MK Keamanan Siber (CAK3CAB3)

ProDi S1 Informatika

Semester 2425-1

Laporan Pengerjaan

Tugas 1 - Analisis dan Peningkatan Keamanan Program

Kelas : IF-46-04

Kelompok : 04

Anggota :

No.	NIM	Nama	Kontribusi dalam pengerjaan tugas	Tanda tangan
1.	**Yongky**	**1301223429**	**Mengerjakan semua poin A,B,C,D**	
2.	**Muhammad Rifki Hidayatullah**	**1301220250**	**Mengerjakan semua poin A,B,C,D**	
3.	**Muhammad Fakhrul Hizrian**	**1301223174**	**Mengerjakan semua poin A,B,C,D**	
4.				

A. Instalasi program

Program di-*install* pada: komputer lokal (*native installation*) / *local virtual machine* / cloud / ….
 # pilih salah satu

Disini kami memilih komputer lokal atau native installation

Langkah-langkah instalasi: *# tuliskan tahapan aktifitas Anda dalam meng-install program*

1) Buka github yang sudah diberikan

2) Clone file dari github

3) Ketik 'python -m venv .venv' untuk membuat virtual environtment

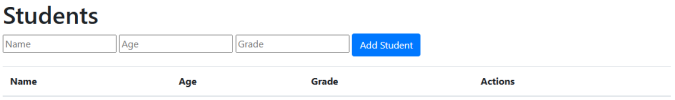
4) Ketik .venv\Scripts\activate untuk mengaktifkan VE yang telah dibuat

5) Ketik 'pip install flask flask_sqlalchemy' untuk menginstal depedensi

6) Ketik 'python app.py' untuk menjalankan aplikasinya

7) Lalu buka 'http://127.0.0.1:5000' di web

Bukti keberhasilan instalasi program: *# bisa dengan screen capture atau lainnya*



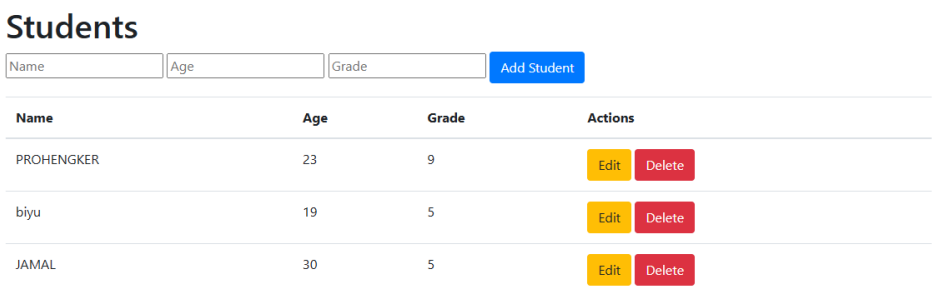

```
(.venv) C:\Users\Yongky\OneDrive\Documents\KAMSIB\python-sqlite>python app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.100.67:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 125-818-038
127.0.0.1 - - [16/Dec/2024 22:10:08] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [16/Dec/2024 22:10:08] "GET /favicon.ico HTTP/1.1" 404 -
```

B. Hasil analisis dan peningkatan keamanan program

Berikut adalah hasil analisis dan peningkatan keamanan program yang telah dilakukan:

1	a	**Celah keamanan** yang teridentifikasi
		
		Deskripsi celah keamanan:
		
		Aplikasi ini tidak memiliki fitur autentikasi dan otorisasi. Siapa pun yang mengetahui URL aplikasi dapat melakukan proses CRUD.
		
		CWE terkait:
		
		CWE-306: *Missing Authentication for Critical Function*
		
		Bagian program yang rentan: *# salin baris (atau baris-baris) program yang terkait dengan kerentanan ini, sertakan nomor barisnya.*
		
		function selectStudentsById($id) {
		  $db = connectDB();
		  if ($db) {
		  $query = "SELECT * FROM student WHERE id=".$id;
		  $result = $db->query($query);
		  if (!$result) {
		  echo "Query gagal: " . $db->lastErrorMsg();
		  return null;
		  }
		  $students = [];
		  while ($row = $result->fetchArray(SQLITE3_ASSOC)) {
		  $students[] = $row;
		  }
		  return $students;
		  } else {
		  echo "Gagal koneksi ke database.";
		  return null;

