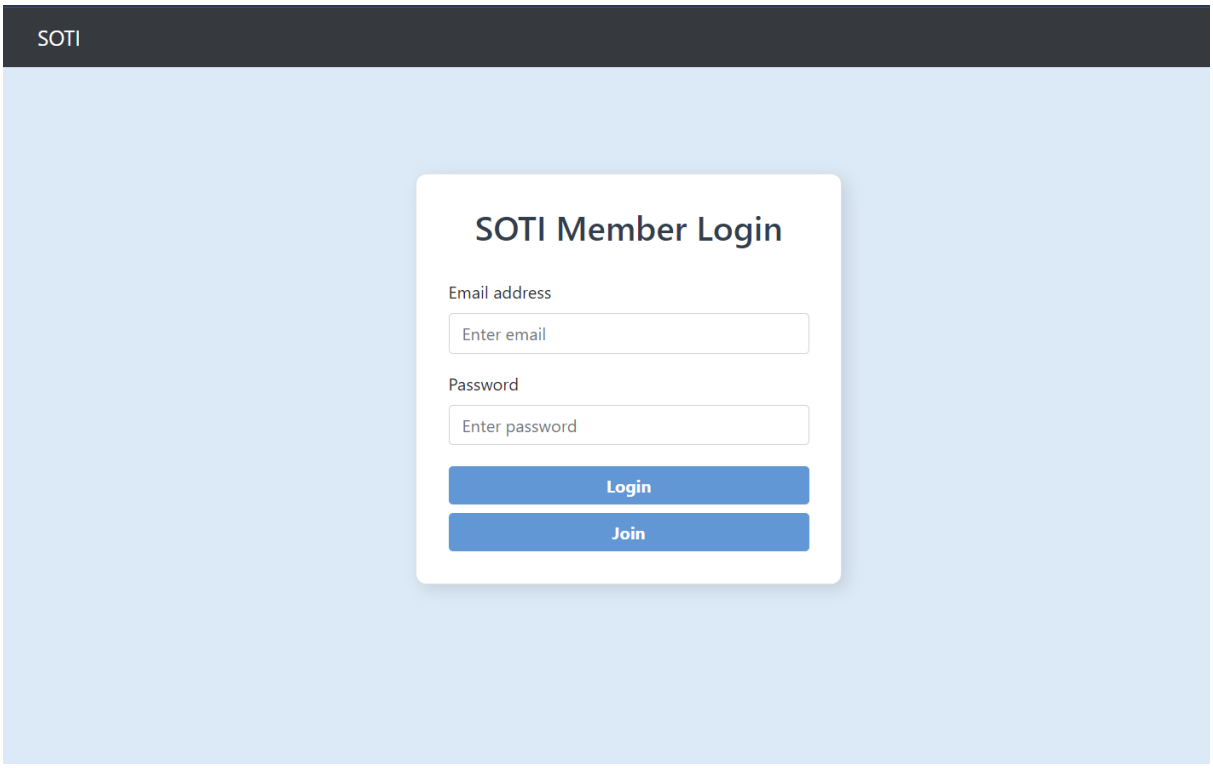
restricted_globbing

문제 분류	web
poc 작성자	김재환
문제 개발자	김재환

1. 문제

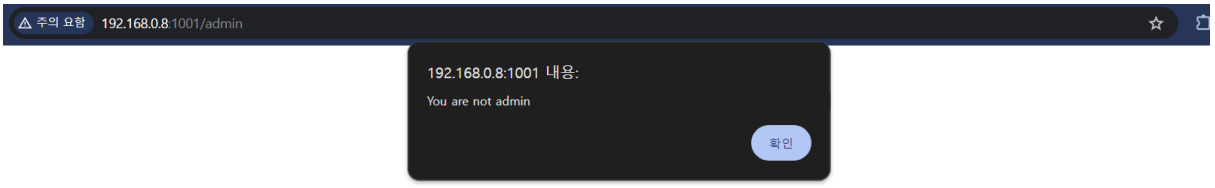
입장과 동시에 회원가입을 해야합니다.



