Special Report

웹 취약점과 해킹 매커니즘 #3 UNION SQL Injection

■ 개요

SQL Injection은 사용자 입력값을 검증하지 않아 설계된 쿼리문에 의도하지 않은 쿼리를 임의로 삽입하여 악의적인 SQL 구문을 실행시키는 공격이다. 이번 Special Report는 UNION SQL Injection의 개념과 SQL Injection 공격에 취약한 소스코드로 구현한 게시판의 검색 기능에서 원하는 데이터를 추출하는 과정까지의 내용을 다룬다.

※ 실제 운영 중인 서버에 테스트 또는 공격을 하는 행위는 법적인 책임이 따르므로 개인용 테스트 서버 구축 또는 bWAPP, DVWA, WebGoat 등과 같은 웹 취약점 테스트 환경 구축을 통해 테스트하는 것을 권장한다.

※ 본 Special Report는 JSP와 Oracle Database 11gR2를 사용하여 취약한 서버를 구축하였다.

■ 환경 구성

UNION SQL Injection은 주소 찾기, 게시글 검색 등 사용자 입력값을 받아 그 결과를 페이지에 출력해 주는 기능에서 볼 수 있다. 이번 리포트에서는 웹 취약점 테스트용으로 구축한 서버의 게시판에서 진행된다. 아래의 그림과 같이 구성되었으며 사용자가 제목에 해당하는 특정 단어를 검색하면 서버는 데이터베이스에서 결과를 불러와 화면에 출력해 준다.

[UNION SQL Injection 취약점이 있는 게시판 페이지]

게시판의 검색 기능은 다음과 같은 취약한 소스코드로 구성되어 있다. 사용자 입력값인 searchWord에 대한 입력값 필터링이 존재하지 않아 공격자는 쿼리 조작이 가능하다.

infosec

```
String sql = "SELECT idx, title, content, userid FROM board
WHERE title LIKE '%" + searchWord + "%'";

Statement stmt = conn.createStatement();

rs = stmt.executeQuery(sql);
```

[SQL Injection 에 취약한 소스코드]

또한 실습용 웹 서버의 데이터베이스는 사용자 정보를 담고 있는 MEMBER 테이블과 게시판 페이지 정보를 담고 있는 BOARD 테이블로 구성되어 있다.

※ 비밀번호는 개인정보보호법에 따라 단방향 해시 처리된 값으로 구성되어 있다.

infosec

아이디 (userid)	비밀번호 (userpw)	이름 (username)	이메일 (usermail)	전화번호 (usertel)
admin	225BC6…	관리자	-	-
eqst	34A603…	이큐스트	-	-
insight	B9E80C…	인사이트	-	-
web	C5F491…	웹	-	-

[MEMBER 테이블]

infosec

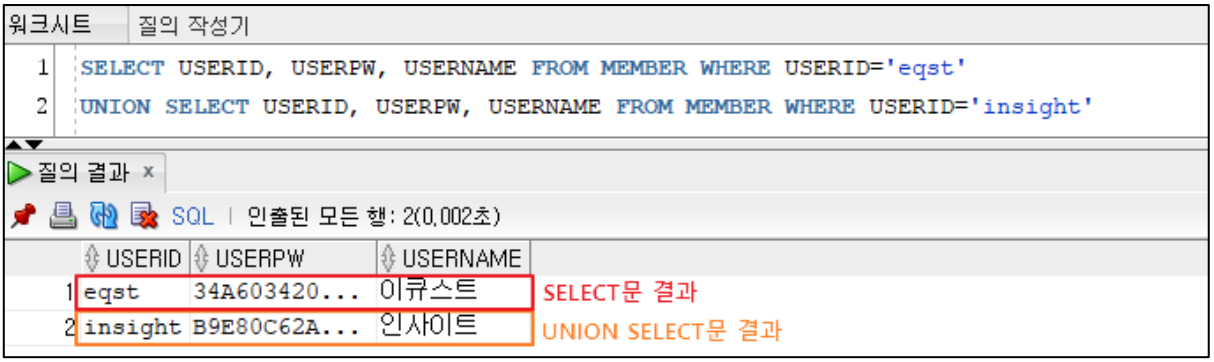
번호 (idx)	제목 (title)	내용 (content)	작성자 (userid)	작성일 (boarddate)
1	eqst	SKshieldus	이큐스트	22/04/11
2	insight	EQST Insight	인사이트	22/04/11
…	…	…	…	…

[BOARD 테이블]

■ UNION SQL Injection

UNION SQL Injection은 기존의 SELECT문에 UNION SELECT문을 추가하여 원하는 정보를 데이터베이스에서 추출하는 공격이다.

