웹 해킹

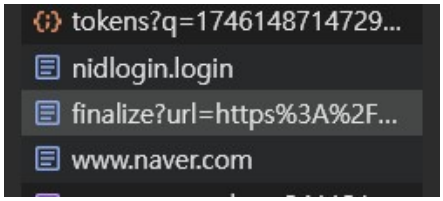
32기 문서현

서론

저번 시간에 http와 웹에 대해 공부하였고, F12를 이용하여 개발자 도구를 켠 후 실습을 해보았다. 또한 개발자 도구를 이용해서 플래그를 찾는 실습도 해봤고, 쿠키를 공부하면서 쿠키 실습을 해보았다. 쿠키와 세션이 활용되는 문제를 통해 새롭게 문제를 푸는 방법에 대해서도 알게 되었다.

추가로 첫 SSR에 하지 못했던 로그인 응답 확인에 성공했다.

나는 처음에 로그인을 한 후에 개발자 도구를 이용해서 로그인이 되었는 지 확인해보려고 했다. 그런데 이 점이 잘못된 것이었다. 개발자 도구를 켠 후 로그인을 하니 아래와 같이 정상적으로 로그인 응답을 확인할 수 있었다.



이번 시간에는 XSS에 대해 공부한 후, 관련 문제를 풀어보았다. 그 문제들을 풀어 본 후 다른 방법이 더 있는 지 추가로 생각해보았고, “파싱”이라는 것을 처음 공부해봤다.

본론

1-1. ClientSide(클라이언트 사이드)

클라이언트 사이드는 웹 페이지 이용자를 대상으로 세션, 쿠키 등의 정보를 빼앗고 그 계정을 통해 임의의 기능을 수행함으로써 공격할 수 있다. 클라이언트 사이드의 대표적인 공격에는 XSS(Cross Site Scripting)이 있다.

1-2. XSS(Cross Site Scripting)

만약에 A라는 웹 페이지가 있다고 하면 A에 XSS 취약점이 존재하면 공격자는 A에 악성 스크립트를 삽입할 수 있게 된다. 그 후 사용자들의 평소와 같이 A에 접속하기만 했는데도 공격자가 삽입한 악성 스크립트에 의해 세션, 쿠키 등의 정보를 빼앗길 수 있다.

종류	설명
Stored XSS	XSS에 사용되는 악성 스크립트가 서버에 저장되고, 서버의 응답에 담겨오는 XSS
Reflected XSS	XSS에 사용되는 악성 스크립트가 URL에 삽입되고, 서버의 응답에 담겨오는 XSS
DOM-based XSS	XSS에 사용되는 악성 스크립트가 URL Fragment에 삽입되는 XSS
Universal XSS	클라이언트의 브라우저 혹은 브라우저의 플러그인에서 발생하는 취약점으로 SOP 정책을 우회하는 XSS

1-3. XSS의 종류 자세히 살펴보기

Stored XSS는 게시물이나 댓글에 악성 스크립트를 삽입하여 공격하는 방식으로 불특정 다수에게 보여지기 때문에 파급력이 높다. 단순히 조회를 하면 공격을 당한다.

Reflected XSS는 검색창에 스크립트를 포함하여 검색하는 것만으로도 공격을 당하는 방식이다. 이 방식은 URL과 같이 이용자가 직접 요청하여야 하기 때문에 Stored XSS에 비해 파급력은 낮다. 그렇기 때문에 링크를 직접 전달하거나 다른 취약점과 연계하여 이용자에게 이 스크립트를 검색하도록 유도한다.

**XSS 취약점이 클라이언트, 서버 둘 다 발생한다.

2-1. 관련 문제 풀어보기1

문제 설명

여러 기능과 입력받은 URL을 확인하는 봇이 구현된 서비스입니다.

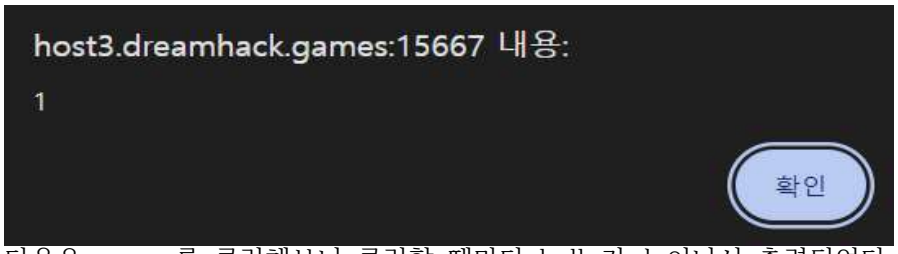
XSS 취약점을 이용해 플래그를 획득하세요. 플래그는 flag.txt, FLAG 변수에 있습니다.

