Command-injection-1

> web

🖥️ Command Injection 취약점

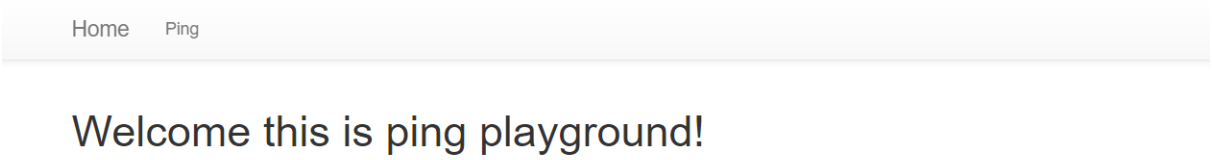
취약한 애플리케이션을 실행 중인 서버에서 임의의 **운영체제 명령**을 실행할 수 있는 취약점을 의미한다. 이 취약점이 존재할 경우 애플리케이션을 구동하고 있는 시스템 계정의 **쉘 권한**을 획득한 것과 같기 때문에 위험도가 높은 취약점 중 하나이다.

📜 문제

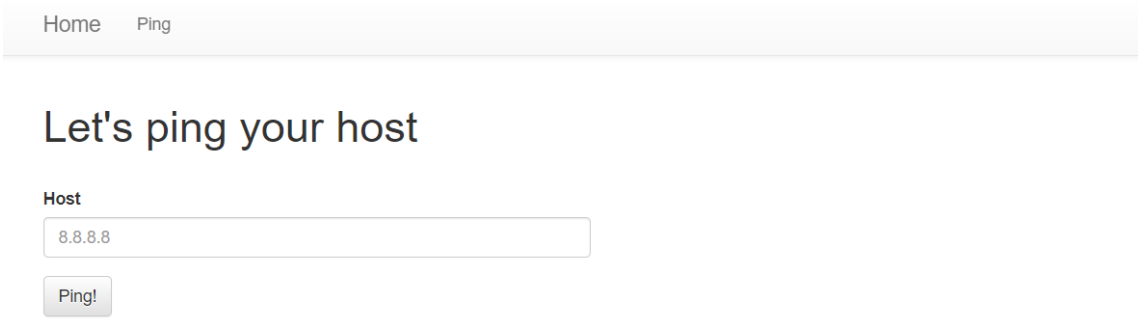
문제 설명

특정 Host에 ping 패킷을 보내는 서비스입니다.
Command Injection을 통해 플래그를 획득하세요. 플래그는 `flag.py` 에 있습니다.

📜 풀이



첫 서버화면이다.



ping 화면이다. `Host`에게 **ping**을 보내 `flag.py`에 있는 **플래그**를 획득해야할 것 같다.

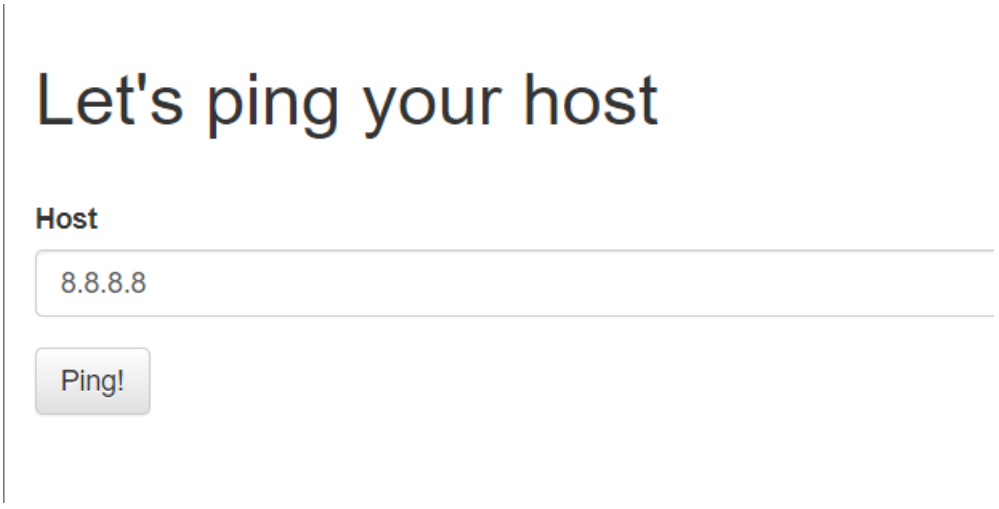
```
#!/usr/bin/env python3
import subprocess

from flask import Flask, request, render_template, redirect

from flag import FLAG

APP = Flask(__name__)


@APP.route('/')
def index():
    return render_template('index.html')


@APP.route('/ping', methods=['GET', 'POST'])
def ping():
    if request.method == 'POST':
        host = request.form.get('host')
        cmd = f'ping -c 3 "{host}"'
        try:
            output = subprocess.check_output(['/bin/sh', '-c', cmd], timeout=
            return render_template('ping_result.html', data=output.decode('ut
        except subprocess.TimeoutExpired:
            return render_template('ping_result.html', data='Timeout !')
        except subprocess.CalledProcessError:
            return render_template('ping_result.html', data=f'an error occurr

    return render_template('ping.html')


if __name__ == '__main__':
    APP.run(host='0.0.0.0', port=8000)
```