UNION 연산자[1]는 아래의 그림과 같이 두 개 이상의 SELECT문에 대한 결과를 하나의 결과로 추출한다.



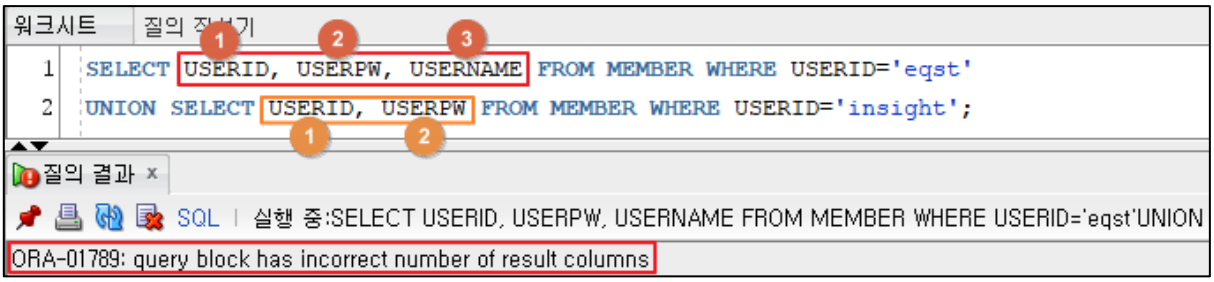
[USER ID 가 eqst 와 insight 인 컬럼 조회 결과]

공격자는 이러한 점을 이용하여 기존의 SELECT문에 원하는 데이터를 추출하기 위한 UNION SELECT문을 추가하여 쿼리 결과를 확인할 수 있다.

UNION 연산자를 사용하기 위해서는 **두 가지 조건**을 만족해야 한다.

1) 기존의 SELECT문과 UNION SELECT문의 컬럼 수가 동일해야 한다.

※ 컬럼 수가 동일하지 않을 경우 아래의 그림처럼 에러 메시지를 반환한다.



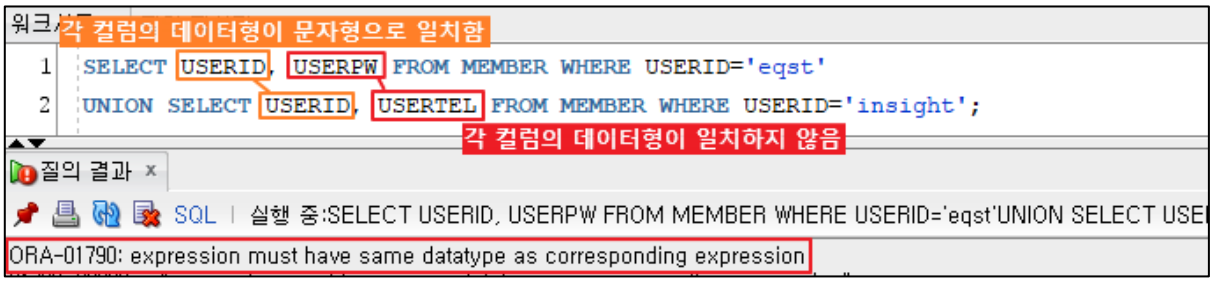
[동일하지 않은 컬럼 수로 인한 에러 발생]

[1] UNION은 기본적으로 중복을 제거한 결과를 보여준다. 중복을 포함한 결과를 추출할 경우 UNION ALL을 사용하면 된다.

2) 각각의 컬럼은 순서 별로 동일한 데이터형이어야 한다.[2]

※ 각 컬럼의 데이터형이 동일하지 않을 경우 아래의 그림처럼 에러 메시지를 반환한다.

현재 데이터베이스의 USERID와 USERPW 컬럼은 문자형이며, USERTEL 컬럼은 숫자형으로 구성되어 있다. 따라서 SELECT문과 UNION SELECT문의 두 번째 컬럼인 USERPW와 USERTEL의 데이터형이 일치하지 않아 에러가 발생한다.