플래그 형식은 DH{...} 입니다.

문제 페이지에 들어가보니 vuln page, memo, flag가 있었다.

우선 무엇인지 알아보기 위해 모두 클릭해봤다.

vuln page를 클릭해보니 1이라는 숫자만 떴다.

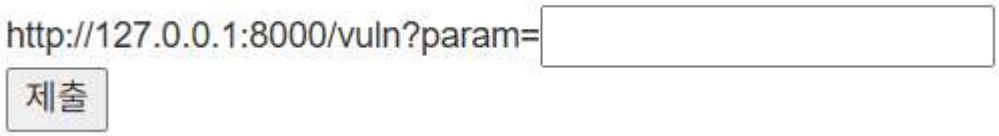


다음은 memo를 클릭해보니 클릭할 때마다 hello가 늘어나서 출력되었다.

```
hello
hello
```

flag를 클릭해보니 http://127.0.0.1:8000/vuln?param= 이렇게 나왔다.

아마 이곳에다가 flag를 입력해야될 것 같다.



다음으로는 같이 주어진 app.py를 살펴보았다.

이곳에 vuln, flag, memo가 모두 있어서 자세히 살펴보았다.

```
@app.route("/vuln")
def vuln():
    param = request.args.get("param", "")
    return param
```

위 사진은 vuln 함수이다. 그런데 “1”이라는 게 없는 데 어떻게 웹 페이지에서 1이 출력되는 지는 잘 모르겠다.

그리고 memo에서도 hello라는 게 없는데 왜 hello가 출력되는 지는 모르겠다.

```
@app.route("/memo")
def memo():
    global memo_text
    text = request.args.get("memo", "")
    memo_text += text + "₩n"
    return render_template("memo.html", memo=memo_text)
```

그래도 memo_texe += text + “\n”을 통해서 클릭할 때마다 반복적으로 hello가 늘어난다는 것은 알겠다.

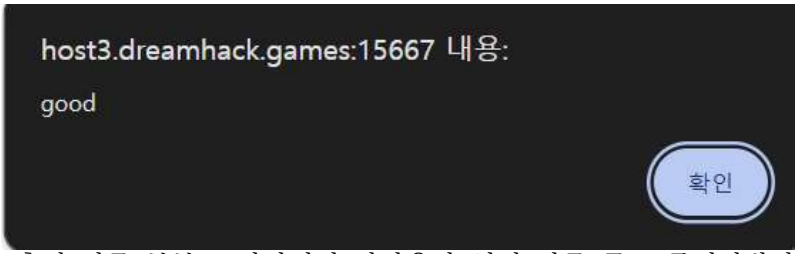
마지막으로 flag함수를 알아보았다.

```
@app.route("/flag", methods=["GET", "POST"])
def flag():
    if request.method == "GET":
        return render_template("flag.html")
    elif request.method == "POST":
        param = request.form.get("param")
        if not check_xss(param, {"name": "flag", "value": FLAG.strip()}):
            return '<script>alert("wrong??");history.go(-1);</script>'

        return '<script>alert("good");history.go(-1);</script>'
```

내가 보기에 이 코드는 데이터를 요청할 때는 get을 사용하고, 데이터를 새로 생성하거나 업데이트할 때는 post를 사용한다는 특성을 이용해서 분석해보면 만약에 데이터를 요청하면 flag를 출력하고, 데이터를 새로 만들거나 업데이트하면 param를 얻을 수 있는 것 같다.
그리고 param을 얻게 된 후 name이 flag이고, value가 flag 값이면 good을 출력하고 그렇지 않으면 wrong??을 출력하는 것 같다.

우선 파이썬 코드에 0.0.0.0이 들어간 부분이 있어서 flag에 들어간 후 0.0.0.0을 입력해봤다.
그러니 good이 출력되었다.



