SQL_INJECTION

한국디지털미디어고등학교 강현우

<목차>

C. Mysql에서 table 정보를 가져오는 방법

D. Mysql에서 column 정보를 가져오는 방법

5. Blind sql injection이란

A. Substr함수를 이용한 blind sql injection

B. mid함수를 이용한 blind sql injection

C. left, right함수를 이용한 blind sql injection

<SQL 이란>

D. <웹에 관한 이해>

현대인들은 하루라도 인터넷이 없다면 생활에 큰 지장을 겪을 것이다. 이 인터넷을 사용하여 우리는 여러가지 종류의 웹사이트에 접속을 하게 된다. 지금부터 우리가 사용하는 웹사이트의 구성에 대해 간단하게 알아볼 것이다.

우선 웹은 프론트엔드, 백엔드, 데이터베이스로 이루어져 있다. 지금부터 이 세 가지를 하나하나 알아 볼 것이다. 밑에 있는 그림처럼 웹은 프론트엔드, 백엔드, 데이터베이스가 긴밀하게 연결되어있다.

프론트엔드(front-end)

개념 : 프론트엔드란 사용자(클라이언트)가 웹에 처음 로그인 했을 때 보게 되는 부분을 이야기한다. 예를 들어 우리가 학교 홈페이지에 로그인을 할 때 우리가 어떤 일을 하면서 접하게 되는 부분들을 이야기한다.

사용 언어 : 프론트엔드를 구현할 때 사용하는 언어는 html, javascript, css, vue.js, react.js등 이 있다.

백엔드(back-end)

개념 : 사용자가 웹사이트를 사용할 때 보는 부분이 프론트엔드라면 실질적인 기능을 수행하는 부분은 백엔드이다. 백엔드에서는 우리가 원하는 기능을 수행하도록 짜여진 서버가 있고, 데이터베이스와 연결되어 있다. 데이터베이스와 연결된 서버는 데이터베이스에 값을 넣거나, 값을 수정하는 등의 역할을 수행한다.

사용 언어 : 백엔드를 구현할 때 사용하는 언어는 java(spring), php, node.js, python(Django, flask) 등이 있다.

데이터베이스(database)

개념 : 데이터베이스는 말 그대로 입력받은 데이터들을 저장하는 역할을 수행한다. 우리가 정보를 입력해야 하는 웹사이트를 만들었을 때 그 정보들을 일일히 파일입출력을 수행하기 어렵기 때문에 데이터베이스를 사용한다. 또한 데이터베이스는 우리가 입력한 정보들의 관리를 용이하게 해준다.

사용 언어 : 데이터베이스를 만들 때 사용하는 언어로는 mysql, mssql, sqlite, mongodb, mariadb 등이 있다.

E. <Sql 구조에 관한 이해>

Sql은 structured query language의 약자이다. 해석해 보자면 구조화된 쿼리 언어정도 일 것이다. Sql은 데이터베이스에서 여러가지 정보를 처리할 때 사용된다. 그런데 만약 정보들이 아무런 패턴 없이 나열되어 있다면 우리가 원하는 정보를 가져오거나 처리하기 어려울 것이다. 따라서 데이터베이스는 저장된 정보를 테이블이라고 하는 집합으로 관리한다. 이 테이블은 칼럼들의 집합이다. 표의 가장 위의 제목 같은 부분을 우리는 필드 또는 칼럼이라고 하고, 여러 칼럼들의 집합으로 이루어진 한 가로줄을 레이블이라고 한다.

F. <sql 구문에 관한 이해>

밑에 보이는 것 같은 테이블이 존재한다고 생각해 보자. 테이블의 이름은 user이고 id, name, password라고 하는 칼럼들이 존재할 때 우리가 어떤 방식으로 sql구문을 사용할 수 있는지 알아보자.

SELECT name FROM user WHERE id='{$_GET[id]}' and password='{$_GET[password]}'

우리가 이와 같은 sql구문을 작성했다고 가정했을 때 어떤 역할을 수행하는지 알아보도록 하자. 우선 우리는 'SELECT' 구문을 사용하여 우리가 가져오고 싶은 칼럼을 선택했다. 이 구문에서는 'SELECT name'을 했기 때문에 우리가 name을 가져오겠다는 것을 표현한 것이다. 그리고 'FROM user'은 우리가 가지고 오길 원하는 칼럼이 들어있는 테이블을 표현한 것이다. 그리고 'WHERE'절은 우리가 가져올 정보의 조건을 한정해 주는 것이다. 즉, 'WHERE id='$_GET[id]' and password='$_GET[password]''은 get방식으로 받아온 id와 password를 가지고 있는 칼럼을 가지고 오라는 의미이다. 즉, WHERE절은 우리가 가져올 정보에 조건을 한정해 주는 것이다. 정리해보면, 위의 sql구문이 의미하는 것은 user라는 이름의 테이블에서 id와 password가 우리가 get방식으로 입력한 id와 password와 일치하는 정보를 가진 name을 가져오라는 의미가 된다.