		} } Berikut merupakan salah satu kode yang rentan (baris 14-42)
	b	**Serangan** yang mengeksploitasi celah keamanan Deskripsi serangan yang dilakukan: *# paparkan detil langkah per langkah cara Anda melakukan serangan untuk mengeksploitasi kerentanan ini* Semua pengguna yang memiliki link tersebut dapat melakukan CRUD tanpa harus diverifikasi Bukti keberhasilan serangan: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat perbedaan yang diakibatkan oleh adanya serangan* **Students** Name Age Grade Add Student Name Age Grade Actions KejuMozarella 22 4 Edit Delete biyu 19 5 Edit Delete Azam 20 4 Edit Delete **Edit Student** JAMAL 30 5 Update



c

Peningkatan keamanan program

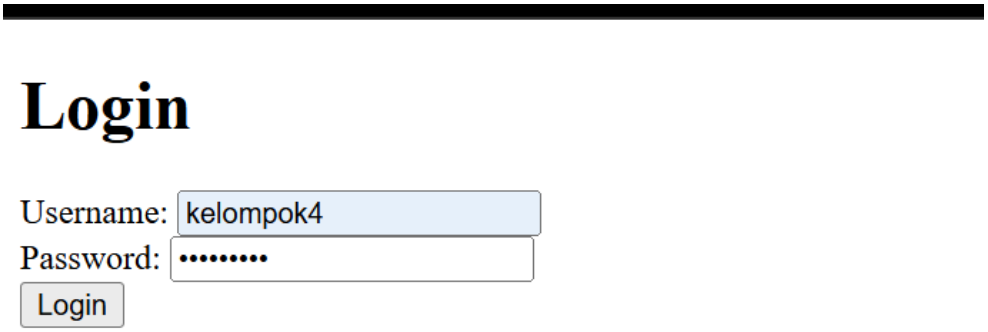
Deskripsi pengamanan (modifikasi terhadap program) yang dilakukan: *# berikan narasi teknik pengamanan yang dilakukan*

Kami menggunakan teknik pengamanan authentikasi

Alasan pemilihan teknik pengamanan: *# jelaskan mengapa teknik ini yang dianggap paling tepat untuk meningkatkan keamanan program*

Teknik pengamanan yang digunakan adalah autentikasi dan otorisasi, karena ini adalah langkah dasar untuk mengontrol akses ke aplikasi dan sumber daya sensitif. Dengan menambahkan autentikasi, kita memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses fungsi aplikasi, seperti CRUD (Create, Read, Update, Delete). Tanpa autentikasi, aplikasi dapat diakses oleh siapa saja yang mengetahui URL, memungkinkan pengguna yang tidak sah untuk mengakses data atau melakukan perubahan yang tidak diinginkan.

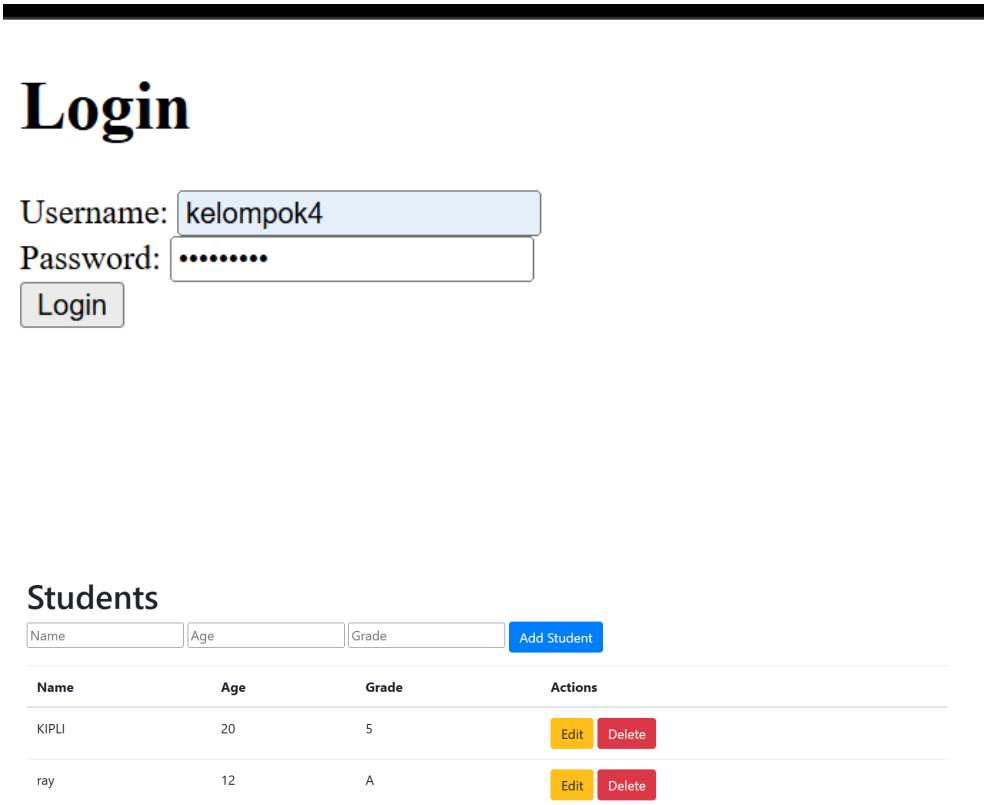
Cuplikan program yang telah diperkuat: *# salin baris (atau baris-baris) program yang diubah, ditambahkan, dan/atau dihapus (jangan betul-betul dihapus, cukup ditandai sebagai komentar saja), dan sertakan nomor barisnya agar mudah ditelusuri*



d

Pengujian keamanan

Hasil penyerangan ulang: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat bahwa serangan tidak berhasil*