혹시 다른 부분도 달라지지 않았을까 싶어 다른 곳도 들어가봤다. -> 동일했다.
이후에 8000, param 등등 다양한 것을 입력해봤는데 flag를 찾는데 실패했다.
그래서 검색을 해보니
<script>location.href = "/memo?memo=" + document.cookie;</script>를 입력해야 flag를 찾을 수 있다고 나왔다.
그런데 나는 이것이 이해가 되지 않아 하나하나 분석해보기로 했다.
우선 앞 뒤에 있는 <script>와 </script>는 사이에 자바스크립트를 넣는다고 알려주는

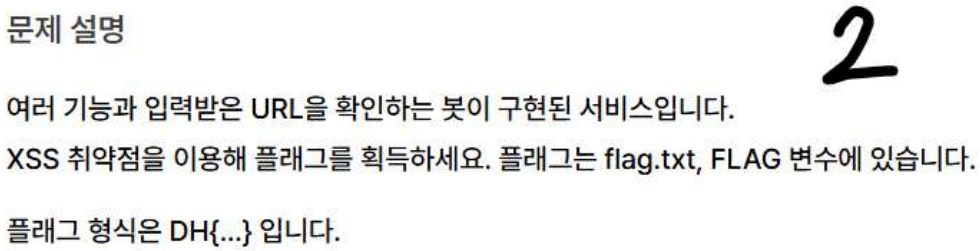
명령어이고, location href는 주소를 강제로 이동시키는 것이라고 한다. /memo?memo=는 새로운 주소에서 알게된 것을 memo 페이지에 남겨주는 것이고, document.cookie는 모든 쿠키값을 보여준다고 한다. 그래서 memo 페이지에는 cookie의 모든 값(즉, flag를 포함한 값)이 남게된다.
<script>location.href = "/memo?memo=" + document.cookie;</script> 이것을 입력했을 때도 good이 나왔다. 이번에는 memo창을 확인해봐야겠다.

```
flag=DH{2c01577e9542ec24d68ba0ffb846508e}
hello
```

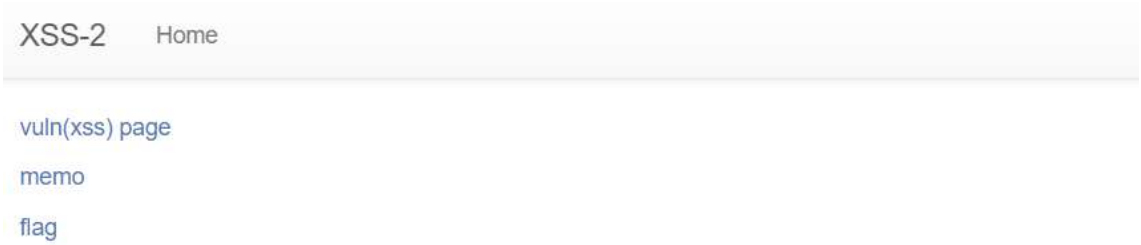
플래그 값이 나왔다!

이것을 한 줄로 요약하면 lcoation.href를 이용해서 document.cookie로 이동한 뒤 memo에 flag를 저장해 flag를 찾는 문제이다.

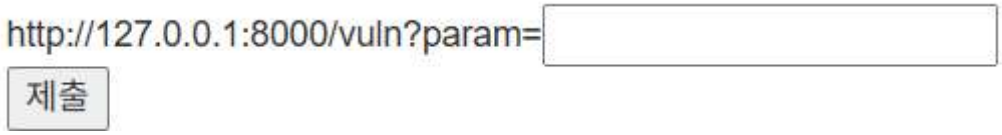
2-2. 관련 문제 풀어보기2



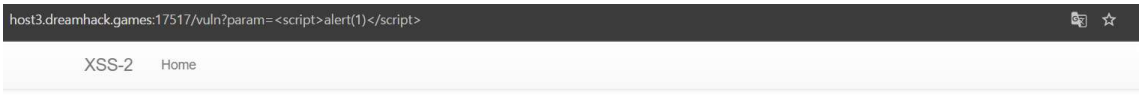
문제 설명은 2-1에서 풀었던 문제와 동일하다.
2-1에서 푼 방식대로 문제 페이지에 먼저 접속을 해보겠다.
문제 페이지의 구성도 2-1과 동일했다.