예를 들자면 우리가 id와 password에 '1'과 '1237'을 넣는다면 '김승우'를 가져오는 sql구문이 되는 것이다.

<Sql injection이란>

A. <Sql injection이란>

Sql injection을 번역해 본다면 sql 삽입이다. 이 공격은 말 그대로 sql구문을 삽입하는 공격이다. 사용자가 입력을 할 수 있는 부분을 보게 되면 sql구문과 연결이 되어 있는 부분들이 많다. 예를 들자면 로그인 페이지나 검색 페이지 등이 있을 것이다. 이러한 입력 페이지는 sql구문과 긴밀하게 연결이 되어있다. 그리고 이러한 입력 페이지에 sql구문을 입력하여 프로그래머가 의도하지 않은 동작을 실행시키는 것을 sql injection이라고 한다.

B. <Sql injection의 종류와 할 수 있는 일>

Sql injection의 종류는 error based sql injection, blind sql injection, error based blind sql injection, union based sql injection, Boolean sql injection 등이 있다. 이 문서에서 다룰 sql injection은 error based sql injection, blind sql injection, union based sql injection을 다룰 것이다. 우선 Error Based Sql injection은 입력 페이지에 sql구문 상에서 발생하는 오류를 숨기지 않았을 때 그 오류를 기반으로 공격을 하는 것을 이야기한다. Error based sql injection은 공격이 간편하다는 장점이 있다. Blind sql injection은 우리가 비밀번호 같은 문자열을 찾아야 할 필요가 있을 때 사용한다. 문자열을 자를 수 있는 함수들을 사용하여 우리가 원하는 칼럼에 담긴 값을 찾는 것이다.

Sql injection이 가능하기만 하다면 우리는 웹사이트에 여러가지 일들을 수행할 수 있다. 우리는 데이터베이스를 속여 웹사이트에 계정을 생성하지 않고 로그인 할 수 있고, 더 나아가 페이지 관리자(admin)의 권한을 탈취할 수 있다. 또한 여러가지 sql구문을 사용하여 데이터베이스에 존재하는 정보들을 탈취할 수 있다.

C. <기초적인 sql injection과 sql injection의 원리>

Sql injection을 시작하기 전 기본적으로 알아야 하는 지식이 있다. 우리는 sql injection공격을 수행하기 전 서버에 어떤 식으로 데이터를 전송하는지 알아야 한다. 여러 가지의 다양한 데이터 전송방식이 존재하지만, 우리가 알아야 할 핵심적이 전송방식은 get방식과 post방식 두 가지이다. 먼저 get방식은 url에 우리가 보낼 값을 변수에 담아서 보내는 방식을 의미한다. 예를 들어 우리가 get방식으로 id라는 변수에 1이라는 값을 보내서 전송하고 싶다고 했을 때 url에 ?를 붙이고 id=1이라고 작성하면 된다. 다음으로는 post방식이다. post방식은 get방식처럼 url에 우리가 보내는 값이 나타나지는 않는다. 다만 패킷에 값을 담아서 전송하는 방식을 의미한다. post방식으로 값을 전송할 때는 입력 페이지에 값을 넘기면 post방식으로 값이 넘어가게 된다. 이 두 가지 전송방식에 대한 이해가 선행되어야 sql injection공격 적절한 방식으로 수행할 수 있다.

기초적인 sql injection을 이해하기 위해 위에서 사용했던 쿼리를 다시 한번 보자

SELECT name FROM user WHERE id='{$_GET[id]}' and password='{$_GET[password]}'

이 쿼리를 보게 된다면 get방식으로 id와 password를 입력 받는 것을 알 수 있을 것이다. 그런데 우리가 id에는 세미콜론(')을 넘기고 password에 임의의 값을 넘기면 어떤 일이 발생할 지 생각해 보자

SELECT name FROM user WHERE id='' and password='1234'

이러한 쿼리문이 만들어 질 것이다. 이 쿼리문을 실행시키면 무슨 일이 일어날까? 이 쿼리문을 실행시키게 되면 에러가 발생한다. 에러는 **WHERE id=''** 부분에서 발생하는데 이유는 **id=''** 부분에서 id에 들어갈 문자열을 의미하는 세미콜론이 우리가 입력한 세미콜론에 의해 닫히게 되고 뒤의 세미콜론이 남게 되어 오류를 발생시킨다. 그럼 조금 더 생각을 해 보도록 하자. 만약 우리가 id에 **' or 1=1%23** 이라는 값을 넘겨 주었고, password에는 1234라는 값을 주었다고 가정하자. 다시 쿼리문이 어떤 식으로 구성될지 살펴보도록 하자.