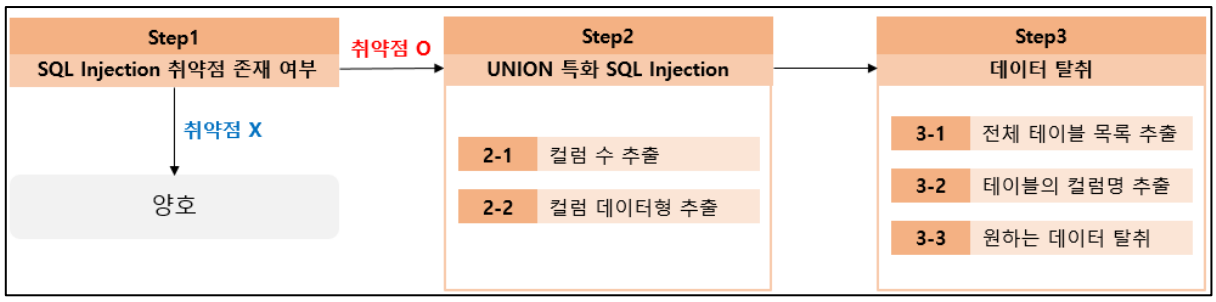
[동일하지 않은 데이터형으로 인한 에러 발생]

따라서 UNION SQL Injection을 진행하기 위해서는 기존의 SELECT문에 대한 컬럼 수와 데이터형을 알아내는 과정이 필요하다.

[2] MySQL의 경우 자동 형 변환이 이루어지기 때문에 데이터형이 일치하지 않아도 된다.

■ 공격 진행 과정

UNION SQL Injection의 공격 진행 과정은 다음과 같다.



[UNION SQL Injection 진행 과정]

SQL Injection 취약점 존재 여부를 확인한 후 취약점이 존재한다면, UNION 연산자를 이용하기 위해 기존의 SELECT문에 대한 컬럼 수와 데이터형을 추출한다. 이 정보를 바탕으로 전체 테이블 목록과 테이블의 컬럼명을 추출하여 원하는 데이터를 탈취할 수 있다.

Step 1. 취약점 존재 여부 확인

사용자 입력값을 결과로 출력해 주는 게시판의 검색 기능에서 SQL Injection 취약점 존재 여부를 확인한다. SQL구문에서 문법적 요소로 작용하는 싱글쿼터('), 파이프(|) 등과 같은 특수문자를 검색했을 때 나타나는 서버의 반응을 보고 취약점 존재 여부를 판단할 수 있다.

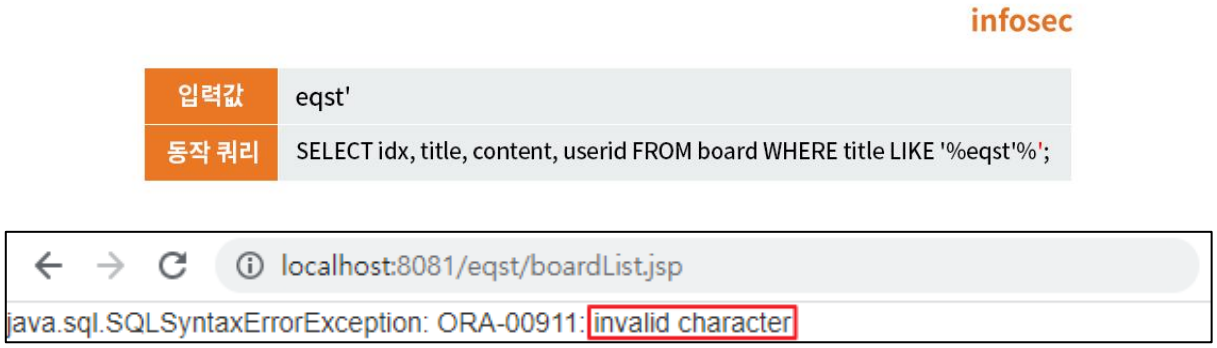
게시판 검색 기능의 SQL 구문은 아래와 같이 구성되어 있다.

```
SELECT idx, title, content, userid FROM board WHERE title LIKE '%" + searchWord + "%';
```

사용자가 제목(title)에 해당하는 특정 단어(searchWord)를 검색하면 글번호(idx), 제목(title), 내용(content), 작성자(userid) 순서로 결과로 보여준다. 예를 들어, 사용자 입력값이 eqst라면 LIKE 이후의 값이 '%eqst%'[3]이 되어 eqst를 포함하는 모든 문자를 검색한 결과가 추출된다.

