jinyPHP 2권 - 027. 인젝션

**27. 인젝션**

====================

SQL 인젝션이란 정상적인 SQL 명령문안에 잘못된 문장을 삽입하여 SQL코드를 변경하여 데이터베이스를 파괴하거나 해킹 공격을 시도하는 것을 말합니다.

**27.1 웹 SQL 공격**

====================

데이터베이스와 연동되는 동적 웹사이트는 사용자의 입력 데이터를 기반으로 SQL 명령 쿼리를 생성하고 결과를 화면에 출력을 합니다. 웹 인젝션은 악의 있는 ​​사용자가 웹 페이지의 입력값을 통하여 SQL 명령 코드를 변경하는 해킹입니다.

악의적인 사용자는 SQL 쿼리 생성에 영향을 줄 수 있는 기호나 값을 함께 입력함으로서 정상적인 프로그램 처리를 방해하고, 악의 적인 결과를 출력하도록 유도합니다. 이렇게 삽입된 SQL 명령은 SQL 문을 변경하고 웹 응용 프로그램의 보안을 손상시킬 수 있습니다.

**$ID = $\_POST[‘uid’];**

**$queryString = “select \* from members where Id = ‘$ID’”;**

위의 코드는 POST 값으로 넘어온 사용자 입력을 기준으로 SQL 쿼리문을 생성하는 예입니다. 하지만 이러한 코드는 SQL 인젝션 공격의 잠재적인 위험을 가지고 있습니다.

**27.2 참조건 인젝션**

====================

대부분의 쿼리는 WHERE 절 구문을 통하여 데이터를 필터링 합니다. 하지만 악의적인 입력값은 이러한 WHERE 조건절을 항상 참인 상태로 조작합니다.

쿼리 조건문 1=1 는 항상 참을 의미합니다. 어떤 조건문에도 or 1=1 을 추하가면 모든 쿼리 조건은 참으로 변경됩니다.

만일 위의 예제에서 $ID 값이 100’ or 1=1 이라는 문자열을 입력한다면, 쿼리는 다음과 같이 바뀔수 있습니다.

**select FirstName, LastName, Password from members where Id = ‘100’ or 1=1**

이 조건은 모두 참으로 동작합니다. 이런 경우 모든 회원의 정보를 해커가 확인할 수 있는 심각한 보안 유출이 될 수 있습니다.

**""=""** 참 조건도 자주 사용되는 인젝션 패턴입니다.

만일 회원로그인 처리를 위해서 사용자 이름과 패스워드를 받아 처리한다고 할 때

$queryString = 'SELECT \* FROM members WHERE Name ="' + 아이디 + '" AND Pass ="' + 비밀번호 + '"' 와 같이 쿼리문을 작성할 수 있을 것입니다. 하지만 인젝션 해커가 사용자 아이디와 패스워드에 **or ""="**" 라고 입력을 한다면 쿼리문은

SELECT \* FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""="" 로 바뀌어 처리될 것입니다. 위 조건에서 or ""="" 는 항상 참 조건으로 그냥 회원 로그인 처리가 될 것입니다.

**27.3 세미콜론 인젝션**

====================

SQL 명령문은 구분자인 세미콜론(;)으로 각각의 명령이 나누어 집니다.

회원조회를 위해서 SELECT \* FROM members WHERE UserId = $ID; 로 작성을 했다고 합시다 만일, 인젝션 해커가 **100; drop tables orders** 라고 입력을 한다면 쿼리문은

SELECT \* FROM members WHERE UserId = 100; drop tables orders; 로 변경이 될것입니다.

기존 쿼리에 새로운 쿼리를 추가하여 2개의 쿼리를 실행할 수 있습니다.

**27.4 주석처리**

====================

SQL의 주석처리문 -- 를 이용하여 인젝션 공격을 하는 유형도 있습니다.

회원조회를 위해서 SELECT \* FROM members WHERE UserId = $ID; 로 작성을 했습니다.

만일, 사용자 가 입력값을 **100 or 1=1 --** 로 입력을 한다면, -- 뒤의 쿼리 문장은 주석으로 처리가 될것입니다.

**27.4 보안방법**

====================

악의적인 값의 입력으로 인하여 인젝션 공격을 방지하는 방법은 매개변수를 필터링하는 것입니다. 또는 검증 함수들을 이용하여 처리하는 것 도한 좋은 방법이 될 것입니다.

**27.4.1 숫자확인**

====================

숫자의 값으로 입력되는 매개변수들은 값을 사전에 체크해 주는 것이 좋습니다. 또는 직접적인 SQL 문장을 만들어 쓰는 것보다

$query = sprintf(“select \* from members where Id=’%d’”,$ID);

형태로 쿼리의 매개변수의 인자 값이 정수로만 들어갈 수 있는 형태의 함수형태의 문장으로 바꾸어 사용하는 방법도 있습니다.

**27.4.2 문자열**

====================

사용자가 문자열 입력할때 매개변수 ($\_POST, $\_GET)의 모든 값을 addslashes() 함수를 이용하여 먼저 처리합니다.

addslashes() 함수는 DB 쿼리와 같이 인용부분앞에 역슬래쉬를 추가함으로써 SQL 인젝션 공격을 방지하는 용도로 자주 사용합니다. addslashes()는 PHP 버전 3이상부터 사용이 가능합니다.