SELECT name FROM user WHERE id='' or 1=1%23' and password='1234'

라는 형태의 쿼리가 만들어질 것이다. 쿼리를 살펴보면 아무것도 들어있지 않은 상태의 id변수와 1=1이라는 항상 참인 구문을 or연산을 실행시킨다. 그리고 뒤에서 다룰 한 줄 주석(%23)을 이용하여 뒤의 쿼리문을 없는 것으로 만들어 준다.

다시 한 줄 주석 처리된 쿼리문을 살펴보자.

SELECT name FROM user WHERE id=" or 1=1%23' ~~and password='$_GET[password]'~~

Id=" or 1=1구문에 집중하여 살펴본다면 우선 우리는 id에 아무런 값을 넣지 않았다. 따라서 데이터베이스에 들어있는 어떤 값도 우리가 넣은 값과 같은 값이 없을 것이므로 id=''구문은 항상 거짓임을 유추해 볼 수 있다 그리고 1=1이라는 항상 참인 구문과 or 연산을 진행하게 되면 id에는 '참'이라는 값이 들어가게 된다. 따라서 이 쿼리가 데이터베이스에 들어가게 된다면 데이터베이스는 user라는 테이블에서 name이라는 칼럼에 id가 참 인 값을 리턴 할 것이기 때문에 우리는 id와 password를 모르는 상태로도 웹 사이트에 로그인을 할 수 있을 것이다. 이것이 sql injection의 기초이다. 다른 blind sql injection이나 union sql injection 또한 이러한 방식으로 이루어진다.

<여러가지 우회방법>

A. <Where 절에서 다른 칼럼을 사용하는 방법>

우리가 계속해서 사용하고 있는 친숙한 쿼리문을 다시 한번 보도록 하자.

SELECT name FROM user WHERE id='{$_GET[id]}' and password='{$_GET[password]}'

위에서 우리는 WHERE절에 대해 알아볼 때 WHERE절은 우리가 가지고 올 정보에 조건을 주는 것이라고 배웠다. 우선 이 쿼리문에서 우리는 id라는 칼럼과 password라는 칼럼에만 조건을 줄 수 있다. 그런데 만약 우리가 nickname이 hello인 값을 찾고 싶다면 어떻게 해야 할까? 여기서 우리는 다시 한번 or연산자를 이용할 수 있다. Nickname이 hello 인 값을 찾는 것은 간단하다. 그저 id에 **' or nickname='hello'%23**이라고 하는 값을 넣어주면 된다. 쿼리를 해석해 보면 id에는 아무런 값이 들어있지 않기 때문에 거짓이 되고, or연산 때문에 id가 거짓이라면 nickname에 hello라는 값이 들어가는 것을 알 수 있다. 그리고 뒤에 있는 한 줄 주석 기호인 %23(#)때문에 and 뒤의 password의 조건을 무시해 버린다.

B. <연산자를 우회하는 방법>

Sql injection에 관련된 여러가지 문제들을 풀어 본다면 가끔씩 and나 or같이 sql injection에서 꼭 필요한 연산자들이 필터링 되어 있는 것을 볼 수 있을 것이다. 이러한 연산자들이 필터링 되어있을 때 우리는 and를 &&로 or을 ||로 대체하여 사용할 수 있다. 프로그래밍 언어를 접해 본 사람이라면 거의 대부분의 언어에서 &&와 ||을 and와 or대신 사용할 수 있는 것을 알 것이다.

C. <주석을 이용하는 방법>

사실 우리는 주석을 어떤 식으로 이용해야 하는 지 이미 위에서부터 배워왔다. 우리가 삽입할 수 있는 값에 주석을 적절하게 배치시킨다면 where절의 조건을 우리의 입맛에 맞게 변화시킬 수 있다. 이처럼 한 줄 주석은 우리가 원하는 조건으로 만들기 위해 사용된다. 그러나 가끔 주석에 사용되는 일부 문자가 필터링 되어있는 상황이 발생할 수 있다. 그런 상황에 대비하여 이 장에서는 다양한 주석들에 대해 알아보도록 하겠다. 우선 가장 대표적인 주석으로는 우리가 위에서부터 사용해 내려왔던 주석인 #이 있다. 그러나 우리가 #을 url에 그대로 입력하지 않고 %23이라는 16진수로 인코딩 하여 넣었던 이유는 #을 그냥 넣게 되면 주석으로 인식이 되지 않기 때문이다. 다만 %23으로 16진수 방식으로 url에 삽입하게 되면 자동으로 디코딩 되어 서버사이드에서는 #으로 인식된다. 다