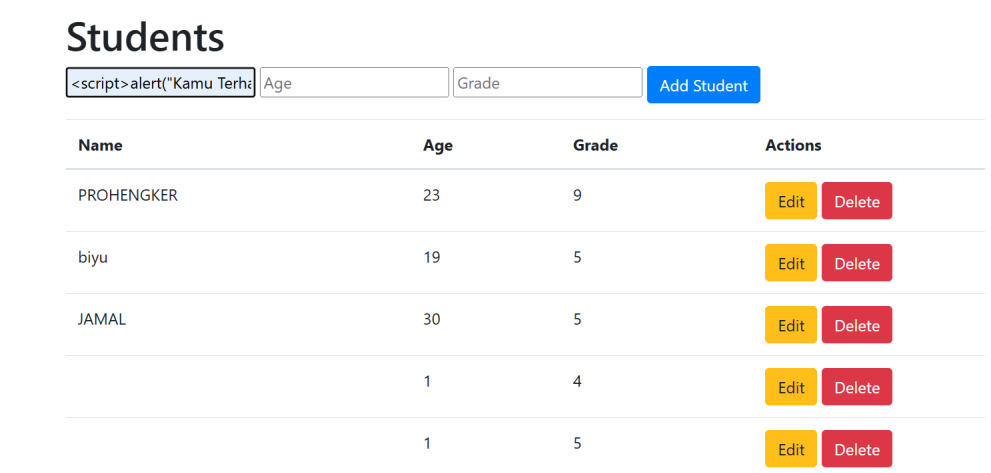
pengguna harus login terlebih dahulu untuk ingin mengedit file
username = kelompok4 dan password kelompok 4

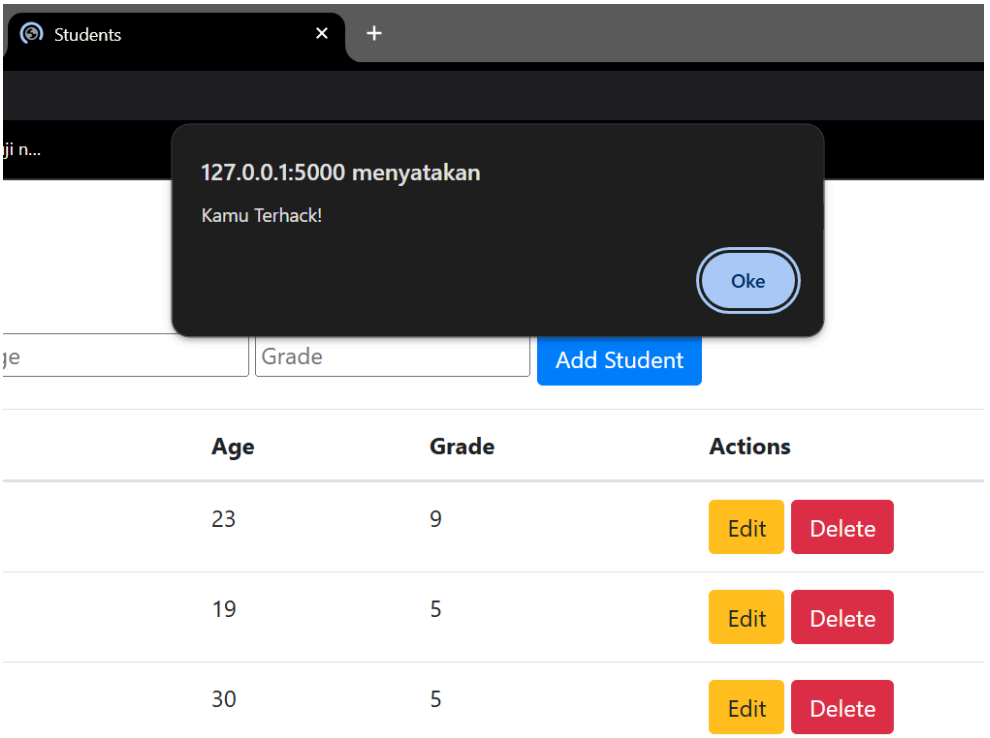
2	a	**Celah keamanan** yang teridentifikasi Deskripsi celah keamanan: Input dari pengguna ditampilkan kembali di halaman index.html tanpa proses sanitasi. Hal ini memungkinkan penyerang menyisipkan kode JavaScript berbahaya ke halaman web yang kemudian dieksekusi oleh browser pengguna. Skrip tersebut juga akan dieksekusi di browser pengguna lain yang membuka halaman tersebut. CWE terkait: CWE-79: *Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')* Bagian program yang rentan: *# salin baris (atau baris-baris) program yang terkait dengan kerentanan ini, sertakan nomor barisnya.* <td>{{ student.name|safe }}</td>
	b	**Serangan** yang mengeksploitasi celah keamanan Deskripsi serangan yang dilakukan: *# paparkan detil langkah per langkah cara Anda melakukan serangan untuk mengeksploitasi kerentanan ini* Penggunaan filter |safe menghilangkan proses sanitasi input, yang berarti konten apa pun yang dimasukkan ke dalam student.name akan dirender ulang di halaman web tanpa pembersihan atau validasi. Hal ini memungkinkan penyerang untuk menyisipkan skrip JavaScript berbahaya. kami menggunakan `<script>alert("Kamu Terhack!");</script>` sehingga saat di masukan akan muncul pesan Bukti keberhasilan serangan: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat perbedaan yang diakibatkan oleh adanya serangan*

