user 명령어 실행

```
SELECT * FROM member_table
WHERE user_id='test'
; Exec master..xp_cmdshell 'net user' --'
AND user_pw=PASSWORD('6789');
```

o ASP 코드상에서의 인증우회

PHP, JSP, ASP 등 웹 애플리케이션용 스크립트 언어는 DBMS와 연동시 만약 클라이언트에서 잘못된 입력 값이 있더라도 구조적으로 이를 검증하지 않으므로 비정상적 SQL 쿼리가 발생할 수 있다. ASP 코드 경우를 보면 웹상에서 아이디와 비밀번호를 입력시 ASP 코드에 아이디와 비밀번호가 입력된다. 공격자가 아이디와 비밀번호에 ' abs ''=' 방식으로 입력하면 다음결과가 나타난다. 아이디: ' abs ''=', 비밀번호: ' abs ''='

```
SELECT user_id FROM member WHERE
user_id = '' abs '' = '' AND password = '' abs '' = ''
```

이같은 쿼리문이 만들어지면서 조건문(Where문)은 항상 만족하고, 값도 항상 '참'이 되기 때문에 공격자는 사용자 인증에 성공하게 된다. 이 메커니즘을 이용하여 간단하게 로그인 창의 로그인을 우회하게 된다

[6][7][8].

4. 문제점 도출

웹해킹 동향, SQL Injection 메커니즘, SQL Injection 코드 사례를 분석해보면 SQL을 통하여 DB를 핸들링 운용 시 다음 문제점이 도출된다[8][9].

o SQL Injection 발생영역의 다양성

취약점 발생영역은 소스코드, DB, 입력값등 다수의 분야에 존재한다. 따라서 어떤 대책을 채택할지 판단의 어려움이 있다. 특정 방법 하나만 적용시 보안문제를 야기시킨다.

o SQL Injection 기술적 난이도 용이성

SQL Injection 코드 사례분석에 나타난 바와 같이 SQL injection은 기술적으로 어려운 기법도 있지만 고난이도 기술이 아니어서 쉽게 배울 수 있다. 해커에 의해 SQL Injection 공격이 초래될 가능성이 매우 높다는 것을 의미한다.

o 공격용 자동화 툴의 획득 용이성

SQL Injection 공격용 자동화 툴은 대부분 국외에서 제작되고 취약점 진단용으로 만들어져 서 공개된다. HDSI, NBSI, FG-Injector, SQL Power Injector 등 공개용 툴은 누구나 쉽게 사용이 가능하다 [8].

o 에러 메시지 출력의 위험성

MS-SQL은 SQL 에러발생시 DBMS 정보를 출력해 준다. 에러 메시지를 통해 데이터베이스, 테이블, 필드, 데이터까지 확인할 수 있다. 데이터베이스와 연동하여 데이터를 처리하는 모든 웹 페이지에서 이와 같은 위협이 존재한다.

o 입력값 검사 확인의 어려움

DB 입력 모든 변수를 확인하고 수작업으로 소스를 수정해야 하지만 모든 값을 검사 확인하는 방법은 현실적으로 매우 어렵다. SQL 공격이 공개된지 수 년이 지났지만 웹해킹 공격중 가장 위험한 공격으로 남아 있는 이유이다.

5. 대응방안 모델 정립

5.1 대응모델의 목적

대응방안 모델이란 SQL Injection 대책 적용 시한가지 기술이나 대책이 아닌 종합화 대책을 구성하고 그 틀속에서 정보보안을 단계적으로 실행하는 방법이다. 다수의 SQL Injection 대책이 있지만 어느 한가지 방법만 의존 시 많은 보안 공백을 발생 시킬 수 있다. 본 연구에서 제안하는 단계적 대응방안 모델은 대책의 구조를 사용단계, 개발단계, 운용단계 구분하여 발생 가능한 취약성에 대해 단계적인 대책을 적용한다.