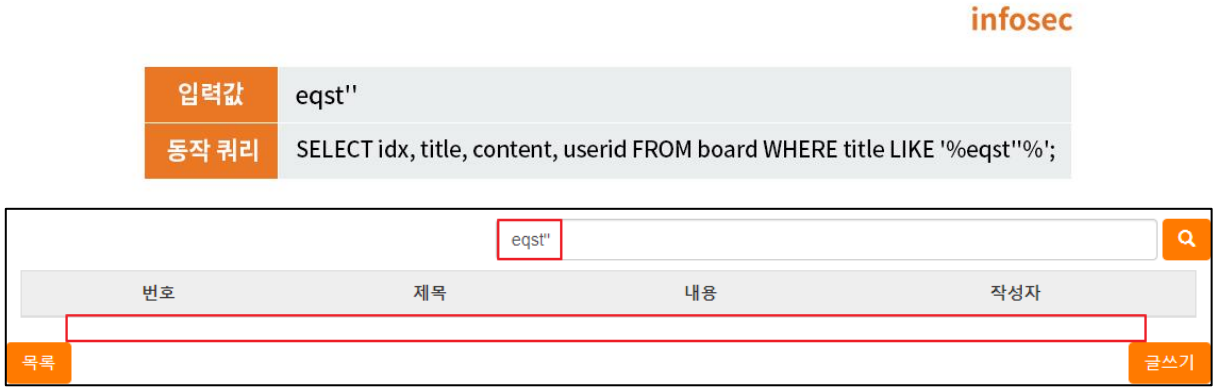
[3] LIKE 연산자는 문자열의 패턴을 검색하는데 사용되며, 퍼센트(%)와 언더바(_)를 통해 패턴을 지정할 수 있다. 퍼센트는 '모든 문자'를 의미하고 언더바는 '문자 하나'를 뜻한다.

SQL Injection 취약점 존재 여부를 확인하기 위해 입력값에 싱글쿼터 하나를 붙여 eqst'을 검색해 보면 문자 부적합을 뜻하는 에러 메시지가 반환되는 것을 볼 수 있다. 이는 검색어에 포함된 싱글쿼터가 서버 측의 SQL구문에서 문법적 요소로 작용하여 싱글쿼터 하나가 남게 되어 발생한다.



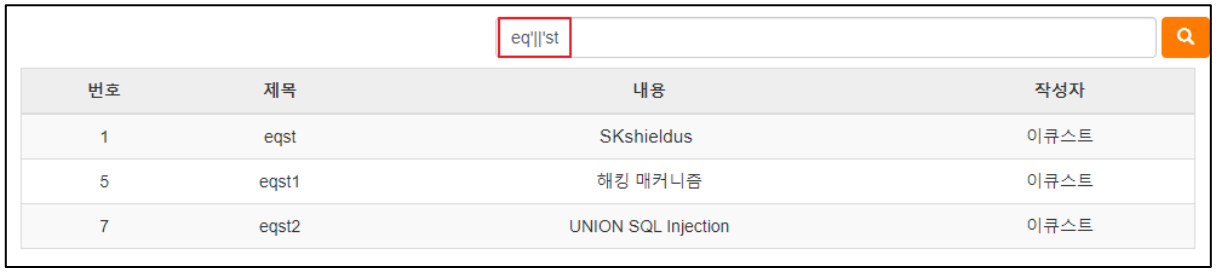
[eqst' 조회 결과]

입력값에 싱글쿼터 두 개를 추가하여 eqst''을 검색하면 SQL 구문에 있는 싱글쿼터의 개수가 짝이 맞아 에러 발생 없이 조회되는 것을 확인할 수 있다.



[eqst'' 조회 결과]

또한, ORACLE에서 문자열 연결에 사용되는 특수문자인 '||'을 포함한 eq'||'st를 검색해보면 eqst를 검색했을 때와 동일한 결과를 출력하는 것을 볼 수 있다.



[eq'||'st 조회 결과]

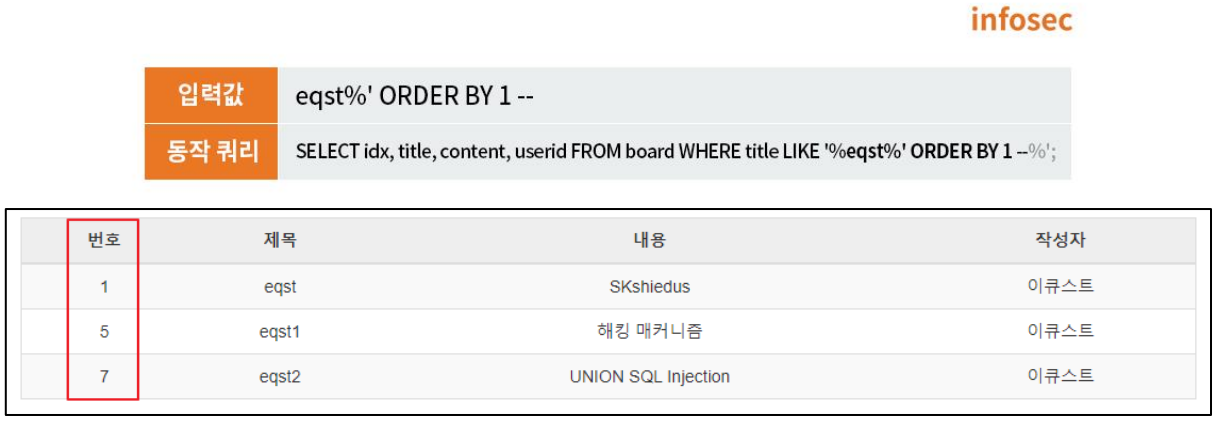
위의 결과들을 통해 공격자가 입력한 특수문자가 SQL 구문에 삽입되어 문법적 요소로 동작함을 알게 되었고 SQL Injection 취약점이 존재한다고 판단할 수 있다.