Students

Name | Age | Grade | Add Student

Name	Age	Grade	Actions
PROHENGKER	23	9	Edit Delete
biyu	19	5	Edit Delete
JAMAL	30	5	Edit Delete


Students
<script>alert("Kamu Terha
Age
Grade
Add Student
Name
Age
Grade
Actions
PROHENGKER
23
9
Edit
Delete
biyu
19
5
Edit
Delete
JAMAL
30
5
Edit
Delete
1
4
Edit
Delete
1
5
Edit
Delete


Students
ji n...
127.0.0.1:5000 menyatakan
Kamu Terhack!
Oke
ge
Grade
Add Student
Age
Grade
Actions
23
9
Edit
Delete
19
5
Edit
Delete
30
5
Edit
Delete

c

Peningkatan keamanan program

Deskripsi pengamanan (modifikasi terhadap program) yang dilakukan: *# berikan narasi teknik pengamanan yang dilakukan*

Teknik pengamanan yang kami lakukan adalah dengan menghapus penggunaan filter |safe pada code index.html dan menambahkan regular expression (regex) untuk validasi input code untuk nama agar memastikan bahwa semua input pengguna melalui proses sanitasi. Hal ini mencegah ekekusi kode JavaScript yang berbahaya. teknik ini dilakukan dengan cara menghapus code |safe dan menambahkan pattern code required pattern="[a-zA-Z\s]+" title="Only letters and spaces are allowed."> pada file index.html.

Alasan pemilihan teknik pengamanan: *# jelaskan mengapa teknik ini yang dianggap paling tepat untuk meningkatkan keamanan program*

Teknik ini dipilih karena menghilangkan celah keamanan Cross - Site Scripting(XSS) dengan cara memastikan bahwa data yang di render halaman web tidak mengandung kode JavaScript yang berbahaya.

Cuplikan program yang telah diperkuat: *# salin baris (atau baris-baris) program yang diubah, ditambahkan, dan/atau dihapus (jangan betul-betul dihapus, cukup ditandai sebagai komentar saja), dan sertakan nomor barisnya agar mudah ditelusuri*

```
<tbody>
    {% for student in students %}
    <tr>
        <td>{{ student.name|safe }}</td>
        <td>{{ student.age }}</td>
        <td>{{ student.grade }}</td>
        <td>
            <a href="/edit/{{ student.id }}" class="btn
```

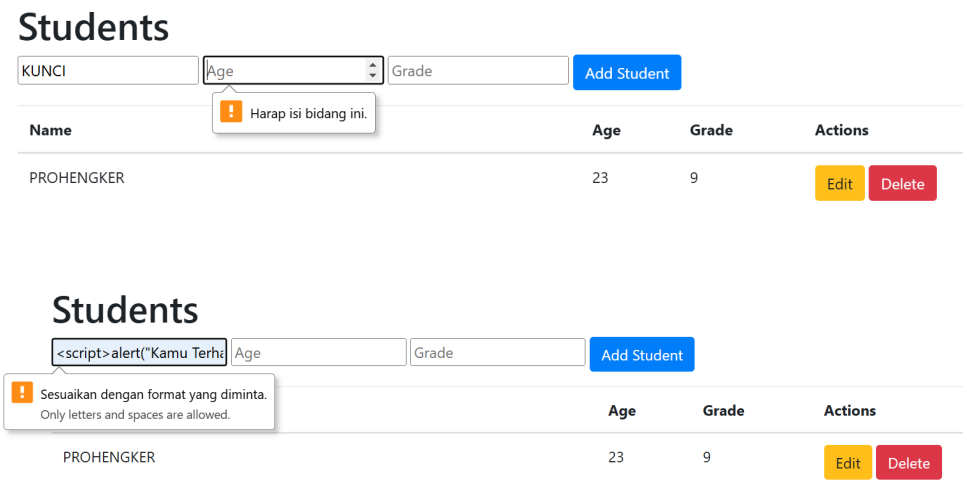
```
{% for student in students %}
<tr>
    <td>{{ student.name}}</td>
    <td>{{ student.age }}</td>
    <td>{{ student.grade }}</td>
    <td>
```

```
<h1>Students</h1>
<form action="/add" method="POST">
    <input type="text" name="name" placeholder="Name" required pattern="[a-zA-Z\s]+"title="Only letters and spaces are allowed.">
    <input type="number" name="age" placeholder="Age" required>
```

d

Pengujian keamanan

Hasil penyerangan ulang: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat bahwa serangan tidak berhasil*