5.2 모델의 프레임워크

프레임워크 구조는 사용단계, 개발단계, 시스템운용단계로 구분된다. 3개단계에서 각기 적용할 이행방안을 도출하여 구성하는데 시스템사용자 - 개발자 - 시스템운용자 단계로 시스템 흐름에 의거한 대응책을 발굴한다. 개발, 운용 절차를 정립하는 과정이다. 사용단계는 사용자시스템이용, 데이터 입력 시 이행 이며 개발단계는 사용자 요구사항 분석, 개발 시 보안수칙 준수이고 시스템 운용단계는 소스코드 운영, 웹서버 운용. SQL 서버 운용, 데이터베이스 핸들링 등 영역별로 보안이행 사항을 정립한다. 연구내용은 요구사항 분석, 보안방법론, 이행방법 평가의 순서로 기술한다.

<표4> SQL Injection 대응 프레임워크 구조

SQL Injection 대응 프레임워크			
시스템 단계	사용단계	개발단계	시스템 운용단계
이해 관계자	정보시스템 사용자 (user)	정보시스템 개발자 (developer)	정보시스템 운용자 (maintainer)
이행 사항	● 사용자시스템이용,데이타 입력시 이행사항	● 사용자 요구사항 분석 ● 개발시 보안수칙 준수	● 소스코드 운영 ● 웹서버 운용 ● SQL 서버 운용 ● DB핸드링

5.3 단계별 대응방안

5.3.1 시스템 개발단계

<표5> 이해관계자와 관심사항

이해관계자	정보 사용자 (user)	정보시스템 개발자 (developer)	정보시스템 운용자 (maintainer)
관심사항	● 로그인 및 사용자 입력값을 대상으로DB Query 정밀점검 ● 웹 애플리케이션과SQL 사용시 보안취약점 탐지 방법	● 사용자 요구사항 분석의 정확한 수행 ● SQL Injection 진단 시스템 개발	● 클라이언트 „웹서버, SQL 서버, 데이터베이스, 공통영역별 보안대책 여부 ● 웹해킹 예방을위한 일반보안이행

웹 애플리케이션과 SQL을 사용하여 DBMS를 운용하는 이해관계자는 정보 사용자(user), 정보시스템 개발자(developer), 정보시스템 유지보수자 (maintainer)로 나눌 수 있다. 요구사항 분석 단계에서는 이해관계자를 도출하여 이해관계자의 관심을 식별하여 그 결과는 개발단계에 반영한다.

SQL Injection 취약점은 입력값 검증 절차 문제에 기인하므로 개발단계에서부터 반드시 모든 입력값에 대해 적절한 검증절차를 설계하고 구현해야 한다. 검증절차는 입력값 크기, 특수문자 의 경우 위험하지 않은 문자로 치환 후 입력값이 허용범위 내에 존재하는지 검사하는 방식이다. SQL Injection 취약점 진단시스템을 사용하여 SQL Injection 분석을 통한 공격기법을 진단하고 데이터 형에 따른 injection 패턴을 조사한다. 설계단계에서는 변수명세, 인터페이스를 설계하여 검색엔진을 개발한다. 분석결과를 토대로 엔진을 설계하며 변수명세, 인터페이스를 설계한다. SQL 공격진단 엔진에 사용될 명세를 기반으로 함수를 작성한다.

5.3.2 시스템 운영단계

□ 클라이언트 소스코드 운영단계

o SQL코드 사례분석을 참고한 소스 차원 대책으로서 웹사이트 로그인 및 사용자 입력값 사용 코드를 대상으로 DB Query 생성방식을 점검한다.

o 소스코드에서 사용자 입력값에 대한 정당성 체크 루틴을 도입한다.

o 웹 애플리케이션 코드를 수정하여 사용자 입력값이 SQL문장으로 사용되지 않도록 한다. 사용자 입력값 존재 따옴표를 제거토록 설정한다.

