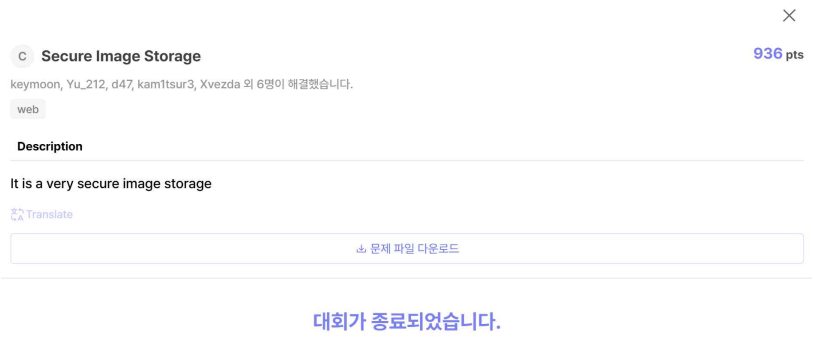
[Dreamhack] Season 6 Round #7
Secure Image Storage

32기 김예진

문제 파일을 다운 받아 보았다.



우선 schema 파일을 열어보았다.

```
flask==3.0.*
requests
selenium==4.18.0
```

이것만 봐서는 무슨 의미인지 이해하지 못했다.

그래서 app 코드 파일을 열어보았다.
코드에 대한 설명은 다음과 같다.

- Database Configuration

```python
def get_db():
    db = getattr(g, '_database', None)
    if db is None:
        db = g._database = sqlite3.connect(DATABASE)
        db.row_factory = sqlite3.Row
    return db
```

현재 애플리케이션의 컨텍스트(g)에 데이터베이스 연결을 저장하고, 필요할 때 해당 연결을 반환한다. 데이터베이스 연결이 없을 경우 새로 연결을 생성한다. sqlite3.connect(DATABSE)로 데이터베이스를 연결하고 g는 애플리케이션 컨텍스트에 속하는 객체로 요청 간에 데이터를 유지한다. row_factory 설정으로, 데이터베이스에서 반환된 결과를 리스트가 아닌 Row 객체 다루어서 더 직관적으로 칼럼 값을 참조할 수 있도록 되어 있다.

- close_conntection(exception)

```python
@app.teardown_appcontext
def close_connection(exception):
    db = getattr(g, '_database', None)
    if db is not None:
        db.close()
```

애플리케이션이 종료될 때 데이터베이스 연결을 닫는다. 애플리케이션 컨텍스트가 끝날 때 호출되면서 열려 있는 데이터베이스 연결을 닫는다.

- init_db()

```python
def init_db():
    with app.app_context():
        db = get_db()
        with app.open_resource('schema.sql', mode='r') as f:
            db.cursor().executescript(f.read())
        db.commit()
```

애플리케이션의 데이터베이스를 초기화한다. schema.sql 파일을 읽어서 SQL 명령어를 실행하여 데이터베이스 테이블을 생성한다. 애플리케이션의 컨텍스트에서 get_db()를 호출하여 데이터베이스 연결을 얻은 후, SQL 스크립트를 실행하여 데이터베이스 스키마를 설정한다.

- get_data(query, args=(), one=False)

```
def get_data(query, args=(), one=False):
    cur = get_db().execute(query, args)
    rv = cur.fetchall()
    cur.close()
    return (rv[0] if rv else None) if one else rv
```

데이터베이스에서 데이터를 조회하는 함수이다. SQL 쿼리를 실행하고, 해당 쿼리의 결과를 반환한다. one=True일 경우, 첫 번째 결과만 반환하고, 그렇지 않으면 모든 결과를 리스트로 반환한다. 결과가 없을 경우 None을 반환한다.

- insert_data(query, args=())

```
def insert_data(query, args=()):
    db = get_db()
    cur = db.cursor()
    cur.execute(query, args)
    db.commit()
    rowcount = cur.rowcount
    cur.close()
    return rowcount
```

데이터베이스에 데이터를 삽입하는 함수이다. SQL 삽입 쿼리를 실행하고, 삽입된 데이터의 수를 반환하는 기능을 한다. 삽입 후 db.commit()으로 변경 사항을 데이터베이스에 커밋하여 저장한다.