3 a

Celah keamanan yang teridentifikasi

Deskripsi celah keamanan: *# tuliskan kerentanan yang Anda temukan pada program.*

Celah keamanan yang ditemukan adalah potensi **CSRF** (Cross-Site Request Forgery), di mana aplikasi tidak memverifikasi bahwa permintaan yang datang berasal dari sumber yang sah. Dalam kode ini, tidak ada perlindungan terhadap permintaan POST yang dapat dieksploitasi oleh penyerang untuk mengubah data atau melakukan tindakan berbahaya tanpa izin pengguna.

CWE terkait: *# tuliskan Common Weakness Enumeration yang terkait. Daftar lengkap ada di https://cwe.mitre.org/data/definitions/699.html*

CWE-352: **Cross-Site Request Forgery (CSRF)**

Bagian program yang rentan: *# salin baris (atau baris-baris) program yang terkait dengan kerentanan ini, sertakan nomor barisnya.*

```
@app.route('/add', methods=['POST'])
```

```
def add_student():

    name = request.form['name']

    age = request.form['age']

    grade = request.form['grade']

    connection = sqlite3.connect('instance/students.db')

    cursor = connection.cursor()

    query = f"INSERT INTO student (name, age, grade)
VALUES ('{name}', {age}, '{grade}')"

    cursor.execute(query)

    connection.commit()

    connection.close()

    return redirect(url_for('index'))
```

b

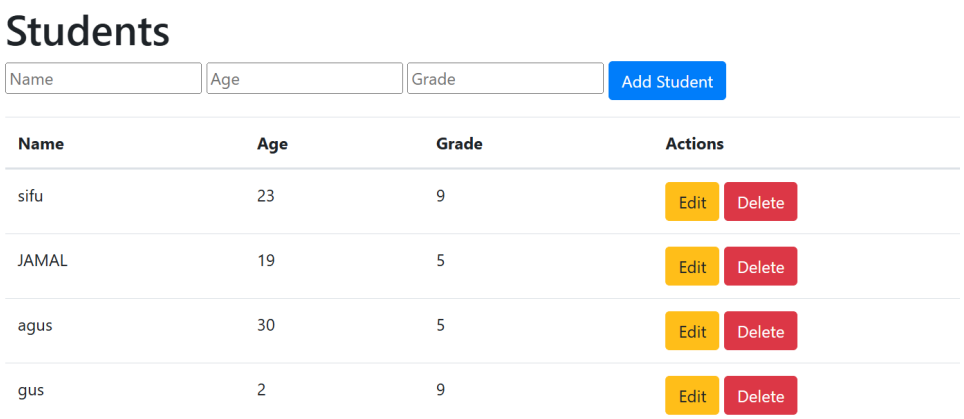
Serangan yang mengeksploitasi celah keamanan

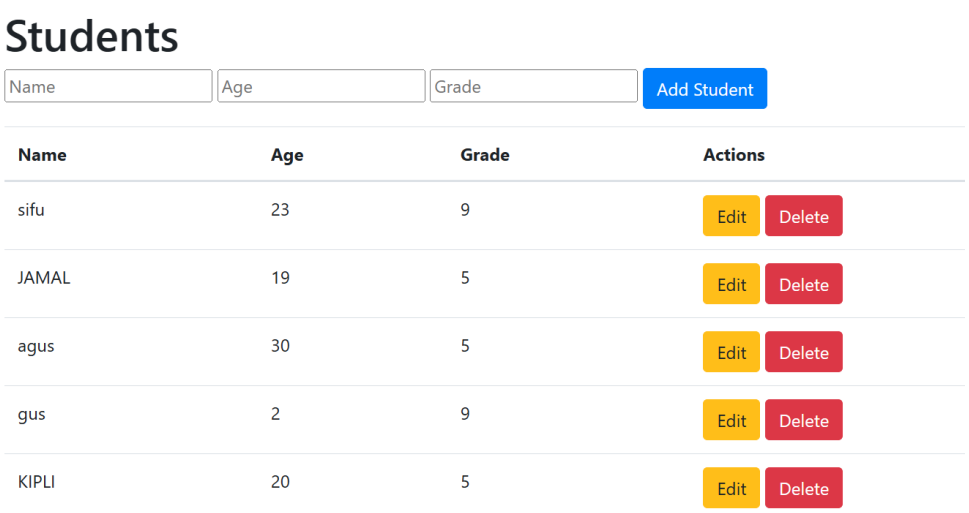
Deskripsi serangan yang dilakukan: *# paparkan detil langkah per langkah cara Anda melakukan serangan untuk mengeksploitasi kerentanan ini*