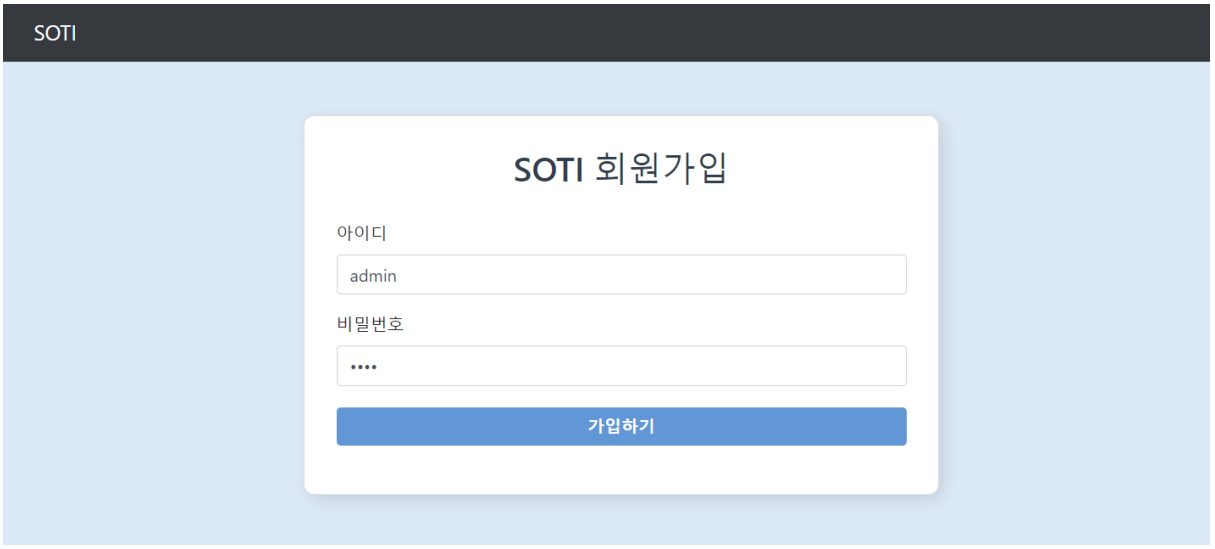
회원가입을 진행하게되면, 기본적으로 `role`이 Guest로 생성이 됩니다.

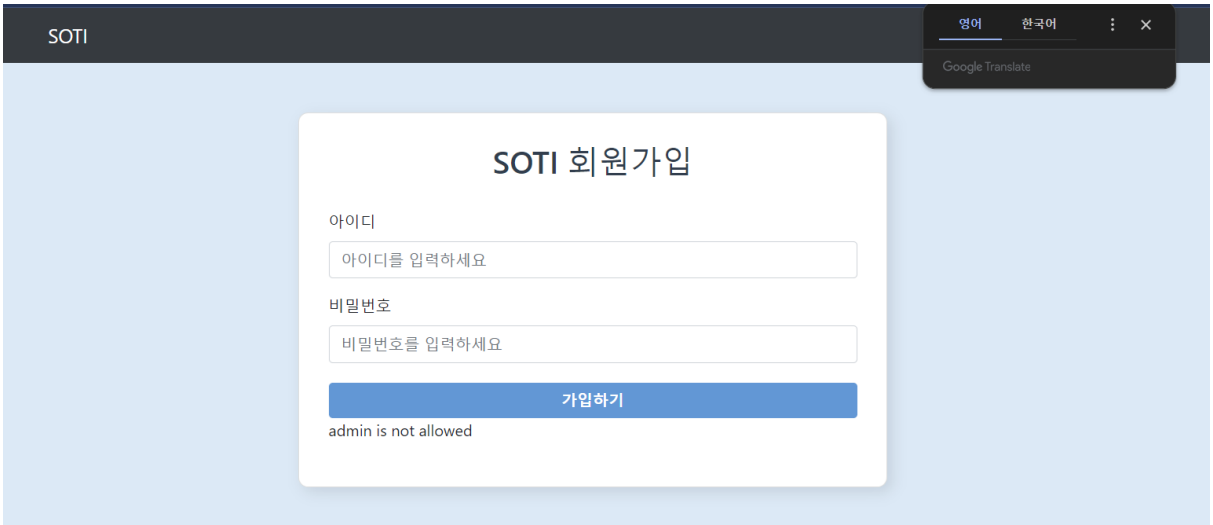
- 내용을 보면 role을 업그레이드하라고 되어있습니다.