Step 2. UNION 특화 SQL Injection

UNION 연산자 사용 조건에 만족하기 위해 SELECT문의 컬럼 수와 데이터형을 알아내는 과정이 필요하다.

2-1) 컬럼 수 추출

기존의 SELECT문의 컬럼 수는 데이터 정렬 기능을 하는 ORDER BY절[4]을 사용하여 알아낼 수 있다. 게시판 검색 기능에 **eqst%' ORDER BY 1 --**[5]을 검색하면 제목에 'eqst' 문자가 포함된 모든 값이 ORDER BY 1에 의해 첫 번째 컬럼인 '번호'를 기준으로 오름차순 정렬이 되어 조회된 것을 볼 수 있다.



[eqst%' ORDER BY 1 -- 조회 결과]

[4] ORDER BY절은 **'ORDER BY 컬럼명(컬럼 순서) [ASC|DESC]**'의 형태로 쓰이며 컬럼명(컬럼 순서)을 기준으로 정렬한다. 정렬 기준은 ASC(오름차순)과 DESC(내림차순)이 있으며 기본적으로 ASC 정렬을 한다.

[5] ORDER BY 1 이후의 쿼리문은 '--'에 의해 주석 처리되었다.

ORDER BY절의 컬럼 순서를 나타내는 숫자를 증가시키며 조회하다 보면 아래의 그림처럼 에러가 발생하는 때가 있다. 이는 SELECT문의 컬럼 수보다 높은 숫자로 정렬할 경우 발생하는 에러로 **eqst%' ORDER BY 5 --**에서 에러가 발생한 것을 바탕으로 SELECT문의 컬럼 수는 4개임을 알 수 있다.

[eqst%' ORDER BY 5 -- 조회 결과]

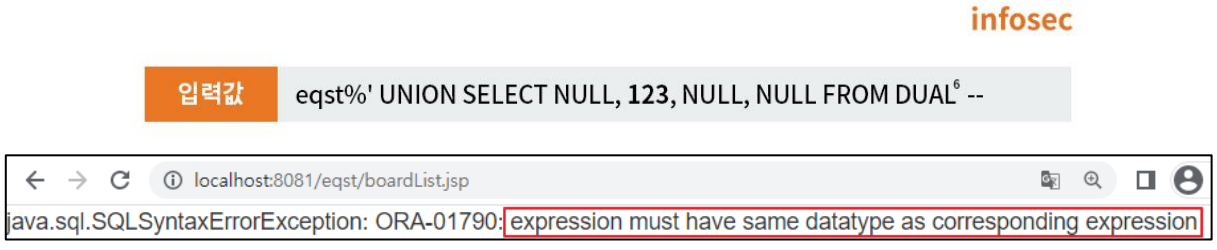
2-2) 컬럼 데이터형 추출

SQL은 문자열을 싱글쿼터로 감싸서 표현하므로 숫자형과 문자형의 구별이 가능하다. 앞서 추출한 컬럼 수만큼의 NULL문자를 추가하여 UNION SELECT문을 작성한 후, 데이터형을 알고자 하는 컬럼에 NULL문자 대신 숫자나 문자를 입력하여 검색했을 때 서버 측의 반환 결과를 보고 데이터형을 판단할 수 있다.

SELECT문의 두 번째 컬럼의 데이터형을 알아내기 위해 NULL문자 대신 숫자 123과 문자 123을 의미하는 '123'을 넣어 조회한 결과는 다음과 같다.