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>CSRF Attack</title>

</head>

<body>

    <h1>Welcome to the CSRF Page</h1>
```

```
    <p>Click the button to be redirected...</p>

    <form action="http://127.0.0.1:5000/add" method="POST"
style="display:none;">

        <input type="text" name="name" value="KIPLI">

        <input type="number" name="age" value="20">

        <input type="text" name="grade" value="5">

        <input type="submit" value="Submit">

    </form>

    <script>

        document.forms[0].submit();

    </script>

</body>

</html>
```

Bukti keberhasilan serangan: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat perbedaan yang diakibatkan oleh adanya serangan*





c

Peningkatan keamanan program

Deskripsi pengamanan (modifikasi terhadap program) yang dilakukan: *# berikan narasi teknik pengamanan yang dilakukan*

Untuk mengatasi masalah CSRF, kita dapat menambahkan token CSRF ke dalam formulir HTML untuk memastikan bahwa permintaan POST yang diterima adalah sah dan berasal dari pengguna yang sedang login. Flask menyediakan ekstensi Flask-WTF untuk menangani CSRF secara otomatis dengan menambahkan token pada setiap formulir. Ini mencegah penyerang mengirimkan permintaan POST palsu.

Alasan pemilihan teknik pengamanan: *# jelaskan mengapa teknik ini yang dianggap paling tepat untuk meningkatkan keamanan program*

Flask dengan proteksi CSRF adalah pilihan yang tepat karena secara otomatis menambahkan perlindungan CSRF ke semua formulir di aplikasi. Dengan menambahkan token CSRF di setiap permintaan, aplikasi memastikan bahwa setiap permintaan POST yang diterima berasal dari sumber yang sah dan bukan dari halaman berbahaya yang mencoba mengeksploitasi pengguna yang sudah login.

Cuplikan program yang telah diperkuat: *# salin baris (atau baris-baris) program yang diubah, ditambahkan, dan/atau dihapus (jangan betul-betul dihapus, cukup ditandai sebagai komentar saja), dan sertakan nomor barisnya agar mudah ditelusuri*

```
You, 3 minutes ago | 3 authors (MCBPro Pebri and others)
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy import text
from flask_wtf import FlaskForm
from wtforms import StringField, IntegerField
from flask_wtf.csrf import CSRFProtect
from wtforms.validators import DataRequired, NumberRange

app = Flask(__name__)

# Setup Secret Key for CSRF
app.config['SECRET_KEY'] = 'secretkey12345'
csrf = CSRFProtect(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

```
# Flask-WTF Form
You, 3 minutes ago | 1 author (You)
class StudentForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    age = IntegerField('Age', validators=[DataRequired(), NumberRange(min=1)])
    grade = StringField('Grade', validators=[DataRequired()])
```

```
<form action="/add" method="POST">
    <input type="hidden" name="csrf_token" value="{{ csrf_token() }}"/>
```

lakukan pada edit.html dan index.html untuk token crf

d

Pengujian keamanan

Hasil penyerangan ulang: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat bahwa serangan tidak berhasil*

Students

Name | Age | Grade | Add Student

Name	Age	Grade	Actions
sifu	23	9	Edit Delete
JAMAL	19	5	Edit Delete
agus	30	5	Edit Delete
gus	2	9	Edit Delete
KIPLI	20	5	Edit Delete

user add manual:

Students

Name | Age | Grade | Add Student

Name	Age	Grade	Actions
sifu	23	9	Edit Delete
JAMAL	19	5	Edit Delete
agus	30	5	Edit Delete
gus	2	9	Edit Delete
KIPLI	20	5	Edit Delete
ray	12	A	Edit Delete

		saat seseorang ingin menambahkan dengan file html untuk add akan gagal karena tidak memiliki token seperti sebelumnya --- **Bad Request** The CSRF token is missing.

4	a	**Celah keamanan** yang teridentifikasi
		
		Deskripsi celah keamanan: *# tuliskan kerentanan yang Anda temukan pada program.*
		
		Kode ini memiliki potensi kerentanan SQL Injection karena parameter yang diterima dari input pengguna ($_GET['id'] dan $_POST) tidak dilakukan sanitasi atau validasi sebelum digunakan dalam fungsi selectStudentsById($id) dan updateStudent($id, $name, $age, $grade).
		