o 개발과정에서 디버깅을 목적으로 에러가 반환되도록 허용하고 개발 종료 후 그 기능을 제거한다. 에러를 표시하기 보다 메인 페이지로 반환한다.

o 웹사이트에서 수행하는 기능 테이블에서는 select, insert, update, delete 중에서 한정된 기능만을 사용한다.

o 입력받은 변수와 DB 필드의 데이터형을 일치 시켜야 하고, 사용 중인 SQL 구문을 변경시킬 수 있는 특수문자가 포함되어 있는지 체크한다.

o 스크립트 코드는 취약한 SQL 예제와 안전한 SQL 예제 사례를 준비하여 적용한다.

□ 웹서버 운용단계

o IIS, Apache 등 웹서버에서 관련 오류나 경고이벤트 및 IIS 로그를 주기적으로 체크하는 과정이 긴요하다

o 웹사이트 오류 발생 시 클라이언트와 마찬가지로 웹서버에서도 오류 발생 메시지를 표시하지 않도록 한다. 웹서버의 오류메시지를 수정해서 특정 페이지로 리다이렉션(redirection) 처리 한다.

.o 사용자로부터 입력받은 변수로 SQL 쿼리 구문을 생성하는 CGI는 입력받은 변수를 체크하거나 변경하는 로직을 포함하고 있어야 한다.

o 개발과정에서 디버깅을 목적으로 에러가 리턴되도록 허용하나 개발 종료 시 그 기능을 제거해야 한다. 에러발생 시 에러 발생 사실을 표시하기 보다 메인 페이지로 반환한다. 기타 원칙은 클라이언트 소스코드 단계와 동일하다.

□ SQL 서버 운용단계

o 공격자는 리턴되는 에러 메시지에 대한 역공학 분석을 통하여 공격 가능한 SQL Injection 스트링을 알아낼 수 있다. 따라서 SQL 서버의 에러 메시지

를 디스플레이 하지 않도록 한다.

o 특히, 사용이 허용된 문자들의 조합으로 공격 스트링을 만들 수 있기 때문에 최소 사용가능 안전성 문자집합을 사용해야 한다. 최소 사용가능 문자집합에는 가급적 숫자만을 포함하도록 하고 어렵다면 숫자와 문자만를 혼용한다.

o 만약 사용자 입력으로 기호문자나 구두문자 같은 문자들을 사용해야 한다면 꼭 필요한 최소의 문자만을 포함하고 허용된 문자를 HTML 문자로 변환해서 처리하도록 한다.

o SQL 서버 외부는 항상 노출되어 있기 때문에 SQL 서버에 반드시 잠금 장치를 해야한다.

□ 데이터베이스 핸들링 공통영역

o 데이터베이스와 연동되는 스크립트의 모든 파라미터를 필수적 점검하여 사용자 입력값이 SQL Injection을 발생시키지 않도록 수정한다.

o 일반유저 권한으로 모든 system stored procedures에 접근을 방지한다. SQL Injection 취약점을 이용하여 데이터베이스 전체에 대한 제어권을 얻거나 데이터베이스 운용서버 접근 불가토록 한다.

o DB 연동 웹사이트에서는 최소화된 권한만을 가진 해당 DB에 권한이 있는 계정을 이용 한다.
데이터베이스와 다른 백엔드 시스템 접속 시 최소한 권한을 강제화한다. 공격자에게 유용한 상세 에러 메세지들을 회피한다.

o DB의 테이블 목록을 정기적으로 점검한다. 해킹된 DB의 경우 주로 비정상적인 명칭의 테이블이 생성된다. 테이블 스키마나, 레코드를 보면 테이블 이름과 비슷한 내용이 들어 있는데 주로 시스템디렉토리 목록이나 웹사이트 디렉토리 목록, 레지스트리 값 조회이다[7][8].

o 데이터베이스 연동 부분에서 동적 SQL 사용을 금지하고 저장 프로시저를 사용한다. 저장 프로시저를 통해 데이터베이스 연동을 구현한다면 이미 프로시저 내부에서 입력값에 대한 자료형 검증이 이루어진다[9].

