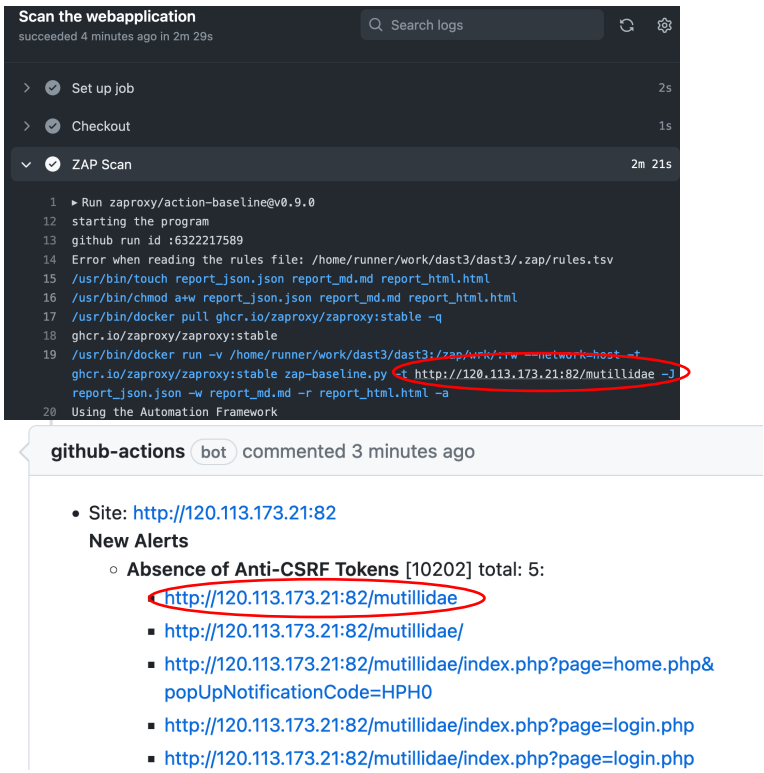
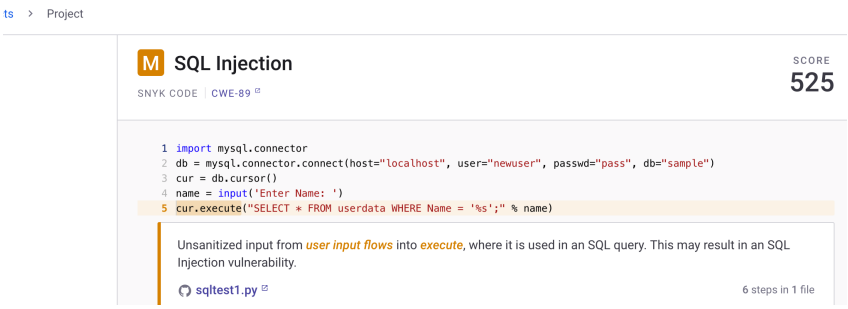
Homework 2: Practice of Web Security & GitHub Actions CI (Continuous Integration) with SAST and DAST

1. CI by GitHub Actions
   (1) Create a repository named XXX_hw2 in your own GitHub account which are imported from the HW2 from teacher's GitHub: https://github.com/wangch64/hw21005.git
   (2) Clone the XXX_hw2 from your GitHub repository to your local machine.
   (3) Modify and add some files (**Copied from ShareFiles**) into the folder XXX_hw2.
   (4) Change a new branch named **newbranch** in your local repository.
   (5) Push your local files to the remove GitHub with the branch: **newbranch**. In Actions you should (on **pull_request**):
      (a) Check the **Maven pom.xml** (**SCA**) by using Snyk
      (b) Check a web-site: **http://120.113.173.21:82/mutillidae** with **DAST: ZAP (baseline scan is faster!)**
      (c) Perform **fuzz test** with Clusterfuzzlite
      (from the source code: https://github.com/wangch64/fuzzapp.git)
   (6) Execute pull_request, **review the security checking results** and **merge the branch**.
   (7) Use Snyk website to check the **python** codes (**SAST**) in XXX_hw2.