음으로 알아볼 주석 기법으로는 --(공백)이 있다. Sql 에서는 하이픈 2개와 뒤에 따르는 공백을 주석으로 인식한다. 다만 하이픈을 연달아 쓰고 스페이스 바를 한번 누르는 것으로 삽입 구문을 마치게 된다면 주석처리가 되지 않은 것을 볼 수 있다. url에서 맨 뒤에 오는 주석을 무시하기 때문인 것으로 추정된다(아마 쓸모가 없는 문자열로 취급하고 서버로 넘기기 전 잘라버리는 것으로 예상). 따라서 우리는 저 공백이 무시되지 않도록 만들어 주어야 한다. 방법은 간단하다. 그저 공백 뒤 임의의 문자를 써넣어 주면 되는 것이다. --(공백)- 처럼 어차피 공백 뒤에 올 글자는 주석처리가 되어 sql구문에 아무런 영향을 끼치지 않을 것이기 때문에 공백이 무시되지 않도록 임의의 글자를 넣어주는 것이다. 마지막으로 알아볼 주석은 바로 ";null"이다. Sql 에서는 세미콜론 뒤 null이 있다면 그것을 주석으로 처리한다. 다만 url에서 null을 그대로 작성하면 그것을 문자열로 받아들일 뿐 null문자로 처리하지 않는다. 따라서 우리는 null문자를 %00즉 16진수로 null을 의미하는 문자로 작성해야 한다. 정리해 보자면 우리가 대표적으로 사용하는 주석은

%23

--%20-

;%00

정도가 될 것이다.

D. <공백을 우회하는 방법>

가끔 sql injection관련 문제들을 풀다 보면 공백(%20)을 사용할 수 없는 문제들을 볼 수 있다. 이번 장에서 우리는 공백을 사용할 수 없는 상황을 만나게 되었을 때 사용할 수 있는 여러가지 공백 우회기법에 대해 알아보도록 하겠다. 우선 공백
은 %09(tab), %0a, %0b, %0c, space()함수, /**/로 대체할 수 있다. 다른 주석 우회방법들은 그냥 공백을 넣은 자리에 넣게 된다면 쉽게 사용할 수 있지만 space함수의 사용법에 대해서는 조금 다루겠다. space함수를 사용할 때는 space(원하는 공백의 개수)를 입력하면 된다. 만약 2개의 공백을 넣고 싶다면 space(2)라고 쓰면 된다.

정리하자면

%09

%0a,

%0b,

%0c

/**/

space함수

등이 있다.

E. <세미콜론을 우회하는 방법>

우선 세미콜론을 우회하는 방법을 이야기하기 전 세미콜론을 우회할 때는 앞에서 다룬 공백과는 다르게 다른 16진수를 이용하여 우회할 수 없다. 따라서 우리는 16진수를 사용하는 방법이 아닌 다른 방법을 사용해야 한다.

SELECT name FROM user WHERE id='{$_GET[id]}' and password='{$_GET[password]}'

우리가 계속해서 사용하던 쿼리를 다시 한번 보자. 이번에는 우리가 세미콜론을 사용할 수 없다고 생각하고 어떤 방법으로 sql injection을 수행해야 할 지 생각해보자. 우리가 사용할 방법은 바로 '문자열 이스케이프'이다. 혹시라도 다른 프로그래밍 언어를 접해본 적이 있다면 출력을 해야 하는 구문에서 세미콜론(')을 출력시킬 때 어떤 일을 했는지 생각해보자. 우리는 세미콜론(')을 출력해야 할 때 세미콜론 앞에 역슬레시(₩)를 붙여 세미콜론이 가지고 있는 문자열을 구분하는 기능을 없애주었다. 이 방법과 마찬가지로 우리는 id에는 ₩를 넣을 것이고 password에는 || 1=1%23을 넣어 줄 것이다. 우리가 넣은 값들이 들어간 쿼리문을 살펴보자

SELECT name FROM user WHERE id='₩' and password='|| 1=1%23'

이라는 쿼리문이 만들어지게 될 것이다. 쿼리문을 해석해 보자면 두번째 세미콜론(')이 앞에 있는 역슬레시(₩)때문에 문자열을 닫아주는 기능을 상실하게 된다. 따라서 id에는 'and password라는 문자열이 들어가게 되고, 이 문자열과 항상 참인 식인 1=1과 or연산을 실행하게 된다. 따라서 쿼리문은 user라는 값에서 항상 참인 값을 가져오는 식으로 바뀌어 버렸기 때문에 프로그래머가 의도하지 않은 방향으로 동작하게 된다.