그냥 `/admin`으로 접근하게 되면 `role`이 `guest`이기 때문에 거부 당합니다.



따라서 `admin` 으로 회원가입을 해야하는데, `admin을 검사하여 admin이 있을 경우 차단합니다.`





2. 풀이

2.1 `python regex bypass`

python 코드를 살펴보면, 회원가입 시 `regex`로 `admin`을 잡아 내는 게 있습니다.

```
@app.route("/join", methods=["GET", "POST"])
def signup():
    if request.method == "GET":
        return render_template("join.html")

    if request.method == "POST":
        username = request.form.get("id")
        password = request.form.get("pw")

        if username == "" or password == "":
            return render_template("join.html", msg="Enter us

        # regex
        m = search(r".*", username)

        if username or m:
            if m.group().strip().find("admin") == 0:
                return render_template("join.html", msg="admi
            else:
```

```
                username = username.strip()
                sha256_password = sha256((password).encode())
                join(username, sha256_password)
                return redirect("/login")
```

- 하지만 코드 리뷰를 잘 해보면 `admin`을 검사한 후, `strip()`을 통해 양쪽의 공백을 삭제합니다.
- `'          admin          ' -> admin`

여기서 활용할 수 있는 문제인데, `python의 re라이브러리의 search는 개행(엔터) 전까지만 검사합니다` 따라서 `\nadmin`을 입력할 경우 우회할 수 있습니다.

```
>>> from re import search
>>> m = search(r".*", "\nadmin")
>>> m.group().strip()
''
>>> m.group().strip().find("admin")
-1
>>> m = search(r".*", "admin")
>>> m.group().strip().find("admin")
0
>>>
```

- 우회되어 저장되는 `\nadmin`은 추후에 코드에 따라서 `strip()`되어 저장되기 때문에 최종적으로 아이디는 `\nadmin` 이 아닌 `admin`으로 회원가입이 됩니다.

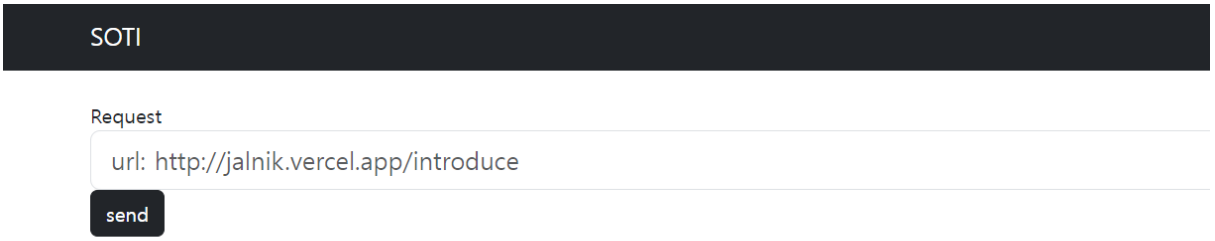
따라서 개행을 뜻하는 `%0A`를 아이디 앞에 추가해 줌으로써, admin으로 접근할 수 있게 됩니다.

```
Accept-Encoding: gzip, deflate
Accept-Language: ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: session=eyJyb2xlIjoiZ3Vlc3QiLCJ1c2VybmFtZSI6ImFzZGYifQ.ZhE1IQ.RwlbhqbLKrO6Qty46l3jbD16AAw

id=%0Aadmin&pw=1234
```

2.2 Curl URL Globbing

admin으로 접근하면, url로 뭔가를 가져올 수 있는 기능이 존재합니다.