		CWE terkait: *# tuliskan Common Weakness Enumeration yang terkait. Daftar lengkap ada di https://cwe.mitre.org/data/definitions/699.html*
		
		CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
		
		Bagian program yang rentan: *# salin baris (atau baris-baris) program yang terkait dengan kerentanan ini, sertakan nomor barisnya.*
		
		@app.route('/edit/<int:id>', methods=['GET', 'POST'])
		
		def edit_student(id):
		
		  if request.method == 'POST':
		
		  name = request.form['name']
		
		  age = request.form['age']

		grade = request.form['grade'] # RAW Query db.session.execute(text(f"UPDATE student SET name='{name}', age={age}, grade='{grade}' WHERE id={id}")) db.session.commit() return redirect(url_for('index')) else: # RAW Query student = db.session.execute(text(f"SELECT * FROM student WHERE id={id}")).fetchone() return render_template('edit.html', student=student)
	b	**Serangan** yang mengeksploitasi celah keamanan Deskripsi serangan yang dilakukan: *# paparkan detil langkah per langkah cara Anda melakukan serangan untuk mengeksploitasi kerentanan ini* Deskripsi serangan yang dilakukan: Penyerang mengakses URL dengan parameter injeksi SQL, misalnya: http://127.0.0.1:5000/edit/1 Dalam formulir update, mereka dapat menyisipkan nilai berbahaya seperti: Name: ' OR 1=1 -- Bukti keberhasilan serangan: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat perbedaan yang diakibatkan oleh adanya serangan*

Students

Name | Age | Grade | Add Student

Name	Age	Grade	Actions
jamalidom	23	9	Edit Delete
supri	19	5	Edit Delete
sifu	30	5	Edit Delete
polka	2	9	Edit Delete

Edit Student

' OR 1=1 -- | 23 | 9 | Update

Students

Name | Age | Grade | Add Student

Name	Age	Grade	Actions
1	23	9	Edit Delete
1	19	5	Edit Delete
1	30	5	Edit Delete
1	2	9	Edit Delete

c

Peningkatan keamanan program

Deskripsi pengamanan (modifikasi terhadap program) yang dilakukan: *# berikan narasi teknik pengamanan yang dilakukan*

Untuk menghindari serangan SQL Injection, kami mengganti penggunaan query SQL langsung dengan Prepared Statements dan parameterized queries yang memvalidasi dan menyaring input pengguna secara aman. Hal ini akan menghilangkan risiko manipulasi query melalui input.

Alasan pemilihan teknik pengamanan: *# jelaskan mengapa teknik ini yang dianggap paling tepat untuk meningkatkan keamanan program*

Penggunaan Prepared Statements dan parameterized queries adalah teknik yang tepat karena ini memastikan bahwa input pengguna tidak dapat mengubah struktur query SQL. Teknik ini juga membantu memisahkan data dari kode eksekusi SQL, sehingga mencegah serangan SQL Injection.

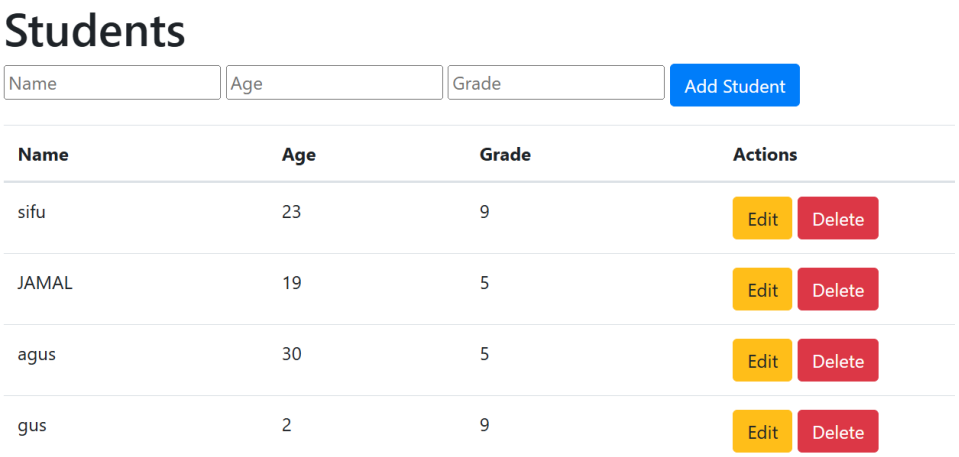
Cuplikan program yang telah diperkuat: *# salin baris (atau baris-baris) program yang diubah, ditambahkan, dan/atau dihapus (jangan betul-betul dihapus, cukup ditandai sebagai komentar saja), dan sertakan nomor barisnya agar mudah ditelusuri*