혹시 몰라 하나하나 클릭을 해서 바뀐게 있나 확인해보았다.
memo와 flag는 2-1과 동일하였다. memo는 접속할 때마다 hello가 하나씩 늘어났고, flag도 2-1과 동일하게 param의 값을 제출할 수 있게 만들어져있었다.



아마 2-1처럼 param에 값을 입력하면 memo에 값이 나올 것이라고 생각해볼 수 있다. 그런데 2-1과 유일하게 다른 부분이 하나 있다. 바로 vuln page 부분이다. 아까는 1이 출력되었는데 이번에는 아무것도 출력되는 것은 없고, 그냥 흰 화면으로 이동하기만 하였다.



이번에는 파이썬 코드를 확인해봐야겠다. 특히 vuln page부분을 중점으로 봐야될 것 같다.

```
@app.route("/flag", methods=["GET", "POST"])
def flag():
    if request.method == "GET":
        return render_template("flag.html")
    elif request.method == "POST":
        param = request.form.get("param")
        if not check_xss(param, {"name": "flag", "value": FLAG.strip()}):
            return '<script>alert("wrong??");history.go(-1);</script>'

        return '<script>alert("good");history.go(-1);</script>'


memo_text = ""


@app.route("/memo")
def memo():
    global memo_text
    text = request.args.get("memo", "")
    memo_text += text + "₩n"
    return render_template("memo.html", memo=memo_text)
```

확인해보니 flag부분과 memo부분은 2-1과 동일했다.

다음은 vuln 부분을 살펴보기로 했다.

```
@app.route("/vuln")
def vuln():
    return render_template("vuln.html")
```

2-1에서는 요청 받은 값을 return 했는데 이번 문제에서는 render_template을 했다. 뭔지 몰라서 찾아보니 쉽게 말해 “vuln.html”파일을 출력하는 것이라고 말할 수 있다.
그래서 이 문제에서 중요한 것은 vuln.html이 아닐까? 라는 생각이 들었다. 이번에도 어떻게 해야될지 하나도 몰라서 검색을 해봤다.
검색을 해보니
<img src="XSS-2" onerror="location.href='/memo?memo='+document.cookie">를 입력하면 플래그 값이 나온다는 것을 알게 되었다. 이것을 분석해보면 location.href='/memo?memo='+document.cookie 부분은 아까와 동일하다. 즉, document.cookie로 강제 이동해서 그곳에서 나온 값을 memo에 출력하라는 말이다.
이제 앞 부분을 해석해봐야겠다. img src="XSS-2" onerror 부분에서 img src=“XSS-2”는 XSS-2에 있는 이미지를 가지고 와서 출력하라는 것이다. 그런데 XSS-2에는 이미지가 존재하지 않는다. 이때 onerror이 실행된다. onerror은 해석한 "location.href='/memo?memo='+document.cookie" 부분이다.
쉽게 말해 존재하지 않는 이미지를 가져다 실행시키고, 실행되지 않으니 onerror을 실행하라는 것이다. 아까와 같이 document.cookie에 있는 플래그를 memo에 출력하게 만드는 코드여서 플래그를 찾을 수 있다.
flag 탭에서
<img src="XSS-2" onerror="location.href='/memo?memo='+document.cookie">를 입력하니 good이라는 단어가 출력되어서 종료하고 memo에 들어가보았다.

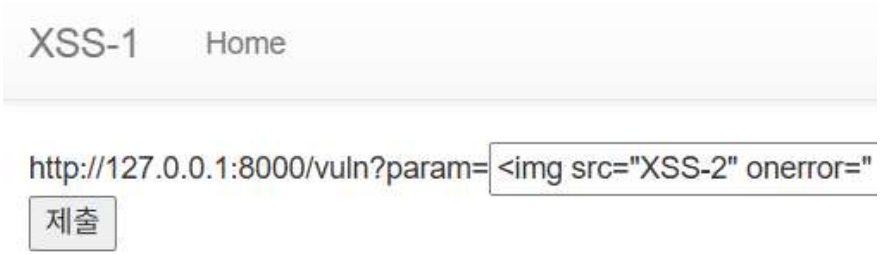
```
hello
hello
hello
hello
flag=DH{3c01577e9542ec24d68ba0ffb846508f}
hello
```

그러니 플래그의 값을 찾을 수 있었다!

