SQL Injection Demonstration

Enter Your Name:

Your Name

Start Experiment

Current Student: **Travis** (Change User)

Database Users

ID	Name	EID	Salary	Birth	SSN	Address	Email	Nickname	Phone
1	admin	99999	400000	3/5	43254314	admin address	admin@example.com	admin	123456789
2	alice	10000	20000	9/20	10211002	alice address	alice@example.com	alice	123456789
3	boby	20000	50000	4/20	10213352	boby address	boby@example.com	boby	123456789
4	ryan	30000	90000	4/10	32193525	ryan address	ryan@example.com	ryan	123456789
5	samy	40000	40000	1/11	32111111	samy address	samy@example.com	samy	123456789
6	ted	50000	110000	11/3	24343244	ted address	ted@example.com	ted	123456789

- Login Page
- Update Profile Page

Reference: SEED Labs – SQL Injection Attack Lab


```
{
    Password: "a5bdf35a1df4ea895905f6f6618e83951a6effc0",
    address: "admin address",
    birth: "3/5",
    eid: "99999",
    email: "admin@example.com",
    id: 1,
    name: "admin",
    nickname: "admin",
    salary: 400000,
    ssn: "43254314"
}
```

SQL Injection Demonstration

Enter Your Name:

Your Name

Start Experiment

Current Student: **Travis** (Change User)

Database Users

ID	Name	EID	Salary	Birth	SSN	Address	Email	Nickname	Phone
1	admin	99999	999999	3/5	43254314			test	
2	alice	10000	999999	9/20	10211002	alice address	alice@example.com	test	123456789
3	boby	20000	999999	4/20	10213352	boby address	boby@example.com	test	123456789
4	ryan	30000	999999	4/10	32193525	ryan address	ryan@example.com	test	123456789
5	samy	40000	999999	1/11	32111111	samy address	samy@example.com	test	123456789
6	ted	50000	999999	11/3	24343244	ted address	ted@example.com	test	123456789

- Login Page
- Update Profile Page

localhost:5000/login?username=admin&Password=5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8 Summarize


```
{
    Password: "353e8061f2befecb6818ba0c034c632fb0bcae1b",
    address: "",
    birth: "3/5",
    eid: "99999",
    email: "",
    id: 1,
    name: "admin",
    nickname: "",
    salary: 999999,
    ssn: "43254314"
}
```

```python
@app.route('/login', methods=['GET'])
def login():
    """
    Login endpoint.
    Expects 'username' and 'Password' as URL parameters.
    This version is intentionally vulnerable to SQL injection.
    """
    username = request.args.get('username', '')
    password = request.args.get('Password', '')
    hashed_pwd = hashlib.sha1(password.encode()).hexdigest()

    #parameterized query to prevent SQL injection
    query = "SELECT id, name, eid, salary, birth, ssn, address, email, nickname, Password FROM credential WHERE name = ? AND Password = ?"

    print("Executing SQL: " + query)  # Debug log

    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute(query, (username, hashed_pwd))
        row = cursor.fetchone()
    except Exception as e:
        conn.close()
        return jsonify({'error': str(e)}), 500
    conn.close()

    if row:
        # Remove the password from the response before sending it back to the client
        safe_row = {k: row[k] for k in row.keys() if k != 'Password'}
        return jsonify(dict(safe_row))
    else:
        return jsonify({'message': 'Authentication failed'})
```

```python
@app.route('/update_profile', methods=['POST'])
def update_profile():
    """
    Update profile endpoint.
    Expects form data: 'id', 'nickname', 'email', 'address', 'Password', and 'PhoneNumber'.
    The new password is hashed with SHA1.
    Note: This SQL query is built using insecure string concatenation and is vulnerable to SQL injection.
    """
    user_id = request.form.get('id', '')
    nickname = request.form.get('nickname', '')
    email = request.form.get('email', '')
    address = request.form.get('address', '')
    password = request.form.get('Password', '')
    phone_number = request.form.get('PhoneNumber', '')

    hashed_pwd = hashlib.sha1(password.encode()).hexdigest() if password else ''

    # Validate user_id is an integer
    try:
        user_id_int = int(user_id)
    except (ValueError, TypeError):
        return jsonify({'error': 'Invalid user id'}), 400

    # paramaterized query to prevent SQL injection
    query = ("UPDATE credential SET nickname = ?, email = ?, address = ?, Password = ?, PhoneNumber = ? "
             "WHERE id = ?")

    print("Executing SQL: " + query)  # Debug log
    conn = get_db_connection()
    cursor = conn.cursor()
    try:
        cursor.execute(query, (nickname, email, address, hashed_pwd, phone_number, user_id_int))
        conn.commit()
    except Exception as e:
        conn.close()
        return jsonify({'error': str(e)})
    conn.close()

    return jsonify({'message': 'Profile updated successfully'})
```