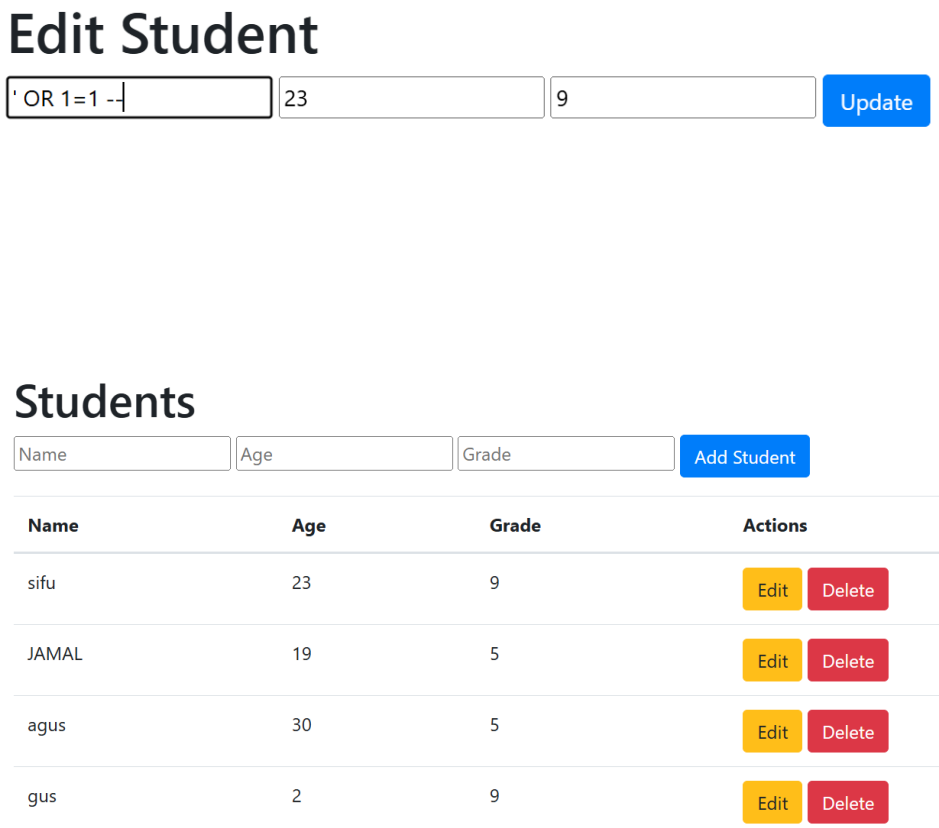
```
# RAW Query
if name.isalpha() and age.isnumeric() and grade.isnumeric():
```

d

Pengujian keamanan

Hasil penyerangan ulang: *# tunjukkan kondisi sistem sebelum dilakukan serangan, proses dilakukannya serangan, dan kondisi sistem setelah dilakukan serangan, sehingga terlihat bahwa serangan tidak berhasil*





C. Refleksi

 # tuliskan apa yang Anda pelajari dari pengalaman pengerjaan tugas ini, baik dari segi pengetahuan, keterampilan, maupun sikap.

 # tuliskan hambatan yang Anda hadapi selama proses pengerjaan tersebut dan bagaimana cara Anda mengatasinya?

Kami mempelajari betapa pentingnya authentikasi dalam suatu data agar tidak sembarangan orang dapat mengakses data tersebut

hambatan yang kami dapat adalah dalam mengerjakan tugas ini adalah dalam membuat fitur token, cara mengatasinya adalah dengan membuat token berdasarkan flask dan mengimport lalu membuat form setelah itu fitur token sudah berhasil

D.

Lampiran:

 # salin kode program lengkap yang sudah dimodifikasi, beri tanda baris-baris yang Anda modifikasi, ditambahkan, dan/atau dihapus (jangan betul-betul dihapus, cukup ditandai sebagai komentar saja)

```
{% for student in students %}
<tr>
    <td>{{ student.name}}</td>
```

menghapus filter |safe

```
rm method="POST">
 <input type="hidden" name="csrf_token" value="{{ csrf_token() }}"/>
```

menambahkan input pada file index.html dan edit.html

```
# Flask-WTF Form
You, 2 hours ago | 1 author (You)
class StudentForm(FlaskForm):
    name = StringField('Name', validators=[DataRequired()])
    age = IntegerField('Age', validators=[DataRequired(), NumberRange(min=1)])
    grade = StringField('Grade', validators=[DataRequired()])
```

```
from flask_wtf import FlaskForm
from wtforms import StringField, IntegerField
from flask_wtf.csrf import CSRFProtect
from wtforms.validators import DataRequired, NumberRange

app = Flask(__name__)

# Setup Secret Key for CSRF
app.config['SECRET_KEY'] = 'secretkey12345'
csrf = CSRFProtect(app)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///students.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)
```

menambahkan form dan import untuk token

```
from functools import wraps
```

```python
# Middleware untuk memeriksa autentikasi
Tabnine | Edit | Test | Explain | Document | Ask
def login_required(f):
    @wraps(f)
    def decorated_function(*args, **kwargs):
        if 'user_id' not in session:
            flash('You need to login first.', 'warning')
            return redirect(url_for('login'))
        return f(*args, **kwargs)
    return decorated_function

# Route login
Tabnine | Edit | Test | Explain | Document | Ask
@app.route('/login', methods=['GET', 'POST'])
def login():
    form = FlaskForm()  # Add this line
    if form.validate_on_submit():  # Change this from request.method == 'POST'
        username = request.form['username']
        password = request.form['password']

        if username == 'kelompok4' and password == 'kelompok4':
            session['user_id'] = username
            return redirect(url_for('index'))
        else:
            flash('Invalid username or password', 'danger')

    return render_template('login.html', form=form)  # Pass form to template
```

membuat login sebelum mengedit data