그 후 왜 이 문제에서 2-1과 같이 <script>location.href = "/memo?memo=" + document.cookie;</script>를 쓰지 않고 문제를 푸는 이유는 무엇일 지에 대해 찾아보기로 했다.
<script>는 필터링 될 확률이 매우 높다고 한다. 그래서 필터링을 우회해서 쓰면 플래그를

찾을 확률이 더 높아지기 때문에 실무에서는 보통 <script>를 쓰지 않고, 우회해서 많이 쓴다고 한다.

이것에 대해 배운 뒤 또 하나의 궁금증이 생겼다.
아까 2-1에서 풀었던 문제에 2-2 방식대로 풀면 문제가 풀릴 지가 궁금해졌다.
문제를 풀어보기 전에 추측해보자면 플래그를 찾을 수 있을 것 같다.

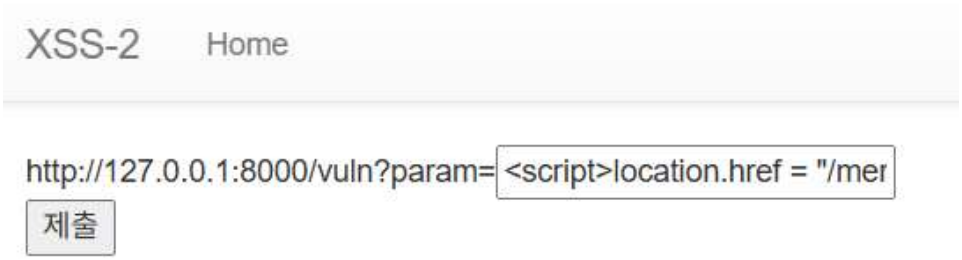


일단 이렇게 입력한 후 제출을 누르니 good이 나왔다. memo에는 flag가 생성되었을까?

```
flag=DH{2c01577e9542ec24d68ba0ffb846508e}
hello
```

정상적으로 출력되었다.

그러면 이번에는 반대로 2-1의 방식을 2-2 문제에 넣어보기로 했다.
이번에도 추측해보면 아마 안 될 것 같다.



2-2문제 플래그에 2-1 방식대로 <script>location.href = "/memo?memo=" + document.cookie;</script>를 넣어보았다. 그러니 good이 출력되었고,

```
hello
hello
```

memo에 들어가보니 플래그 값이 나오지 않았다.
이유는 잘 모르겠다. 딱히 script가 필터링 될만한 파이썬 코드는 없는 것 같은데
왜 작동이 안되는 지 잘 모르겠다.

3-1. 파싱

파싱(Parsing)은 문서, HTML 등에서 어떠한 자료가 있을 때, 내가 원하는 정보만 가공하고 추출해서 원하는 때에 불러올 수 있게 하는 것을 말한다. 그래서 원하는 정보를 추출할 수 있기 때문에 빅데이터도 빠르고 효율적으로 분석할 수 있게 만들어준다. 비구조화된 데이터도 파서를 이용해 파싱을 하면 데이터를 구조화시켜서 쉽게 분석할 수 있도록 만들어줘서 효율성을 높여준다.

파싱을 할 수 있도록 하는 프로그램은 파서(Parser)라고 한다. Beautiful Soup, lxml 등과 같은 프로그램도 있고, 파이썬이나 자바를 이용해서 파서를 직접 만들 수도 있다.

3-2. XML 파싱

XML 문서에서 정보를 추출하고 구조화 된 데이터로 변환하거나 검증하는 등의 역할을 하는 것을 XML 파싱이라고 한다. XML은 시스템 간 데이터를 저장하고 전송하는데 사용하는 마크업 언어를 말한다. 여기에서 마크업 언어는 태그를 이용하여 문서의 의미와 구조를 나타내는 언어를 말한다.

3-3. JSON 파싱

JSON 문서에서 정보를 추출하고 구조화 된 데이터로 변환하거나 검증하는 등의 역할을 하는 것을 JSON 파싱이라고 한다. JSON은 웹 애플리케이션에서 일반적으로 사용되는 가벼운 데이터 교환 형식을 말한다.