F. <숫자를 우회하는 방법>

숫자를 우회해야 할 때는 자동형변환이라고 불리는 auto type cast을 사용하면 된다.

Sql 에서 true는 1, false는 0으로 인식하기 때문에 0이 필요할 때는 false를 사용하고, 1이 필요할 때는 1을 사용하면 된다. 만약 2나 3같은 다른 수가 필요하다면 어떤 식으로 우회해야 할까?

True는 1이므로

true + true = 2이다. 마찬가지로

true + true + true = 3이다.

G. <ereg나 str_replace 함수를 우회하는 방법>

가끔 여러가지 sql injection관련 문제들을 풀다 보면 ereg함수나 str_replace함수를 볼 수 있다. 먼저 이 함수들에 대해 설명하겠다.

우선 ereg함수는 ereg('찾을 문자열', '검사할 문자열')이라는 형태로 이루어져 있다. 이 함수는 보통 if문과 함께 사용되어 우리가 값을 가져오는데 중요한 문자열을 필터링한다.

그리고 str_replace함수가 있다 위의 ereg함수가 검사할 문자열에서 찾을 문자열이 있는지 없는지 검사해 주는 함수라면, str_replace함수 또한 검사할 문자열에서 찾을 문자열이 있는지 검사하고, 만약 우리가 원하는 문자열이 있다면 그 문자열을 다른 문자열로 대체해준다.

따라서 str_replace함수는 str_replace('찾을 문자열', '대체할 문자열', '검사할 문자열')형식으로 이루어져 있다.

select id from users where id='{$_GET[id]}'

라는 쿼리가 있고 우리는 id가 admin인 값을 꺼내야 한다고 가정하자. 그런데 소스코드를 보니 ereg('admin', $_GET[id])라는 함수를 이용하여 admin이라는 값을 필터링 하고 있다고 가정하자. 이때 우리는 서버에서 문자를 입력받는 방식을 사용하여 이 필터링을 우회할 수 있다. 서버에서는 우리가 입력한 문자의 대문자와 소문자를 구별하지 않는다. 따라서 서버사이드에서 'admin'이라는 문자열을 필터링 한다고 했을 때 우리가 'ADMIN'이라는 문자열을 주게 된다면 필터링을 피할 수 있고, 서버에서는 'ADMIN'과 'admin'을 같은 문자로 인식하기 때문에 우리는 'admin'으로 인식시킬 수 있다.

이번에는 str_replace함수를 우회하는 방법에 대해 알아보자.

Str_replace('admin', '', '$_GET[id]')

와 같은 방식으로 필터링 되어있다고 가정하자. 이 함수를 해석해보자면 $_GET[id]에서 admin이라는 문자열이 있다면 존재하는 admin이라는 문자열을 공백으로 대체한다는 뜻이다. 이 함수를 우회하는 방법은 대표적으로 2가지가 있다. 위에서 ereg함수를 우회한 것 처럼 'admin'이라는 문자열을 'ADMIN'이라는 값으로 바꾸어 전송하는 방법이 있고, 다른 하나는 'adadminmin'이라고 전송하는 방법이 있다. 이 값은 str_replace함수를 거치고 나면 'admin'이라는 값으로 바뀐다. 그 이유를 알아보면 'adadminmin'이라는 값이 str_replace함수를 거치면서 문자열 내부에 있는 'admin'이 사라지게 된다. 따라서 앞과 뒤에 있는 문자열인 ad와 min이 합쳐지면서 admin이라는 값이 서버에 넘어가게 된다.

H. <와일드카드를 사용하는 방법>

Sql 구문을 보다보면 'like'라는 구문이 들어간

4. <union sql injection>

A. <기본적인 union연산자의 사용법>

이번 장에서는 sql injection을 할 때 매우 유용한 union연산자에 대해 알아보도록 하겠다. Sql injection을 할 때 혹시 select구문을 다시 한 번 사용하고 싶다는 생각을 한 적이 있는가? 그럴 때 우리는 union구문을 사용할 수 있다. union문은 select 문을 한번 더 사용할 수 있도록 해 주지만 union을 이용한 sql injection이 마냥 단순한 것은 아니다. Union sql injection을 할 때 가장 중요한 것은 union뒤에 올 select 문에서 사용할 칼럼의 개수와 앞에서 사용한 칼럼의 개수가 일치해야 한다. 이렇게 글로만 읽어서는 이해가 잘 되지 않을 수 있으니 예를 들어보도록 하겠다.