- access_file(path, cookie={“name”: “name”, “value”: “value”})

```python
def access_file(path, cookie={"name": "name", "value": "value"}):
    try:
        service = Service(executable_path="/chromedriver-linux64/chromedriver")
        options = webdriver.ChromeOptions()
        for _ in [
            "headless",
            "window-size=1920x1080",
            "disable-gpu",
            "no-sandbox",
            "disable-dev-shm-usage",
        ]:
            options.add_argument(_)
        driver = webdriver.Chrome(service=service, options=options)
        driver.implicitly_wait(3)
        driver.set_page_load_timeout(3)
        driver.get(f"http://127.0.0.1:8000/")
        driver.add_cookie(cookie)
        driver.get(f"http://127.0.0.1:8000/{path}")
        sleep(1)
    except Exception as e:
        print(e, flush=True)
        driver.quit()
        return False
    driver.quit()
    return True
```

Selenium을 이용하여 특정 파일에 접근한다. 주어진 경로에 접근하고 지정된 쿠키를 설정한 후 해당 경로로 이동해 파일에 접근을 시도한다. Chrome 웹드라이버를 headless 모드로 실행해 브라우저 창을 띄우지 않고 페이지를 로드한다. 주어진 쿠키를 추가하고 파일에 접근을 시도한 뒤 성공 여부를 반환한다.

- index()

```python
@app.route('/', methods=['GET'])
def index():
    return redirect('/upload')
```

루트 경로로 요청이 들어오면 /upload 경로로 리디렉션한다. Flask의 redirect() 함수를 이용해 /upload 페이지로 리디렉션하는 것이다.

- upload()

```python
@app.route('/upload', methods=['GET', 'POST'])
def upload():
    if request.method == 'GET':
        return render_template('upload.html')
    else:
        username = request.form['name'].strip()
        if len(username) > 30 :
            return render_template('upload.html', msg= 'The length of the username must be 30 characters or less')
        file = request.files['file']
        if file.filename == '':
            return redirect('/upload')

        if len(path.splitext(file.filename)[0]) > 30:
            return render_template('upload.html', msg= 'The length of the filename must be 30 characters or less')

        if ('.' in path.splitext(file.filename)[0]) or ('/' in path.splitext(file.filename)[0]):
            return render_template('upload.html', msg= 'Not allowed character')

        if path.splitext(file.filename)[1].lower() != '.png':
            return render_template('upload.html', msg= 'Only PNG files are allowed')

        if file.content_type != 'image/png':
            return render_template('upload.html', msg= 'Only PNG files are allowed')

        username = parse.quote(username, safe='')
        filename = username + '_' + parse.quote(path.splitext(file.filename)[0],safe='%')
        uuid = str(uuid4())
        base_path = getcwd()
        full_path = path.join(base_path+'/static/', uuid)
        makedirs(full_path)
        try:
            file.save(full_path + '/' + filename)
        except Exception as e:
            print(e, flush=True)
            return render_template('upload.html', msg= 'Something error')

        date = datetime.now().isoformat()
        encoded_data = (username+filename).encode('utf-8')
        md = md5()
        md.update(encoded_data)
        hash = md.hexdigest()
        insert_data('INSERT INTO files (username, filename, hash, uuid, date) values(?,?,?,?,?)', [username,filename,hash,uuid,date])
        return render_template('upload.html', msg='Upload success')
```

파일을 업로드하는 페이지를 렌더링하거나 파일 업로드 처리를 담당한다. GET 요청일 경우 파일 업로드 양식을 표시한다. POST 요청일 경우 파일을 업로드하고, 사용자가 이름과 파일 이름에 대한 유효성 검사를 수행한 뒤, 파일을 저장하고 정보를 데이터베이스에 기록한다. 파일명과 사용자 이름에 대한 길이 제한, 확장자 제한, 금지된 문자 체크, 그리고 오직 PNG 파일만 허용하는 로직이 포함되어 있다.

- storage(username)

```python
@app.route('/<username>', methods=['GET','POST'])
def storage(username):
    files=get_data('SELECT * FROM files WHERE filename LIKE ?',[username+'_%'])
    return render_template('storage.html', files=files)
```

특정 사용자가 업로드한 파일 목록을 표시한다. 사용자 이름을 기반으로 데이터베이스에서 해당 사용자가 업로드한 파일 목록을 조회한다. 해당 파일을 storage.html 템플릿을 통해 렌더링하여 사용자에게 보여준다.