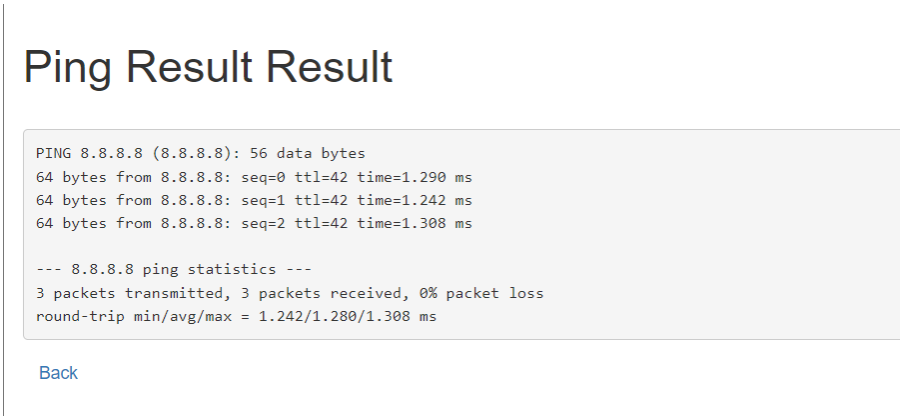
소스코드를 살펴보면 요청할 때 `host`라는 변수 데이터를 가져온다. 가져온 `host` 값을 사용하여 `ping` 명령어를 실행한다. `subprocess.check_output`함수를 사용하여 `/bin/sh` 에서 cmd 명령어를 실행하고, 결과 출력을 변수 `output`에 저장한다.

- 명령어 실행 중 예외가 발생하지 않으면, 결과 출력을 **UTR-8**로 디코딩한 후 `ping_result.html`템플릿과 함께 렌더링(서버로부터 html 파일을 받아 브라우저에 뿌려주는 과정)하며 반환합니다.
- 만약 명령어 실행 시간이 5초를 초과하면 `subprocess.TimeoutExpired` 예외가 발생하며, `Timeout !` 메시지와 함께 `ping_result.html` 템플릿을 렌더링하여 반환합니다
- 명령어 실행 중 오류가 발생하면 `subprocess.CalledProcessError` 예외가 발생하며, 해당 오류 메시지와 함께 `ping_result.html` 템플릿을 렌더링하여 반환합니다.

subprocess.check_output함수

서브 프로세스를 실행하고 출력 문자열을 파이썬 로직에서 변수에 담아 사용하고 싶은 경우 사용하는 함수이다. 비정상 종료되면 **CalledProcessError** 예외를 발생시킨다.





`8.8.8.8 (구글 DNS)`주소에 **ping**을 보내본 결과이다.

`ping -c 3 “8.8.8.8”` 명령어가 올바르게 실행되었다.

`cmd` 변수는 사용자로부터 입력 받은 `host` 값을 포함하고 있다. 하지만 **사용자의 입력을 그대로 실행하기 때문에, 사용자가 악의적인 공격 코드를 주입**할 수 있다. 이를 **Command Injection** 공격이라고 한다.

Command Injection을 발생시키는 방법은 주로 “메타 문자”를 통해 발생한다.