해당 기능을 살펴 보기 위해 코드를 보면, `subprocess`로 os에 있는 `curl`을 사용하여 url로 요청을 보냅니다.

```python
if request.method == "POST":
    url = request.form["url"].strip()

    if (url[0:4] != "http") or (url[7:34] != "jalnik.versel.app/introduce"):
        return render_template("admin.html", msg="Not allowed URL")

    if (".." in url) or ("%" in url):
        return render_template("admin.html", msg="Not allowed path traversal")

    if url.endswith("secret"):
        return render_template("admin.html", msg="Not allowed string or character")
    try:
        response = subprocess.run(
            ["curl", f"{url}", "-m", "1"], capture_output=True, text=True, timeout=2
        )
        return render_template("admin.html", msg=response.stdout)

    except subprocess.TimeoutExpired:
        return render_template("admin.html", msg="Timeout !!!")
```

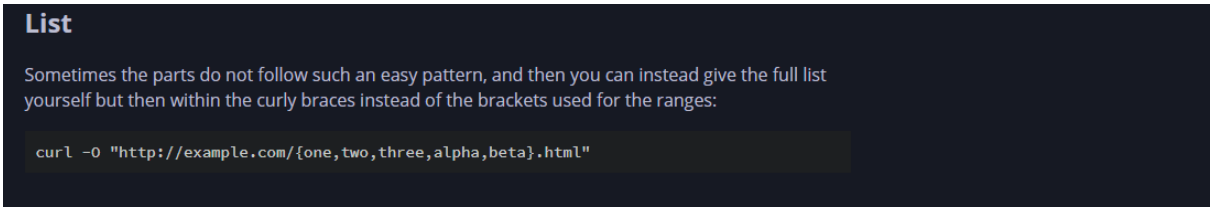
좀 더 내려가서 코드를 보면, secret이라는 url이 존재하며, 최종적으로 여기를 공략해야 함을 알 수 있습니다. 하지만 해당 url은 `localhost` 즉, 서버 내부에서만 접근할 수 있는 규칙이 존재합니다.

```python
@app.route("/secret", methods=["GET"])
def flag():
    ip_address = request.remote_addr
    if ip_address == "127.0.0.1":
        return FLAG
    else:
        return "Only local access allowed", 403
```

`curl 127.0.0.1/secret`을 입력하주면 쉽게 풀 수 있는 SSRF문제이지만, 조건식에 반드시 `http://jalnik.versel.app/introduce`로만 접근할 수 있도록 해놓았습니다.

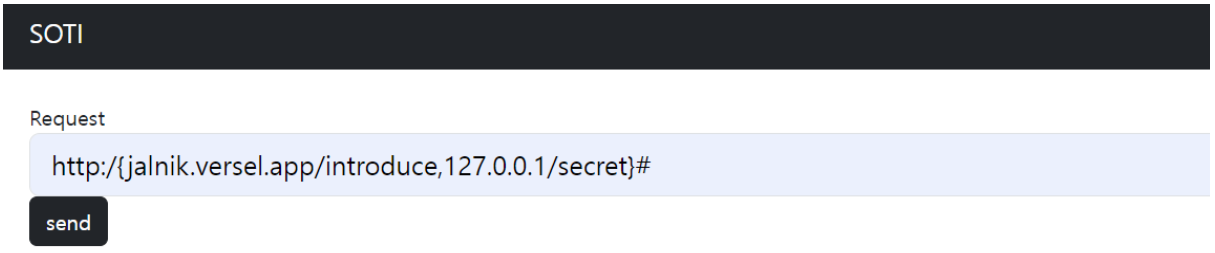
```python
if (url[0:4] != "http") or (url[7:34] != "jalnik.versel.app/introduce"):
    return render_template("admin.html", msg="Not allowed URL")
```

하지만 우리에게는 `curl urlglobbing이 존재합니다.`



- curl에서는 `{}`로 감싸져 있는 부분에 대하여 모두 시도합니다.
- 예를 들어 `curl {127.0.0.1, abcd.com }`이면 `127.0.0.1`과 `abcd.com` 둘다 접근한다는 뜻입니다.

따라서 최종 페이로드를 작성하면 다음과 같습니다.



- 참고로 http요청을 진행할 때는 URL Parser에 따라서 구문을 달리해도 되는데, curl의 경우 `http://` 와 `http:/`를 동일하게 처리합니다.

플래그 추출 성공