- show(uuid, filename)

```python
@app.route('/static/<uuid>/<filename>')
def show(uuid, filename):
    uuid_obj = UUID(uuid, version=4)
    if str(uuid_obj) != uuid:
        return 'Not vaild UUID'
    file = get_data('SELECT * FROM files WHERE uuid=?',[uuid], one=True)
    if not file:
        return 'Directory does not exist'
    base_path = getcwd()
    if path.exists(base_path+f'/static/{uuid}/{filename}'):
        res = make_response(send_file(base_path+f'/static/{uuid}/{filename}'))
        # res.headers.add('X-Content-Type-Options', 'nosniff')
        res.headers.add('Content-Type', 'image/png')
        res.headers.add(f'X-Confirmed-{parse.unquote(filename)}', parse.unquote(file[1]) +'_' + file[3])
        for header in res.headers.to_wsgi_list():
            if 'Content-Type' in header[0]:
                if (not header[1].startswith('image/png')) and (not header[1].startswith('application/octet-stream')):
                    return 'Invaild Content-Type'
        return res
    else:
        return 'File does not exist'
```

특정 UUID아 파일 이름에 맞는 파일을 찾아서 다운로드할 수 있도록 한다. 주어진 UUID를 검증한 후, 해당 UUID로 저장된 파일이 있는지 데이터베이스와 파일 시스템에서 확인한다. 파일이 존재하면 해당 파일을 send_file() 함수로 클라이언트에게 전달하고, 없으면 오류 메시지를 반환한다.

- report()

```python
@app.route("/report", methods=["GET", "POST"])
def report():
    if request.method == "POST":
        path = request.form.get("path")
        if not path:
            return render_template("report.html", msg="fail")
        else:
            if access_file(path, cookie={"name": "flag", "value": FLAG}):
                return render_template("report.html", msg="Success")
            else:
                return render_template("report.html", msg="fail")
    else:
        return render_template("report.html")
```

파일에 접근 시도하는 기능을 테스트하는 페이지이다. 사용자가 경로를 입력하면 Selenuim을 사용하여 지정된 경로로 접근을 시도한다. 접근 성공 여부에 따라 성공 또는 실패 메시지를 렌더링하여 사용자에게 보여준다.

문제 풀이를 위해서 서버 링크를 타고 들어가봤다.

가장 먼 뜬 창이다.

파일을 업로드해보려고 아무 파일이나 선택해보았다.

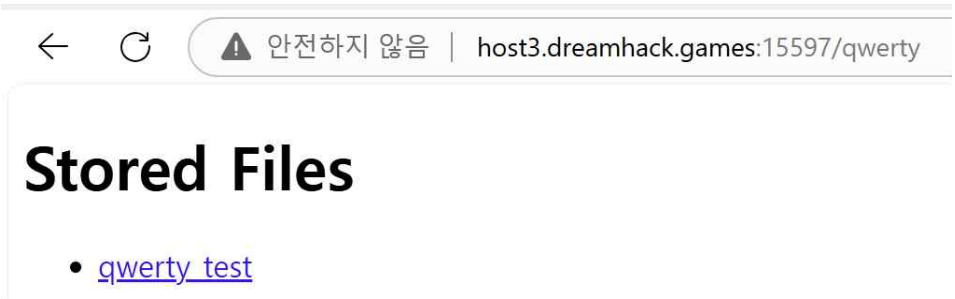
그랬더니 PNG 파일만 가능하다는 문구가 떴다.

그래서 PNG 파일을 업로드해보았다.