(ZAP DAST & Actions Example)



(Snyk SAST Example)

2. Web Security Practice - **OWASP Top 10 (2017)** **A2** – Broken Authentication and **A3** - Sensitive Data Exposure

Create the followings Python file named myform.py

```
import flask
from flask import Flask, render_template, request

app = flask.Flask(__name__)
@app.route("/myform")
def form():
	return render_template('myform.html')

@app.route("/submit", methods=['POST'])
def submit():
	account = request.values['account']
	password = request.values['password']
	if password == "Testtest":
		return render_template('resp.html', **locals())
	else:
		return "<H1> Error Account or Password! <H1>"

if __name__ == '__main__':
	app.run()
```

➢ In **./templates**, you should create two files:

(1) myform.html

```
<!DOCTYPE html>
<html lang="en">
<head>
	<H1> Welcome to my TestSite </H1>
	<form method="POST" action="{{url_for('submit')}}">
		<p><input type="text" class="form-control" name="account"
placeholder="Account">
		<p><input type="text" class="form-control" name="password"
placeholder="Password">
		<p><button type="submit" class="btn btn-primary">Submit</button>
</head>
<body>

</body>
</html>
```

(2) resp.html

```
	<!DOCTYPE html>
	<html lang="en">
	<head>
```

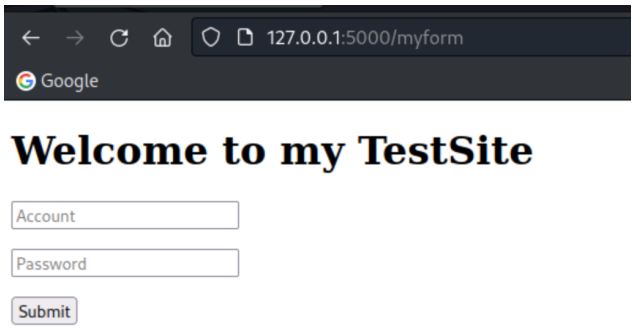
```
        <H1>Welcome {{account}}. Thank you for your login.</H1>
    </head>
    <body>

    </body>
  </html>
```

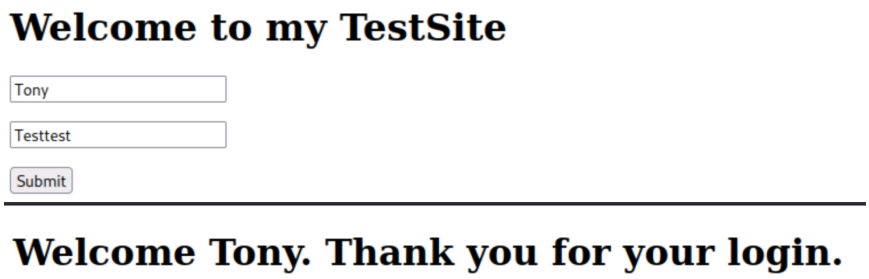
Run the program: python myform.py

```
┌──(one㉿kali)-[~/hw2-2]
└─$ python myform.py
 * Serving Flask app 'myform'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
tead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

Open browser and type: http://127.0.0.1:5000/myform you will see



If you type any account name and the password = Testtest, you will enter the system, as in the following:



Problem: However, **the secret password is written in the python program**. How do you fix it?
if password == "Testtest":

In Flask, there is a package named werkzeug (https://pypi.org/project/Werkzeug/). We can use it to generate the secure hash-code.

- The following is a sample code: **werktest.py**

```
import flask
from flask import Flask, render_template, request
from werkzeug.security import generate_password_hash, check_password_hash

def set_password(password):
    return generate_password_hash(method='pbkdf2:sha512:150000',
password=password)

def check_password(passwordhash, password):
    return check_password_hash(passwordhash, password)

password = "Testtest"
passwordhash = set_password(password)

if check_password(passwordhash, password) is True:
    print("Check OK \n" + password + "\n" + passwordhash)
```

The running result:

```
┌──(one㉿kali)-[~/hw2-2]
└─$ python werktest.py
Check OK
Testtest
pbkdf2:sha512:150000$r7jEGcvslSKgoWZv$f2e0f88fbdc93b44fa23d0ab842392562a4dcdb2c9134
e38b5813494ebd4b05f10d0ce3fe8a1ad92e39b02dab8f822665d
```

<u>The Practice:</u>

(1) How to use **generate_password_hash** and **check_password_hash** in werkzeug package to protect the secret in myform.py program? **Note that you should not write the plain secret and hash code in the program!**

(2) Read technical articles about PBKDF2 from Internet and **<u>write down something you can understand</u>**.