□ 사용자 입력값 검증 공통영역

o 입력값 검증범위는 나타나거나 저장된 데이터를 받아들이기 전에 길이, 유형, 구성, 비즈니스 규칙에 대한 모든 데이터이다. 검증을 위해 표준검증 메커니즘을 적용한다.

o 알려진 공격패턴을 기반으로 사용자입력 사용이 불가능한 스트링을 결정한다. 새로운 공격에 대비해서 사용자 입력으로 사용가능한 최소의 문자 집합을 결정한다.

o 따옴표에 대한 검증 수행 다음 사용자 입력으로 사용이 불가능한 스트링을 제거하고, 사용 가능한 문자집합에 포함되지 않은 모든 문자들을 제거하도록 설정한다.

o 사용자 입력이 SQL Injection을 발생시키지 않도록 사용자 입력 시 특수문자(′, ;, ″, \, Space, --, + 등) 포함 여부를 체크한다. 허용되지 않은 문자열, 문자 포함 시 에러 처리한다.

o 검색어 필드 및 로그인ID, PASSWD 필드에 큰 따옴표(″), 작은 따옴표(′), 세미콜론(;) 등을 입력후 DB 에러 발생을 확인한다[5][7][8].

□ 웹해킹 예방을 위한 일반보안 영역

대책은 이상과 같은 영역별 대책 외에도 SQL Injection 예방을 위해서는 보안 공백이 발생치 않도록 최신 패치나 서비스 팩을 사용하고 자동화 Tool을 사용하여 운영체제 및 어플리케이션 보안 취약점을 주기적으로 진단해야 하는 대응책이 필수적이다. 불필요한 윈도우 서비스들을 설정하지 않으며 이용하지 않는 계정을 삭제하고 Guest 계정은 항상 disable로 설정한다. 불필요한 공유를 없애고, 파일 공유 생성 시 권한에 유의한다. 윈도우 방화벽에서 원격터미널 서비스로 접속 IP를 지정하여 인가된 IP에게만 접근을 허용한다[10][11].

5.4 대응모델 종합

종합적 대응방안은 소스코드 작성단계와 웹서버 운용단계, SQL 서버 관리단계, 데이터베이스 핸드링 공통단계, 그리고 사용자 입력값 검증 공통활용 단계 5개단계로 설정한다. 대응방안을 단계적으로 적용하여 존재하는 SQL Injection의 공격가능성을 보다 철저히 차단해야한다. 단계적 대응단계, 대상영역, 목적, 대

상자원은 <표6>과 같다.

o 클라이언트 소스코드 작성단계

SQL Injection 대책으로 포괄적 입력검증을 수행하는 영역이다. 스크립트에 존재하는 모든 변수를 점검하여 사용자 입력값이 SQL Injection을 발생하지 않도록 수정하므로 안전장치로서 필수 적용사항 이다.

o 웹서버 운용 단계

IIS, Apache 등 웹서버를 구성하고 운용하는 측면에서의 보완조치사항 이다. 근본적 문제 해결을 위해서는 프로그램 보완 조치가 반드시 필요하지만, 웹서버의 보안 강화 설정을 통해서도 보완적인 효과를 볼 수 있다.

<표6> SQL Injection 대응방안 영역

대응단계	스크립트 코드 단계	웹서버 운용 단계	SQL 서버 운용 단계	DB 핸드링 공통	입력값 검증 공통	웹해킹일반대책
대상영역	소스코드	웹서버	SQL 서버	DB 공통	입력값공동	전체영역
목적	코드 안전성검증	코드 안전성검증	SQL 서버 안전관리	DB 운용안전성	입력값 취약성방지	웹해킹대응
대상자원	애플리케이션	웹서버	SQL 서버	SQL,DBMS	ID,PW, 입력값	전체자원

o SQL 서버 운용 단계

SQL 서버 운용 단계에서 SQL Injection 차단 대책을 강구하는 과정이다. 데이터베이스를 핸들링 하는 과정에서의 안전장치 이므로 매우 긴요한 이행사항이 다수 존재한다.