3-4. HTML 파싱

HTML 문서에서 정보를 추출하고 구조화 된 데이터로 변환하거나 검증하는 등의 역할을 하는 것을 HTML 파싱이라고 한다. HTML은 웹 페이지를 만드는데 사용되는 마크업 언어이다.

결론

<배운 점>
XSS에는 Stored XSS, Reflected XSS ,DOM-based XSS, Universal XSS라는 것이 있다는 것을 알게되었고, 관련 문제들을 풀면서 문제에 어떻게 접근해야될 지 생각하는 것을 배웠다. 또한 파싱이라는 것은 데이터를 구조화시켜 분석하는 데 효율성을 높여준다는 사실도 알게되었다.

<궁금한 점>
문제 풀이과정에서 모르는 게 많았다.

2-1(드림핵 XSS-1) 부분 파이썬 코드를 보았을 때, vuln코드에 “1”이라는 게 없는 데 어떻게 1이 출력이 될까?
memo코드에서는 “hello”가 없는데 어떻게 hello가 출력이 될까?

2-2(드림핵 XSS-2) 부분에서 <script>location.href = "/memo?memo=" + document.cookie;</script>를 넣었을 때 왜 플래그가 출력이 되지 않는 지 모르겠다.
script가 필터링될만한 부분은 보이지 않는 것 같은데 왜 출력이 되지 않을까?

<어려웠던 점>
저번 시간에는 드림핵을 이용해서 공부를 했는데 이제는 드림핵을 이용할 수 없게 되어서 모든 내용을 인터넷을 찾아보거나 책을 보면서 이해하는 것이 힘들었다. 그리고 XSS라는 것이 생소해서 문제를 푸는 과정에서도 어려움이 있었다. 그리고 파싱 실습을 해보고 싶었는데 너무 복잡해서 하지 못한 점이 조금 아쉬웠다.

<다음 제출 계획>
파싱에 대해 조금 더 공부해보면서 파싱 실습을 하고, 웹 해킹의 다른 부분을 공부해야 될지, 아예 주제를 바꿔서 책 한 권을 처음부터 공부해서 SSR을 작성해야 될지 아직 잘 모르겠다. 파싱에 대한 실습을 하고 싶긴 하나, 이해하기가 어려워서 고민이다.

<참고 자료>

드림핵 . (n.d.). https://dreamhack.io/.

틸코 IT/개발용어::파싱?파서?무슨뜻,개발 직무 용어 살펴보기! . (2021). https://tilkoblet.tistory.com/105.

웹 해킹 강좌 ① - 네이버 서버 시간 웹 파싱 (Web Hacking Tutorial #01) . (2016). https://www.youtube.com/watch?v=OuPjoiXq9gg&list=PLRx0vPvlEmdDQxb41uc1G4ecjV-hklFDM&index=1.

웹 해킹 강좌 ② - 네이버 실시간 검색 순위 웹 파싱 (Web Hacking Tutorial #02) . (2016). https://www.youtube.com/watch?v=pOiqKP-J7-0&list=PLRx0vPvlEmdDQxb41uc1G4ecjV-hklFDM&index=2.

파싱 데이터 파싱이란 무엇인가요? . (n.d.). https://www.bureauworks.com/ko/blog/deiteo-pasingiran-mueosinga#:~:text=HTML%20%ED%8C%8C%EC%8B%B1%20HTML%20%ED%8C%8C%EC%8B%B1%20%EC%9D%80%20HTML%20%EB%AC%B8%EC%84%9C%EC%97%90%EC%84%9C,%EC%86%8D%EC%84%B1%EC%9C%BC%EB%A1%9C%20%EB%B6%84%ED%95%B4%ED%95%98%EC%97%AC%20%EA%B4%80%EB%A0%A8%20%EC%A0%95%EB%B3%B4%EB%A5%BC%20%EC%B6%94%EC%B6%9C%ED%95%98%EB%8A%94%20%EC%9E%91%EC%97%85%EC%9D%84%20%EC%9D%98%EB%AF%B8%ED%95%A9%EB%8B%88%EB%8B%A4..

Markup Language . (n.d.). https://terms.naver.com/entry.naver?docId=3479216&cid=58439&categoryId=58439.