

## 6월 Space Writeup

32기 안현진

‘soeasy’라는 제목의 문제를 풀어보았다. 리버스, pwn 등 여러 분야의 문제가 있었지만, 그나마 조금이라도 시도해 본 분야가 웹 분야라서 이 문제를 선택하게 되었다.

풀 문제를 선택한 뒤, 문제 파일을 다운받았다. 아래에 있는 HTML 문서는 templates 폴더 안에 있는 문서이다. 이제 파일을 하나하나 분석 해 보면서 힌트를 찾아보자.

Challenge7 Solves

soeasy  
988  
easy

exploit may not work all at once

http://war-chall.hspace.io:8000

soeasy-for\_u...

Flag

Submit

templates	파일 폴더	
app	PY 파일	3KB
flag	텍스트 문서	1KB
soeasy-for_user	압축(ZIP) 파일	4KB
admin	Microsoft Edge HT...	2KB
flag	Microsoft Edge HT...	1KB
index	Microsoft Edge HT...	2KB
no_access	Microsoft Edge HT...	1KB

### 1. app-PY 파일

app는 PY파일 유형이다. PY파일은 파이썬 프로그래밍 언어로 작성된 소스 코드를 포함한다. 해당 파일을 열면 긴 코드가 나온다. 하나하나 살펴보도록 하자.

```
1 //import=파이썬에서 다른 모듈을 가져와 사용할 수 있게 하는 키워드
2 import random //random 모듈을 가져오며, random은 다양한 함수를 제공한다
3 import string //문자열 처리를 위해 사용. 특정 문자열 관련 작업을 더 쉽게 해준다
4 import asyncio // 효율적인 비동기 프로그래밍을 할 수 있게 해준다
5 import threading // 멀티스레딩을 지원하는 모듈이다. 병렬처리, I/O 작업을 효율적으로 처리한다

6 app = Flask(__name__) //Flask는 파이썬 코드에서 웹 서버를 실행하고, HTML을 렌더링 할 수 있게 해준다
7 app.config['SECRET_KEY'] = REDACTED //FLASK 애플리케이션의 설정 중 하나이다. 보안 관련 작업에 사용되는 비밀키를 설정하는 부분이다
```

-> 이 부분은(6~7) Flask 앱을 설정하는 부분이다.

-> ‘config’는 Flask 앱의 설정을 관리하는 객체로, Flask 앱의 동작을 조정하는 여러 가지 설정 항목을 포함한다.

-> ‘SECRET\_KEY’는 Flask의 설정 항목 중 하나로, 세션 데이터를 안전하게 저장하기 위해 서명을 추가하는데 사용되거나 CSRF 보호에 사용된다.

```

9     def generate_random_string(length) : //def=파이썬에서 함수를 정의할 때 사용하는 키워드
10         return ''.join(random.choices(string.ascii_letters + string.digits, k = length))

```

-> 랜덤 문자열 생성 함수를 정의하는 부분이다.

-> join은 문자열을 합치는 매서드이다. ' '에 의해 random.choices()에서 반환된 리스트의 각 요소들을 빈 문자열을 사이에 두고 연결한다.

-> 'random.choices'함수는 주어진 시퀀스에서 무작위로 선택된 요소들을 반환하는 함수이다.

-> 괄호 안의 string.ascii\_letters는 두 문자를 통해 예측할 수 있듯, 알파벳의 모든 대소문자를 포함하는 문자열을 의미한다.

-> k = length는 random.choices() 함수가 문자열에서 k 길이 만큼의 요소를 무작위로 선택하여 리스트에 반환하도록 한다. 예를 들어 k=4이면 ['k', 'f', 't', 'b']와 같은 리스트가 생성되고, join에 의해 kftb라는 문자열이 생성된다.

```

12         async def create_file_async() :
13             global data
14             await asyncio.sleep(3)
15             data = generate_random_string(256)
16             return data

```

-> 비동기

파일 생성 함수를 정의하는 부분이다.

-> 'async def'를 통해 async(비동기 함수)를 사용한다. create\_file\_async()라는 이름의 함수를 통해 파일을 생성하는 비동기 작업을 수행한다.

-> 비동기: 병렬적으로 작동하는 방식이다. 특정 코드가 끝날때 까지 코드의 실행을 멈추지 않고 다음 코드를 먼저 실행하는 것을 의미한다.

-> global 키워드를 통해 data를 전역변수로 사용한다. 함수 내부에서 변수를 수정할 때, 함수 바깥의 data 변수를 참조한다

-> await는 비동기 함수 내에서 다른 비동기 함수의 완료를 기다리는 키워드이다. 함수가 실행되면 3초동안 아무 작업도 수행하지 않은 상태로 3초를 대기한다.