① 숫자 123 조회 결과 - **에러 발생**

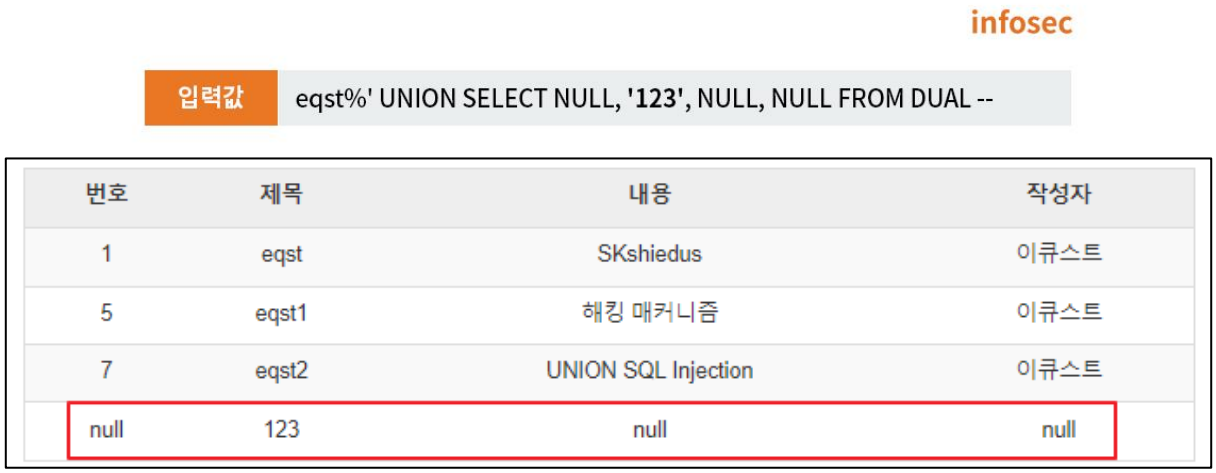
데이터형이 동일하지 않다는 에러 메시지가 반환되는 것을 확인할 수 있다.



[숫자 123 조회 결과]

② 문자 123 조회 결과 – **정상 출력**

두 번째 컬럼의 위치에 문자 123이 정상적으로 출력되는 것을 보아 데이터형이 문자임을 알 수 있다.



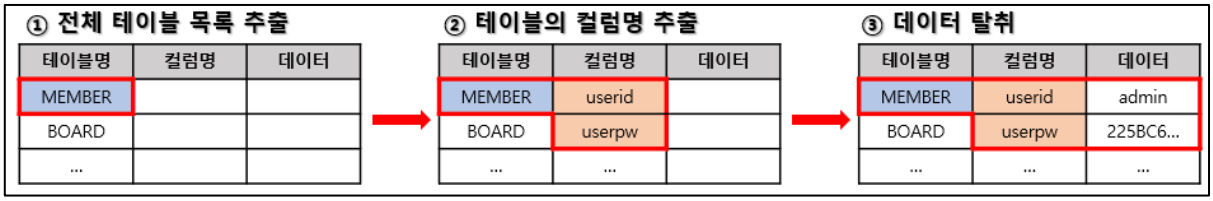
[문자 123 조회 결과]

각각의 컬럼마다 위의 과정을 반복해서 모든 컬럼에 대한 데이터형을 추출할 수 있다.

Step 3. 데이터 탈취

SELECT문에 대한 컬럼 수와 데이터형을 획득했다면, 데이터베이스에서 원하는 데이터를 탈취하기 위한 UNION SELECT문을 작성할 수 있다.

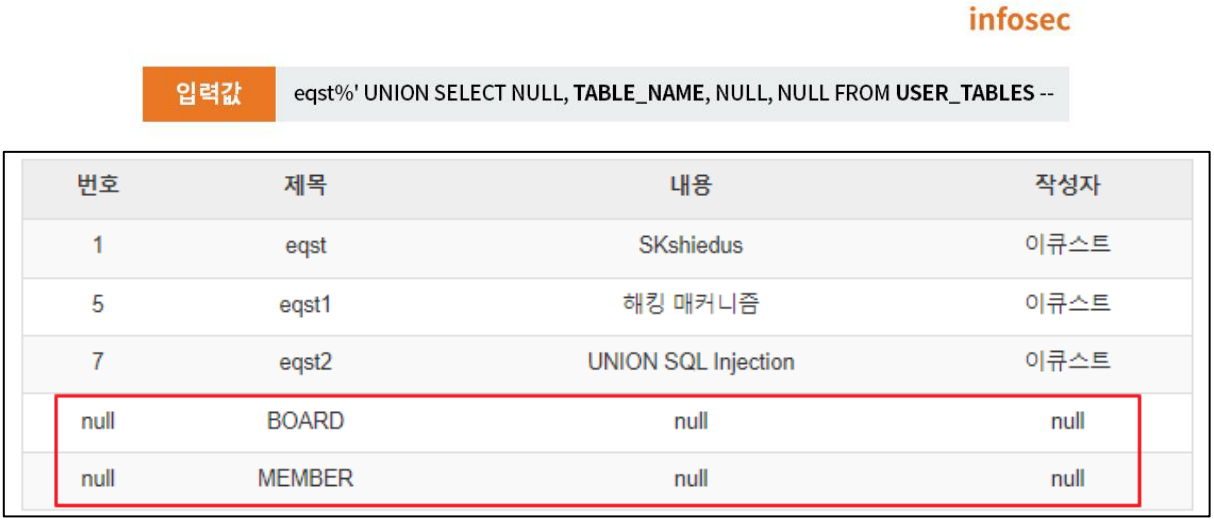
데이터 탈취 과정은 아래와 같다. 데이터베이스의 전체 테이블 목록을 추출한 후 데이터를 탈취할 테이블의 컬럼명을 확인한다. 획득한 테이블명과 컬럼명을 조회하여 원하는 데이터를 탈취할 수 있다.