o 데이터베이스 핸들링 공통

어느 단계이건 데이터베이스 유저의 권한을 제한시켜야 하는 단계이다. 데이터베이스는 많은 분량의 정보가 실시간 관리되는 영역으로서 SQL Injection이 직접 발생치 않더라도 예방적 차원의 운용수칙이 적용되어야 한다.

o 사용자 입력값 검증 공통

어느 단계든지 값 입력 수행구간에 적용 수칙이며 특히 클라이언트코드 분야이다. 잠재적 악성 데이터를 검사, 삭제하는 과정이므로 업무현장에서는 보안이행에 가장 많은 노력이 필요하다.

6. 결 론

SQL Injection은 공개된 지 수 년이 지났지만 웹해킹 공격 중 가장 위험한 공격으로 분류되어 있다. SQL Injection에 대한 대책으로 다수의 방법이 발표되었으나 어느 한 가지 방법만 의존할 경우 많은 보안 공백이 발생할 수 있다. 단계적 대응모델 제안이유는 소스코드 작성 단계, 운용단계, 데이터베이스 핸들링 단계, 서버관리 단계 ,사용자 입력값 검증활용 단계 등 5개 단계의 대책을 프레임워크로 구성하여 적용하는 방법이다. 단계적 대응모델을 적용할 경우 운용과정에서 존재하는 SQL Injection의 공격 가능성을 보다 효율적으로 차단이 가능하다. 본 연구에서 제안된 방법을 시스템 운용현장에서 참고하여 SQL Injection 대처할 수 있기를 기대한다.

참고문헌

[1] OWASP, CSRF Guard, http://www.owasp.org/index.php/CSRF_Guard

[2] David Gourley and Brian Totty, “HTTP: The Definitive Guide”, O'Reilly Media, 2002.

[3] http://www.owasp.org/index.php/Cross-Site_Request_Forgery

[4] 이미정,노시춘, SQL Injection 취약점 진단 프로그램,2005.6

[5] Stepen Cost, An Introduction to SQL Injection Attacks,for Oracle develops, 2007.3

[6] http://redsea23.egloos.com/243019 SQL Injection 공격과 방어 방법

[7] 박상옥, 웹 관리자를 위한 응급처치법-SQL Injection 해킹 보안,2011.11

[8] http://www.krcert.or.kr/unim

[9] http://www.krcert.or.kr/index.jsp
[10] http://www.superuser.biz/tag/sql
[11] http://support.oullim.co.kr/portal/Tec hletter/20080615/news4.htm
[12] http://dev.mysql.com/downloads/gui-tools/5.0.html
[13] http://kline03.egloos.com/445826
[14] http://www.google.co.kr/imgres?imgurl=http://blog.outsider.ne.kr/attach/1/1154314780
[15] http://database.sarang.net/database/postgres/tutorial/lecture/c89.htm

[저 자 소 개]

김 점 구 (Jeom goo Kim)

1990년 2월 광운대학교 전자계산학과 이학사
1997년 8월 광운대학교 전자계산학과 석사
2000년 8월 한남대학교 컴퓨터공학 박사
1999년 3월~ 현재 남서울대학교 컴퓨터학과 정교수
IT융합연구소장

email : jgoo@nsu.ac.kr

노 시 춘 (SiChoon Noh)

1987년 2월 고려대학교 경영정보학 석사
2005년 2월 경기대학교 정보보호기술 박사
2002년 11월 KT 시스템보안부장
2004년 12월 KT 충청전산국장
2005년3월 ~ 현재 남서울대학교 컴퓨터학과 교수
IT융합연구소연구위원

email : nsc321@nsu.ac.kr