예를 들어

SELECT id, username FROM user WHERE id='' and passwd=''

이라는 쿼리문이 있다고 하자. 이 쿼리문을 보게 되면 id와 username이라는 2개의 칼럼을 가져오게 된다. 만약 우리가 이 쿼리문에서 union연산자를 사용하여 다시 select 구문을 사용하고 싶다면 어떻게 해야 할까?

위에서 말 한 것처럼 union문에서 사용할 select구문에 올 칼럼의 개수를 앞에서 사용한 칼럼의 개수와 맞춰 주어야 한다.

SELECT id, username FROM user WHERE id=1

UNION

SELECT 1, password FROM user WHERE id=1

위에 작성한 쿼리처럼 말이다. 또한 union문을 사용할 때는 앞의 칼럼과 뒤의 칼럼의 자료형이 같아야 한다.

union연산자의 또 다른 특징으로는 앞에 있는 select구문을 무시하고, 뒤에 있는 select문만 실행시킨다는 것이다. 이제 union에 대해 조금 이해가 되었는가?

정리하자면, union문을 사용할 때는 앞의 select문에서 사용한 칼럼의 개수와 뒤의 select문에서 사용한 칼럼의 개수가 같아야 하고, 앞과 뒤의 칼럼의 자료형이 같아야 한다. 그리고 union연산자는 앞에 있는 select문을 무시하고 우리가 삽입한 select문만을 실행시킨다.

B. <칼럼의 개수를 확인하는 방법>

위의 union연산자를 설명한 장에서 우리가 union sql injection을 할 때 가장 중요한 것들 중 하나는 칼럼의 개수를 확인하는 것 이라는 언급을 했다. 그렇다면 우리는 칼럼의 개수를 어떻게 확인할 수 있을까? 칼럼의 개수를 확인하는 방법은 대표적으로 2가지가 있다.

우선 첫번째 방법의 형태는 이렇다.

1 union select '1'-- -

1 union select '1', '2'-- -

1 union select '1', '2', '3'-- -

...

위의 쿼리문처럼 '1', '2', '3'을 하나씩 늘려나가는 방법이다. 이 방법은 우리가 넣은 숫자의 개수가 앞의 select문의 칼럼의 개수보다 작거나 같다면 오류가 발생하지 않지만, 만약 칼럼의 개수보다 많다면 오류가 발생한다. 따라서 우리는 오류가 발생하는 것을 보고 앞의 select문에서 사용한 칼럼의 개수를 알 수 있는 것이다.

예를 들어

SELECT id, username FROM user WHERE id="

이러한 쿼리문이 있을 때 앞의 select문에 사용된 칼럼의 개수는 2개이다. 이 때 우리가 id에

1 union select '1'-- -

1 union select '1', '2'-- -

1 union select '1', '2', '3'-- -

이러한 값들을 순차적으로 대입한다면 어떤 일이 발생할까? 첫 번째 쿼리와 두 번째 쿼리에서는 오류가 발생하지 않겠지만, 세 번째 쿼리에서 넣은 값의 수가 앞에서 사용한 칼럼의 수보다 많기 때문에 오류가 발생할 것이다. 따라서 우리는 칼럼의 개수가 2개임을 유추할 수 있는 것이다.

두 번째로 칼럼의 개수를 유추할 수 있는 방법은 'order by'를 이용하는 것이다. 원래 order by구문은 테이블에 있는 데이터를 오름차순 또는 내림차순으로 정렬할 때 이용하는 구문이다. 하지만 이 구문을 이용하여 칼럼의 개수를 유추할 수 있다.

예를 들어

SELECT id, username FROM user WHERE id=

라는 구문이 있다고 가정했을 때 order by를 이용하여 칼럼의 개수를 가져오는 방법에 대해 알아보도록 하자.

방법은 id에

1 order by 1-- -

1 order by 2-- -

1 order by 3-- -

…

처럼 order by 뒤에 오는 숫자를 증가시키면서 칼럼의 개수를 찾는 것이다. 위에서 사용했던 방법처럼 우리가 order by 구문을 사용하면서 뒤의 숫자를 증가시킬 때 앞의 select 문에서 사용한 칼럼의 개수보다 order by구문 뒤에 나온 숫자가 작거나 같다면 오류가 발생하지 않지만, 칼럼의 개수보다 크다면 오류가 발생한다. 따라서 이 방법을 사용하여 칼럼의 개수를 찾을 수 있다.

정리하자면,

1 union select '1'-- -

1 union select '1', '2'-- -

1 union select '1', '2', '3'-- -

과

1 order by 1-- -

1 order by 2-- -

1 order by 3-- -

를 이용하여 칼럼의 수를 찾을 수 있다.