[데이터 탈취 과정]

3-1) 전체 테이블 목록 추출

사용자가 생성한 테이블의 정보가 담긴 'USER_TABLES'[6]에서 전체 테이블 목록을 추출할 수 있다. 앞서 확인한 데이터형이 문자인 두 번째 컬럼에 테이블명을 뜻하는 'TABLE_NAME'을 넣어 검색하면 다음과 같은 결과가 조회된다.

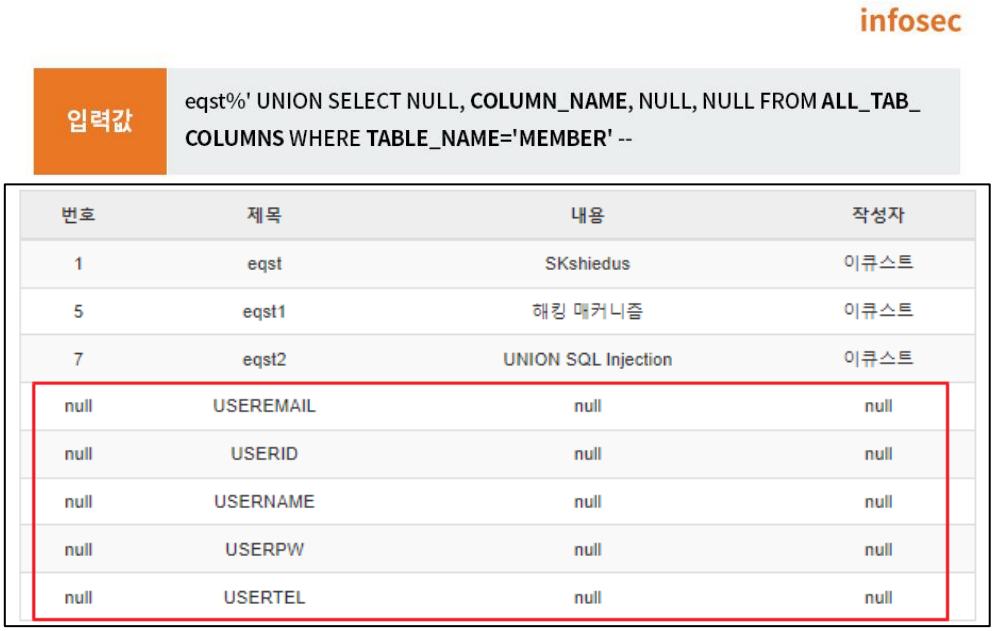


[USER_TABLES 의 TABLE_NAME 조회 결과]

[6] USER_TABLES는 ORACLE 기본 테이블 중 하나이며 이외에도 데이터베이스의 전체 테이블 목록을 담고 있는 'ALL_TABELS'을 사용할 수 있다.

3-2) 원하는 테이블의 컬럼명 추출

테이블의 목록 중 회원정보가 담긴 MEMBER 테이블의 컬럼명을 추출을 위해 모든 테이블의 컬럼 정보가 담긴 'ALL_TAB_COLUMNS' 테이블에서 MEMBER 테이블의 컬럼명을 추출한다. 두 번째 컬럼에 컬럼명을 뜻하는 'COLUMN_NAME'을 넣어 검색해 보면 다음과 같은 결과가 조회된다.