-> data = generate\_random\_string: 아까 위에서 본 random 키워드가 보인다. 길이가 256인 string문자열을 랜덤으로 생성한다.

```

18     def background(func) :
19         loop = asyncio.new_event_loop()
20         asyncio.set_event_loop(loop)
21         loop.create_task(func())

```

-> def background를 통해 백그라운드 작업 실행 함수를 정의한다.

```

23     @app.route('/', methods = ['GET'])
24     def form() :
25         return render_template('index.html')

```

-> 이후 나오는 문장들은 라우팅 설정을 위해 작성된 코드들이다.

-> @app.route('/')는 decorator이다. decorator는 함수를 더 강력하게 만들어주는 역할을 한다. /는 이 decorator의 인자로 전달된 문자열의 URL경로를 나타낸다. 여기서는 루트 경로를 나타낸다.

-> method=['Get']은 해당 경로에 대해 허용할 HTTP 메서드를 지정한다. 여기서는 GET 메서드만 허용하도록 설정되어있다.(GET= 서버로부터 데이터를 취득/ POST= 서버에 데이터를 추가. 작성 등의 방법을 통해..)

-> render\_template('index.html')도 함수이다. 이 부분은 Flask를 위에서 정의하고 내려왔기에 사용 가능한 부분이다. HTML 템플릿 파일을 렌더링하여 그 결과를 클라이언트에게 전송한다.

```

27     @app.route('/finder', methods = ['POST'])
28     def stage1() :
29         keyword = request.form.get('keyword')
30         arr = ['weather', 'news', 'stocks', 'sports', 'politics']
31
32         blacklist = ['script', 'img', 'W', ' ', 'cat cute', '-', '!', '/', ';', 'frame']
33
34         for word in blacklist :
35             if word in keyword :
36                 return render_template_string(f"<h1>Keyword {word} is not allowed!</h1>")
37
38         if keyword and keyword in arr :
39             return render_template_string(f"<h1>keyword {keyword} found!</h1>")
40         else :
41             return render_template_string(f"<h1>keyword {keyword} is not found!</h1>")

```

-> 이 부분도 데코레이터와 함께 사용되고있다. 이 데코레이터는 /finder 경로로

들어오는 POST 요청을 처리하도록 Flask에게 지시하는 역할을 한다. def를 통해 stage1 함수를 정의한다. 이 함수는 /finder 경로로 들어오는 POST 요청을 처리한다.

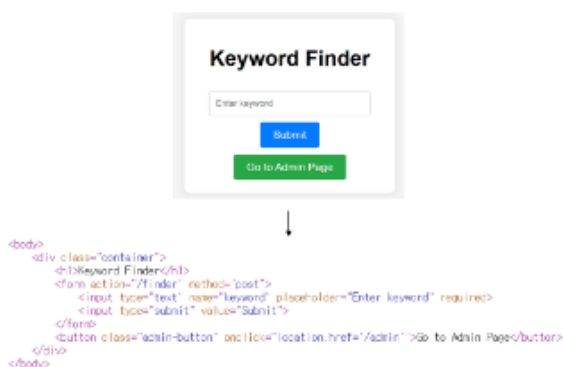
-> request.form.get('keyword'): POST 요청의 폼 데이터에서 keyword라는 이름의 값을 추출한다. 사용자가 입력한 키워드를 가져오는 부분이다.

-> 배열이 나왔다(드디어 내가 아는 부분이다). 배열 arr은 사용자가 입력한 키워드를 검사할 대상 단어 리스트이다. 그리고 blacklist 배열은 사용자가 입력한 keyword가 배열 내에 있는 문자열과 동일하면 에러 메시지를 반환하는데 사용된다. scripts, img 등의 문자열이 블랙리스트에 있는데, 왜 이 단어들을 사용 금지하는지는 모르겠다.

-> for word in blacklist는 for문의 이름?만 봐도 무슨 기능을 할지 알것 같다. if문을 사용해, 사용자가 입력한 단어가 블랙리스트 내의 단어라면 “Keyword {word} is not allowed!” 문장을 출력한다. 위에서 보았던 render\_template('index.html')과 유사한 구조의 문장이 쓰인 것을 보니, render\_template\_string도 템플릿에 의해 사용 가능한 문장인 것 같다.

-> 그 다음의 if문을 살펴보자. 여기서는 사용자가 입력한 keyword가 arr 배열에 있는 단어와 같을 때 출력되는 문장을 정의한다. 여기서도 render\_template\_string를 사용하여, 만약 keyword==arr 배열 내 단어일 경우 “keyword {keyword} found!”라는 문장이 출력되도록 하고 있다. keyword !=arr 배열 내 단어일 경우에는 “keyword {keyword} is not found!”문장을 출력한다.