C. <mysql에서 table 정보를 가져오는 방법>

우리가 sql injection을 할 때 중요한 것 중 하나는 우리가 탈취할 정보가 들어있는 테이블을 찾는 것이다. 이번 장에서는 우리가 어떻게 공격을 수행할 테이블을 찾는지에 대해 알아보겠다. Mysql에는 information_schema라는 테이블이 존재한다. 이 테이블은 사용자가 만든 데이터에 의한 데이터로써 사용자가 사용한 데이터들에 관련된 정보들이 모두 존재한다. 따라서 우리는 이 테이블을 사용하여 우리가 공격할 테이블을 찾을 수 있다.

우리는 information_schema라는 테이블에서 정보를 가져올 것이기 때문에 select문을 한 번 더 사용해야 한다. 우리가 select문을 다시 한번 사용하기 위해서는 union연산자를 이용해야 한다.

SELECT id, username FROM user WHERE id=

이라는 쿼리문이 있다고 가정하자. 우리는 공격할 테이블을 찾을 때, information_schema라는 데이터베이스를 사용한다고 했다. 이 데이터베이스 속에 information_schema.table이라는 이름의 테이블이 존재한다. 이 테이블 속에는 데이터의 이름과 같은 테이블과 관련된 정보들이 들어있다. 따라서 우리가 사용할 쿼리를 구상해보자.

우리가 가져올 테이블 이름은 table_name이라는 칼럼에 저장되어 있다. 그리고 저 칼럼이 속한 테이블은 information_schema.table에 저장되어 있다. 따라서 우리가 사용할 쿼리를 생각해 보면

1 UNION SELECT 1, table_name FROM information_schema.table

이라는 형태가 될 것이다. 그러나 이 쿼리를 실행해 본다면 우리가 원하는 테이블이름이 나오지 않는 것을 볼 수 있다. 이 칼럼에서 우리가 원하는 데이터베이스 이름을 가져오려면 실제로 사용하고 있는 데이터베이스를 가져올 필요가 있다. 실제로 사용하고 있는 데이터베이스를 가져오려면 where table_schema=database()라는 조건을 주게 되면 실제로 사용하는 데이터베이스를 찾을 수 있다.

1 UNION SELECT 1, table_name FROM information_schema.table where table_schema=database()

따라서 쿼리는 이러한 형태로 구성될 것이다.

그런데 우리가 가져와야 할 데이터베이스의 개수가 한 개가 아니라면 어떻게 해야 할까? 그때 우리는 limit라는 조건을 사용한다. 위의 쿼리를 보면 table_name이라는 칼럼에 있는 실제로 사용되고 있는 데이터베이스 중 가장 위에 있는 스키마를 가져오는 것이다. 따라서 두 번째나 세 번째 스키마를 가져오려면 limit 2,1그리고 limit 3,1이라고 작성하면 된다. Limit구문은 limit '가져올 스키마의 번호', '가져올 개수'인 형식으로 이루어져 있다.

따라서 limit 절을 적용하면 최종적으로

1 UNION SELECT 1, table_name FROM information_schema.table where table_schema=database() limit 1,1

과 같은 형태가 될 것이다.

정리하자면,

1 UNION SELECT 1, table_name FROM information_schema.table where table_schema=database() limit 1,1

1 UNION SELECT 1, table_name FROM information_schema.table where table_schema=database() limit 2,1

1 UNION SELECT 1, table_name FROM information_schema.table where table_schema=database() limit 3,1

처럼 limit 절 뒤의 숫자를 하나씩 증가시키면서 테이블의 이름을 가져올 수 있는 것이다.

D. <mysql에서 칼럼 정보를 가져오는 방법>

Sql injection공격을 수행할 때 우리는 우리가 공격할 테이블의 이름 뿐만이 아니라 우리가 원하는 정보를 가진 정보를 가진 칼럼을 가져와야 한다. 이 때 우리는 위에서 테이블을 가져온 방식과 유사하게 information_schema에 들어있는 칼럼의 이름을 가져와야 한다. 사용자가 데이터베이스를 만들 때 지은 칼럼의 이름은 information_schema.columns에 들어있고, column_name이라는 테이블에 저장되어 있다. 따라서 우리는 위에서 테이블의 이름을 빼 오던 것 과 같이 칼럼의 이름을 가져올 수 있다.

SELECT id, username FROM user WHERE id=

위에서 사용했던 것과 같이 이러한 쿼리가 있다고 가정하자. 그리고 우리가 공격할 테이블의 이름을 user라고 할 때 칼럼의 이름을 가져오는 방법에 대해 알아보자.