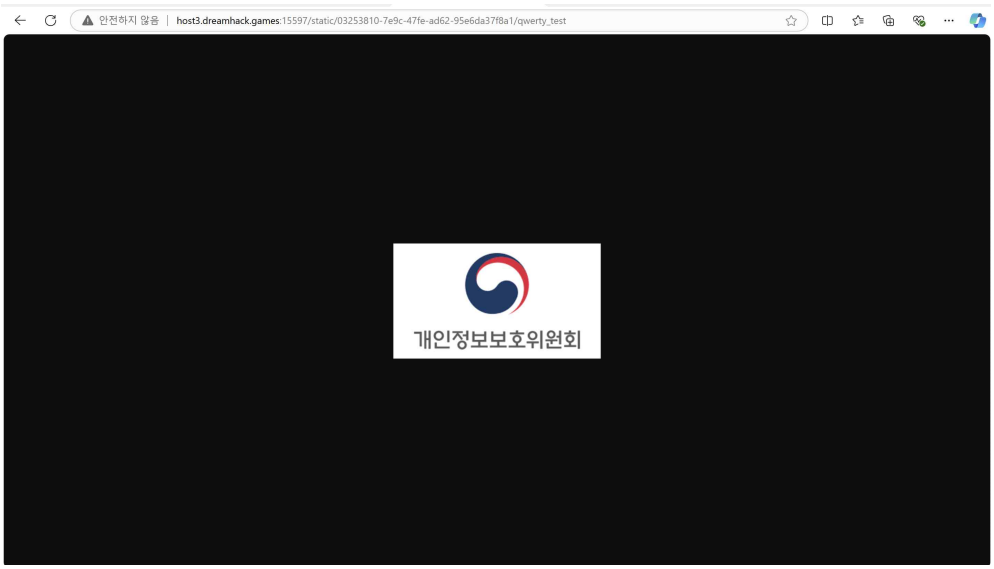
그렇게 했더니 아래와 같이 Upload Success 라고 떴다.

제공된 app 코드를 보고 /qwerty를 입력해보았다.
여기서 qwerty는 내가 설정한 username이다.

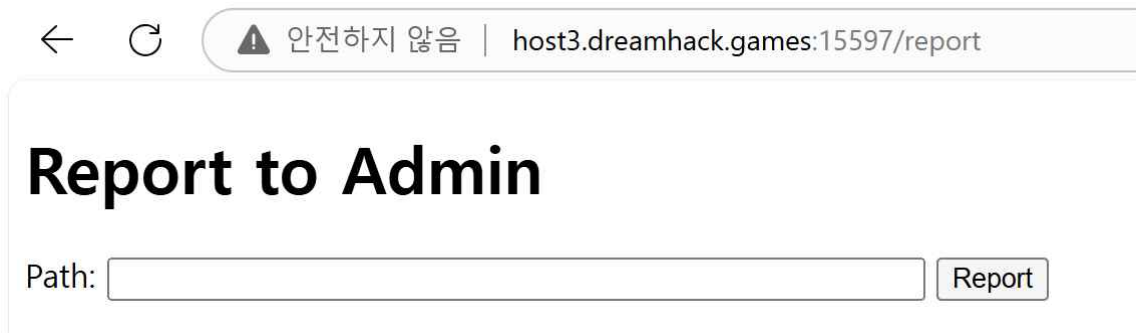
그랬더니 아래와 같이 내가 업로드한 사진 목록이 링크가 걸린 채로 뜬다.



눌렀더니 내가 업로드한 사진이 떴다.

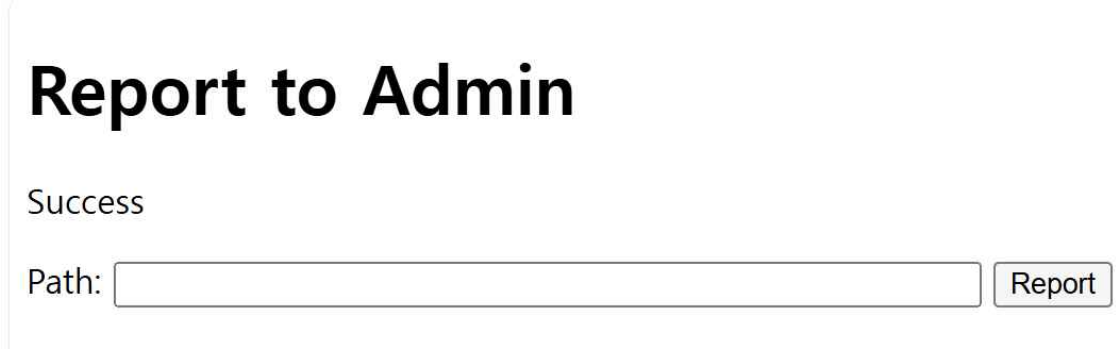


그리고 코드에 따라 /report를 입력해보았더니 Report to Admin 페이지가 떴다.



Path 빈칸에 내가 업로드한 사진의 경로를 입력해보았다.

Report를 누르니까 다음과 같이 Success라는 문구가 떴다.



이 다음부터는 어떻게 해야 할지를 모르겠다..