Dreamhack csrf-1

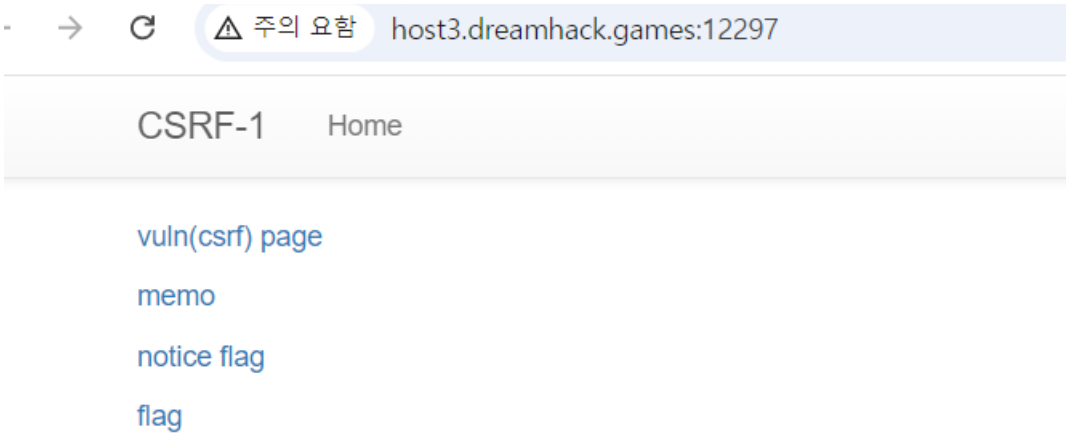
먼저 소스 코드 파일을 열어 분석해보았다.

```python
def read_url(url, cookie={"name": "name", "value": "value"}):
    cookie.update({"domain": "127.0.0.1"})
    try:
        service = Service(executable_path="/chromedriver")
        options = webdriver.ChromeOptions()
        for _ in [
            "headless",
            "window-size=1920x1080",
            "disable-gpu",
            "no-sandbox",
            "disable-dev-shm-usage",
        ]:
            options.add_argument(_)
        driver = webdriver.Chrome(service=service, options=options)
        driver.implicitly_wait(3)
        driver.set_page_load_timeout(3)
        driver.get("http://127.0.0.1:8000/")
        driver.add_cookie(cookie)
        driver.get(url)
    except Exception as e:
        driver.quit()
        print(str(e))
        # return str(e)
        return False
    driver.quit()
    return True
```

- URL, cookie를 인자로 받음, cookie를 받지않은 경우 코드에 있는 기본값이 적용됨.
- selenium 모듈을 사용하여 함수를 실행할 때 입력받은 url에 접속하고 쿠키를 추가한 후 브라우저에서 해당 URL을 로드함.
- 정상적으로 실행된 후 브라우저가 종료되면 True를 리턴.
- 예외 발생 시 브라우저를 종료하고 오류 메시지를 출력.

```python
def check_csrf(param, cookie={"name": "name", "value": "value"}):
    url = f"http://127.0.0.1:8000/vuln?param={urllib.parse.quote(param)}"
    return read_url(url, cookie)
```

- param, cookie를 인자로 받음, cookie를 받지 않은 경우 코드에 있는 기본값이 적용됨.
- 127.0.0.1:8000/vuln?param=urlib(모듈을 통해 URL 인코딩된 텍스트)를 url 변수에 저장.
- read_url 함수를 실행. 이때 위에서 만든 이 인자값으로 전송됨.

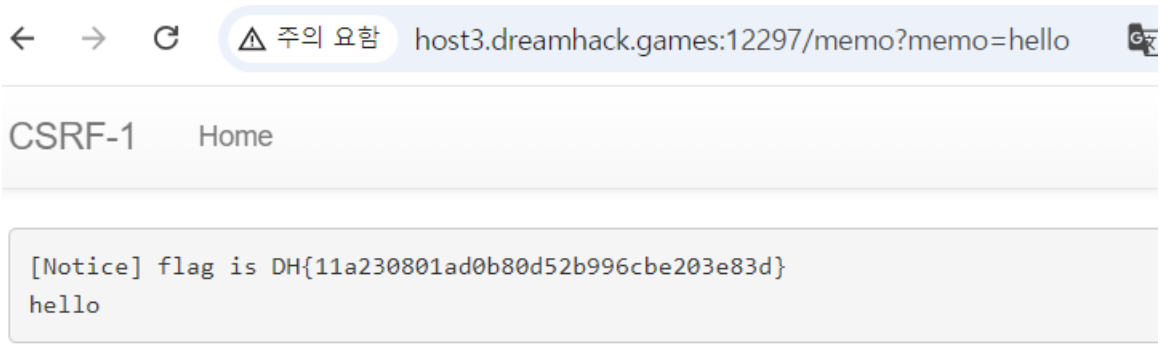


서버를 열어보면 이렇게 되어 있는 웹사이트가 나온다. 각 페이지에 대해 분석해보았다.

- @app.route("/"): 기본 경로(/)로 접근 시 index.html 템플릿을 렌더링한다.
- @app.route("/vuln"): 특정 파라미터(param)를 받아서 필터링한 후 반환한다. 이 필터링은 XSS 공격 방어를 위한 것으로, frame, script, on 등의 문자열을 *로 치환한다.
- @app.route("/flag", methods=["GET", "POST"]): GET 요청 시 flag.html 템플릿을 렌더링하고, POST 요청 시 CSRF 체크 후 결과에 따라 다른 메시지를 반환한다. False가 나오면 wrong?? 이라는 alert을 띄우며 이전 페이지로, True가 나오면 good이라는 alert을 띄우며 이전 페이지로 이동한다.
- @app.route("/memo"): 메모 기능을 제공하며, GET 요청으로 전달된 memo 파라미터 값을 누적하여 HTML 페이지로 출력한다.
- @app.route("/admin/notice_flag"): 이 경로는 admin만 접근할 수 있도록 설계되어 있다. 클라이언트의 IP가 127.0.0.1이 아니면 접근을 차단한다. 또한, userid 파라미터가 admin이 아닌 경우에도 접근이 차단된다. 성공적으로 접근하면 플래그를 메모에 추가한다.

XSS 워게임과 웹사이트 구성, 소스 코드가 유사하지만 XSS와 달리 admin이 /vuln페이지를 방문할 경우 flag값을 출력하는 페이지로 요청을 보내도록 해야 한다. Flag값을 얻기 위한 경로는 /admin/notice_flag?userid=admin으로 접근해야 한다. XSS-2에서 쓴 방법을 쓰긴 어려울 거 같아

서 인터넷 서치를 해 본 결과 img 태그를 이용하면 페이지에 접속할 수 있다고 한다. 예를 들어 이 문제에서 <img src=/admin/notice_flag?userid=admin>이라는 코드를 작성할 시 img 태그가 실제 이미지는 아니지만 해당 태그 때문에 이미지를 불러오려는 시도를 하게 되는데, 이때 공격이 성공할 수 있다고 한다. Flag 페이지에 해당 코드를 입력해보았다.



이후 memo 페이지에서 플래그를 확인할 수 있다. 플래그는 **DH{11a230801ad0b80d52b996cbe203e83d}** 이다.