보안 코딩 지침의 사례

신성윤O, 이상원*, 이현창*
O군산대학교 컴퓨터정보통신공학부
*원광대학교 정보전자상거래학부
e-mail:s3397220@kunsan.ac.krO, {sangwonlee, hclglory}@wku.ac.kr*

Case of Security Coding Guide

Seong-Yoon Shin*O, Sang-Won Lee**, Hyun-Chang Lee**
*OSchool of Computer & Information Communication Engineering,
Kunsan National University
****Division Computer and Electronic Commerce(Institure of Convergence and Creativity),
Wonkwang University

● Abstract ●

본 논문에서는 S/W 개발 보안 지침의 사례로서 SQL 삽입에 대하여 설명한다. SQL 삽입은 입력 데이터 검증 및 표현에서 S/W 취약점 유형의 하나이다. 본 논문에서는 SQL 삽입에서 취약점 설명, 취약점 개념도, 보안 대책, 그리고 코드 예제까지 설명하도록 한다.

키워드: SQL 삽입(SQL injection), 취약점(vulnerability), 보안 대책(security policy)

I. Introduction

시큐어 코딩은 SW 개발과정에서 개발자 실수, 논리적 오류 등으로 인해 SW에 내포될 수 있는 보안취약점의 원인, 즉 보안약점을 최소화하는 한편, 사이버 보안위협에 대응할 수 있는 일련의 보안활동을 의미한다[1].

MS-SQL 데이터베이스의 환경과 C++ 환경을 이용하여 데이터베이스 사용자 계정 취약점을 판별한 논문[2]도 있었다. 또한 [2]에서는 MS-SQL 데이터베이스에서 패스워드나 사용자 설정이 변경되었는지를 조사한다. 그리고 사용자 계정이 만료되었거나 오랫동안 패스워드를 변경하지 않았으면 보안 취약점이 존재하는 것으로 판단하는데 이는 악의적인 제3의 사용자가 해킹 등을 하는 것을 예방하고 방지하기 위해서이다.

II. SQL Injection Vulnerability Explain

공격자가 입력한 데이터에 대한 유효성을 점검하지 않아 DB 쿼리로 직이 변경되어 공격자의 의도대로 중요 정보유출(계정정보 등) 또는 DB의 변경을 가하는 공격으로, 외부에서 입력 받은 데이터를 SQL 명령의 일부 또는 전체로 사용하는 프로그램이 부적절한 SQL 명령을 수정할 수 있도록 특수문자 등을 적절히 제거하지 못하는 경우 발생한다.

III. SQL Injection Vulnerability Diagram

다음 그림은 SQL Injection 공격에 대한 개념도로서 공격자는 패스워드를 입력하는 필드에 SQL Injection을 유발하는 명령어 `xx` OR 1=1— 또는 `xx` OR 1=1를 입력하여 항상 참이 되도록 만들어 데이터베이스에 대한 불법적인 접근을 시도하고 있다.

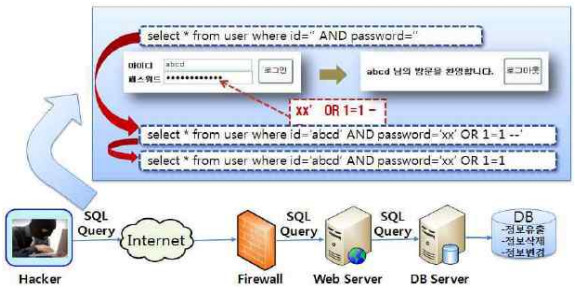

Fig. 1 Vulnerability Diagram

IV. Security Policy

외부 입력이나 외부 변수로부터 받은 값이 직접 SQL 함수의 인자로 전달되거나 문자열 복사를 통하여 전달되는 것은 위험하므로 인자화된 질의문을 사용한다.

자바의 경우에는 preparedStatement 클래스와 하위 메소드 executeQuery(), execute(), executeUpdate()를 사용하는 것이 바람직하다.

IV. Case of Code

SQL 삽입 공격은 데이터베이스 내에 저장되어 있는 정보를 조작하기 위한 SQL Query문에 논리적 완전성 문제를 악용하는 것으로, 외부 입력에 대한 적절성 검증을 수행하지 않을 때 주로 발생한다. 아래 예는 SQL 삽입 공격에 대한 예제를 보여주고 있다.

1. 안전하지 않은 코드의 예(JAVA)

```
1: try
2: String tableName= props.getProperty("jdbc.tableName");
3: String name = props.getProperty("jdbc.name");
4: String query = "SELECT * FROM " + tableName
+ " WHERE Name =" + name;
5: stmt = con.prepareStatement(query);
6: rs= stmt.executeQuery();
7: --
8: catch (SQLExceptionsqle)
9: finally
```

외부 입력으로부터 tableName과 name을 받아서 SQ L 질의문을 생성하고 있고, name의 값으로 "name' OR 'a'='a" 추가하여 문자열의 전달이 가능하다.

2. 안전한 코드의 예(JAVA)

```
1: try

2: String tableName= props.getProperty("jdbc.tableName");
3: String name = props.getProperty("jdbc.name");
4: String query = "SELECT * FROM ? WHERE Name = ? " ;
5: stmt = con.prepareStatement(query);
6: stmt.setString(1, tableName);
7: stmt.setString(2, name);
8: rs= stmt.executeQuery();
9: --
10: catch (SQLExceptionsqle)
11: finally
```

위의 예제와 같이 인자를 받는 PreparedStatement 객체를 상수 스트링으로 생성하고 인자 부분을 setXXX() 메소드로 설정하여, 외부의입력이 질의문의 구조를 바꾸는 것을 방지한다.

IV. Conclusions

본 논문에서는 S/W 개발 보안 지침의 실질적인 예로서 SQL 삽입에 대하여 설명하였다. SQL Injection(삽입)은 입력 데이터 검증 및 표현에 대한 소프트웨어 취약점 의 한 가지 유형이다. 논문에서는 SQL 삽입에서 취약점을 설명하였고, 취약점의 개념을 그림으로 표현하였으며, 여러 가지 보안 대책을 제시하였고, 마지막으로 안전한 코드와 안전하지 않은 코드의 예까지 들어주었다.

References

[1] http://blog.naver.com/neos_rtos/220164190569

[2] Jang Seung-Ju,"Implementation of User Account Vulnerability Checking Function System using MS-SQL Database," J. of KIICE, Vol. 18, No. 10, pp. 2482-2488, 2014