이쯤 와서 내가 지금 무엇을 위해 app파일을 해석하고 있는지 생각해 보았다. 그리고 나서, app파일은 어디에 쓰이는 파일인지 모르는 상태로 무작정 해석만 하고 있는 듯한 느낌을 받았다. HTML 문서로 작성된 웹을 구성하는 html문서 파일과 app파일은 작성된 내용이 다르다. 그렇다고 app 파일이 HTML 문서로 작성된 웹의 작동 방식과 연관이 없지도 않다.



위의 사진은 사용자로부터 keyword를 입력받고 제출하는 HTML 문서로 작성된 웹 페이지이다. 해당 웹 페이지를 구성하는 HTML문서에 app파일을 불러오는 코드가 있는지 살펴봤지만, app 파일의 언급이 없는 것 같았다. 그래서 “app 파일은 웹 작성을 위한 설명서? 느낌인가?”라는 생각을 했다.

아직 다룰수 있는 프로그래밍 언어도 많이 없고, 기초적인 부분만 배우다보니 flask, templates 등의 응용적인 부분을 이해하는데 어려움이 많았다. 또한 CTF 참여 횟수도 적어, 이렇게 app 파일의 용도같은 기본적인 내용도 이해를 하지 못했다. 위게임이나 마이너 ctf에 많이 참여하여 풀이 경험을 늘려야겠다는 생각이 든다. 일단 app파일은 잠시 두고, 문제 파일 내에 있는 다른 파일들을 살펴보기로 했다.

## 2. flag 파일

해당 파일은 기본적인 텍스트 문서로 작성되었다. 파일을 열어보면 아래 사진과 같은 간단한 문장 한 줄만 출력된다.

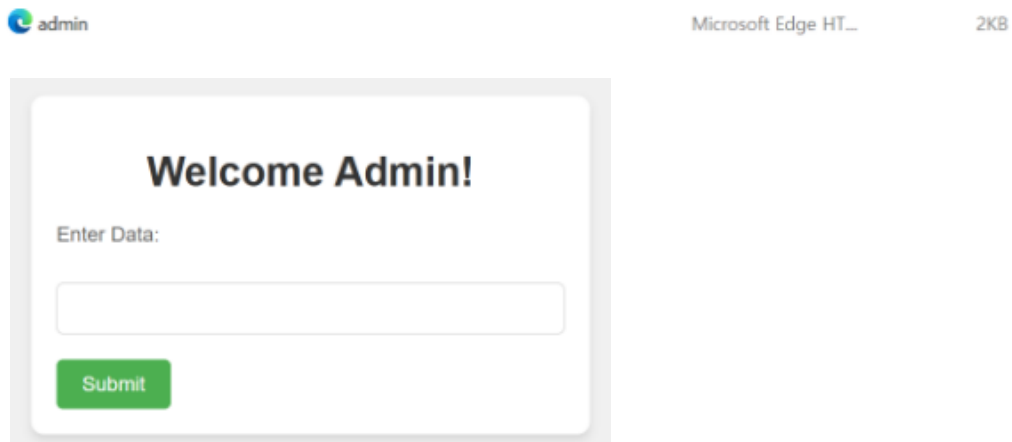


위의 app파일 해석에서 마지막으로 다른 부분 다음에 있는 코드에 REDATED와 관련된 코드가 있어 일단 app 파일의 해당 부분을 가져왔다.

```
43 @app.route('/admin', methods = ['GET', 'POST'])
44 def stage2():
45     cookie = request.cookies.get('cookie')
46     if cookie == "***REDACTED***":
47         background(create_file_async)
48         return render_template('admin.html')
49     else:
50         response = make_response(render_template('no_access.html'))
51         return response
```

-> 아까는 /finder 경로일 경우를 다뤘다면, 여기서는 /admin경로일 때를 다룬다. admin은 ‘운영자 계정’을 의미한다. admin 경로로 어떻게 들어갈 수 있을지 생각해 보았는데, 아까 다운로드 받은 파일 내에 admin관련 페이지로 연결시켜주는

HTML파일이 있었던 것 같다. 해당 파일을 웹과 연결하면 다음과 같은 화면이 뜬다.



The screenshot shows a web browser window with the address bar displaying 'admin' and 'Microsoft Edge HT...'. The page content includes a 'Welcome Admin!' heading, a label 'Enter Data:', a text input field, and a green 'Submit' button.

-> 여기에 flag파일과 app파일에서 공통적으로 보이는 '\*\*\*REDACTED\*\*\*'문장을 입력해보자. 무슨 이유인지는 모르겠지만 에러가 뜬다.. 이 방법이 아닌가...?

flag 파일과 app파일을 살펴보았지만 별 소득은 없었다. 두 파일을 제외하면 남은건 tempalates 파일에 존재하는 HTML 문서들 밖에 없는데... 내가 어느 부분을 놓치고 있는건지 잘 모르겠다.