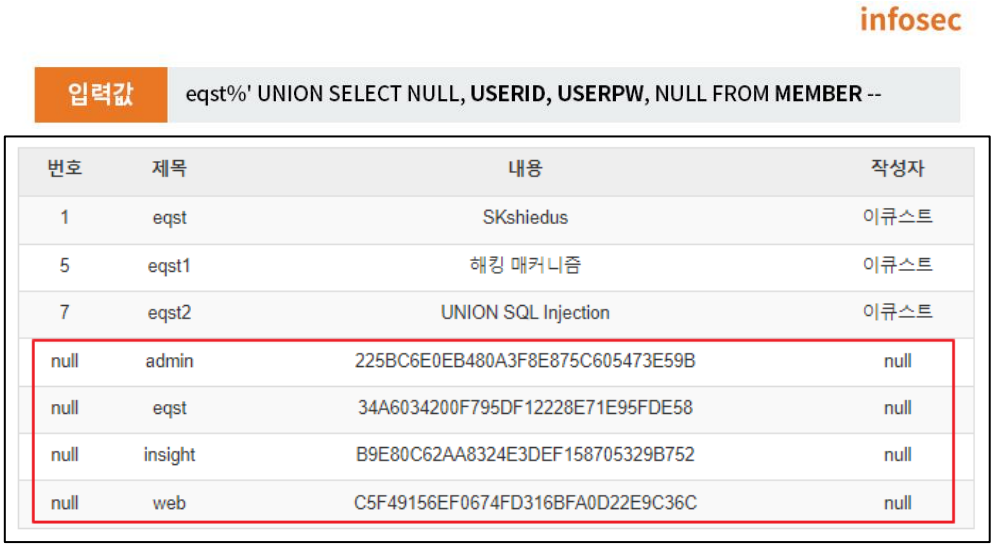


번호	제목	내용	작성자
1	eqst	SKshiedus	이큐스트
5	eqst1	해킹 매커니즘	이큐스트
7	eqst2	UNION SQL Injection	이큐스트
null	USEREMAIL	null	null
null	USERID	null	null
null	USERNAME	null	null
null	USERPW	null	null
null	USERTEL	null	null

[MEMBER 테이블의 모든 컬럼명 조회 결과]

3-3) 데이터 탈취

MEMBER 테이블의 컬럼 중 아이디와 암호화된 비밀번호가 있는 'USERID'와 'USERPW' 컬럼을 조회하여 데이터를 추출한다.



번호	제목	내용	작성자
1	eqst	SKshiedus	이큐스트
5	eqst1	해킹 매커니즘	이큐스트
7	eqst2	UNION SQL Injection	이큐스트
null	admin	225BC6E0EB480A3F8E875C605473E59B	null
null	eqst	34A6034200F795DF12228E71E95FDE58	null
null	insight	B9E80C62AA8324E3DEF158705329B752	null
null	web	C5F49156EF0674FD316BFA0D22E9C36C	null

[MEMBER 테이블의 모든 컬럼명 조회 결과]

■ 보안 대책

SQL Injection 의 보안 대책은 크게 2 가지가 있다.

- Prepared Statement[7] : 근본적인 해결책으로 많이 사용하지만, 문법적/비즈니스 로직 상 사용이 불가한 로직이 있으며 서버가 운영 중일 경우 소스코드 수정이 어려울 수 있다.

- Filtering : White List Filter 방식을 적용해 허용할 문자열을 지정하는 것이 좋다. 상황상 Black List Filter 방식을 적용해야 한다면, 공격 기법에 사용되는 예약어 및 특수 문자를 모두 Filtering 해야 한다.

※ 문자열 필터링 시 대소문자 모두 필터링하는 것이 좋다.

※ UNION SQL Injection 의 경우 UNION, UNION ALL, ORDER BY, NULL 등의 문자에 대한 필터링이 필요하다.

보안 대책에 대한 우회기법 및 시큐어코딩 적용 등에 대한 자세한 내용은 SQL Injection 마지막 챕터에서 이어진다.

■ 맺음말

지금까지 집합 연산자 UNION을 이용한 UNION SQL Injection의 공격 과정에 대해 알아봤다. UNION 연산자 사용 조건에 만족하기 위해 ORDER BY절과 NULL 문자를 사용해 기존의 SELECT문의 컬럼명과 데이터형을 추출했고 그 정보를 바탕으로 전체 테이블 목록, 테이블의 컬럼명을 조회하여 원하는 데이터를 탈취할 수 있다.

[7] Prepared Statement는 컴파일이 미리 되어있기 때문에 입력값을 변수로 선언해두고 필요에 따라 값을 대입하여 처리한다.