위에서 테이블 이름을 가져올 때 우리는 select구문을 하나 더 사용하기 위해 union구문을 사용했다. 마찬가지로 우리는 이번 장에서 칼럼의 이름을 가져오기 위해 select구문과 union구문을 사용할 것이다.

쿼리를 구성해 보자면

1 UNION SELECT 1, column_name FROM information_schema.columns

가 될 것이다. 그러나 이렇게만 작성한다면 우리가 볼 칼럼의 이름은 information_schema.columns의 가장 상위에 있는 테이블의 이름 뿐이다. 따라서 우리는 우리가 가져오고 싶은 이름의 칼럼이 있는 테이블의 이름을 조건으로 줄 필요가 있다. 따라서 쿼리는

1 UNION SELECT 1, column_name FROM information_schema.columns where

table_name='user'

이 될 것이다. 그러나 이 쿼리 또한 user라는 테이블에서 가장 위에 있는 칼럼의 이름을 가져오는 것이기 때문에 우리는 모든 칼럼의 이름을 알 수 없다. 우리가 모든 칼럼의 이름을 가져오려면 이전 장에서 사용했던 것과 같은 limit절을 사용하면 된다. 따라서 우리가 사용할 쿼리는 이런 식으로 구성될 것이다.

1 UNION SELECT 1, column_name FROM information_schema.columns where

table_name='user' limit 1,1

이제 우리는 limit구문에서 숫자를 하나씩 증가시키며 우리가 원하는 테이블에 있는 모든 칼럼의 정보를 알 수 있게 되었다.

5.<blind sql injection>

A. <blind sql injection이란>

우리가 sql injection공격을 수행할 때 우리는 사용자와 관련된 정보를 직접적으로 알아내야 하는 경우들이 있다. 이 때 우리가 사용하는 공격 기법이 바로 blind sql injection이다. 우리는 blind sql injection을 할 때 문자들을 하나하나 대입하며 우리가 원하는 문자열을 추출해 낸다. 이 과정은 사람이 손으로 직접 수행하기에는 많은 무리가 있기 때문에 보통 여러가지 프로그래밍언어를 이용하여 코드를 구성해 공격을 수행한다.

우리는 여러가지 함수들을 사용하여 문자열을 잘라 우리가 대입한 값과 비교를 진행할 것이다. 우리가 사용할 수 있는 함수는 substr함수, mid함수, left함수, right함수 등이 있다.

<substr 함수를 이용한 blind sql injection>

우선 substr함수에 대해 알아보도록 하겠다. Substr함수는 말 그대로 문자열을 잘라주는 함수이다.

substr함수의 사용법을 예를 들자면,

Substr('abcd', 1, 1) = a 이다.

Substr함수의 첫번째 인자는 자를 문자열을 의미하고, 2번째 인자는 몇 번째 글자부터 문자열을 자를 것인지를 의미하고, 마지막으로 3번째 인자는 앞에서 지정한 번째의 글자부터 몇 개의 글자를 자를 것인지를 이야기한다.

<mid 함수를 이용한 blind sql injection>

다음으로 우리가 알아볼 함수는 mid함수이다. 우리가 blind sql injection을 사용할 때 substr함수가 필터링 되어있는 경우가 있다. 이 때 우리는 mid함수를 이용한 blind sql injection을 할 수 있다.

Mid함수의 사용법을 예를 들자면,

Mid('abcd', 1, 1,) = a 이다.

mid함수의 사용법은 substr함수와 동일하다.

<left, right 함수를 이용한 blind sql injection>

다음으로 우리가 알아볼 함수는 left, right함수이다. 이 함수들은 하나만 사용해서는 우리가 원하는 문자열에서 한 글자만을 가져올 수 없다. 따라서 우리는 이 두가지 함수들을 조합하여 사용해야 한다.

Left 함수는 문자열의 왼쪽에서 우리가 인자로 준 숫자 번째부터 잘라준다.

예를 들어

Left('abcd', 1) = a

Left('abcd', 2) = ab

와 같은 방식이다.

반대로 right함수는 문자열의 오른쪽부터 우리가 인자로 준 숫자 번째부터 문자열을 잘라준다.

예를 들자면

Right('abcd', 1) = d

Right('abcd', 2) = cd

와 같은 식이다.

따라서 우리는 이 두 함수를 이용하여 문자열을 한 글자씩 분리할 수 있다.

예를 들어

Right(Left('abcd', 1), 1) = a

Right(left('abcd', 2), 1) = b

Right(left('abcd', 3), 1) = c

…

와 같은 방식이다.