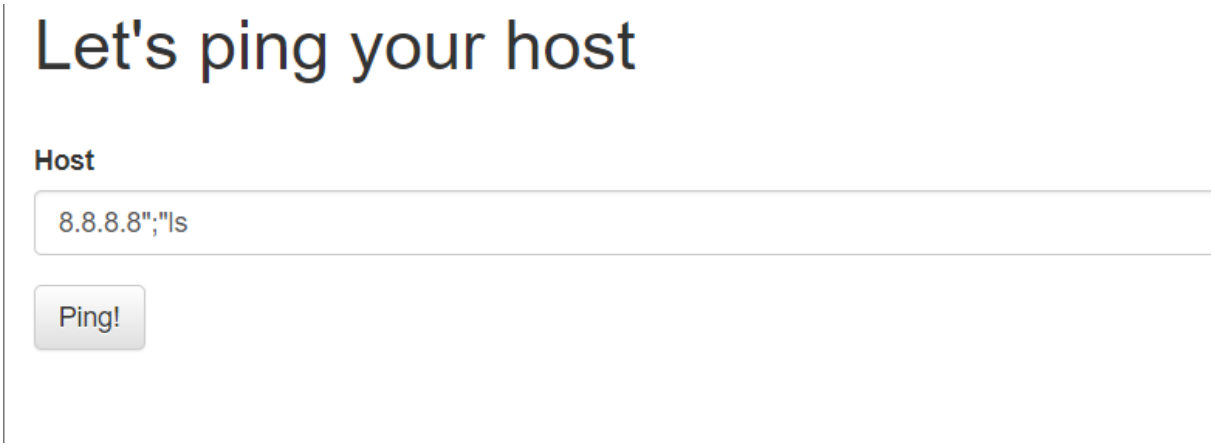
메타문자

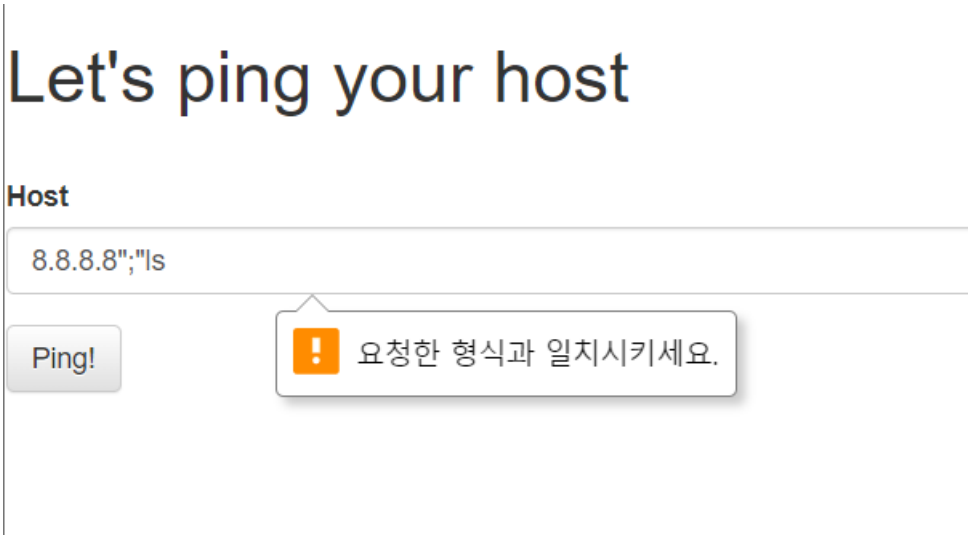
쉘에는 한 줄에 여러 명령어를 실행하는 등의 쉘 사용자 편의성을 위해 제공하는 특수 문자들이 존재한다. 시스템 명령이 수행된다면 특수문자를 활용해 `Command Injection` 공격이 가능하다.

메타문자	설명
``	**명령어 치환** `` 안에 들어있는 명령어를 실행한 결과로 치환
$()	**명령어 치환** $( ) 안에 들어있는 명령어를 실행한 결과로 치환, 중복 사용 가능
&&	**명령어 연속 실행** 한 줄에 여러 명령어를 사용하고 싶을 때 사용. 앞 명령어에서 에러가 발생하지 않아야 뒷 명령어 실행
||	**명령어 연속 실행** 한 줄에 여러 명령어를 사용하고 싶을 때 사용. 앞 명령어에서 에러가 발생해야 뒷 명령어 실행
;	**명령어 구분자** 한 줄에 여러 명령어를 사용하고 싶을 때 사용. ; 은 단순히 명령어 구분을 위해 사용하며, 앞 명령어의 에러 유무와 관계 없이 뒷 명령어 실행
|	**파이프** 앞 명령어 결과가 뒷 명령어 입력으로 들어간다.
그 외	., >, >>, &>, >&, <, {}, ?, *, ~

예를 들어, `host` 입력 값에 `8.8.8.8”;”ls` 를 입력 받을 경우 `cmd` 변수는 `ping -c 3 “8.8.8.8”;”ls”` 문자열을 가지게 된다. `/bin/sh` 로 실행시킨 결과는 의도된 핑 뿐만 아니라 `ls` 라는 명령의 결과를 추가적으로 확인할 수 있다.

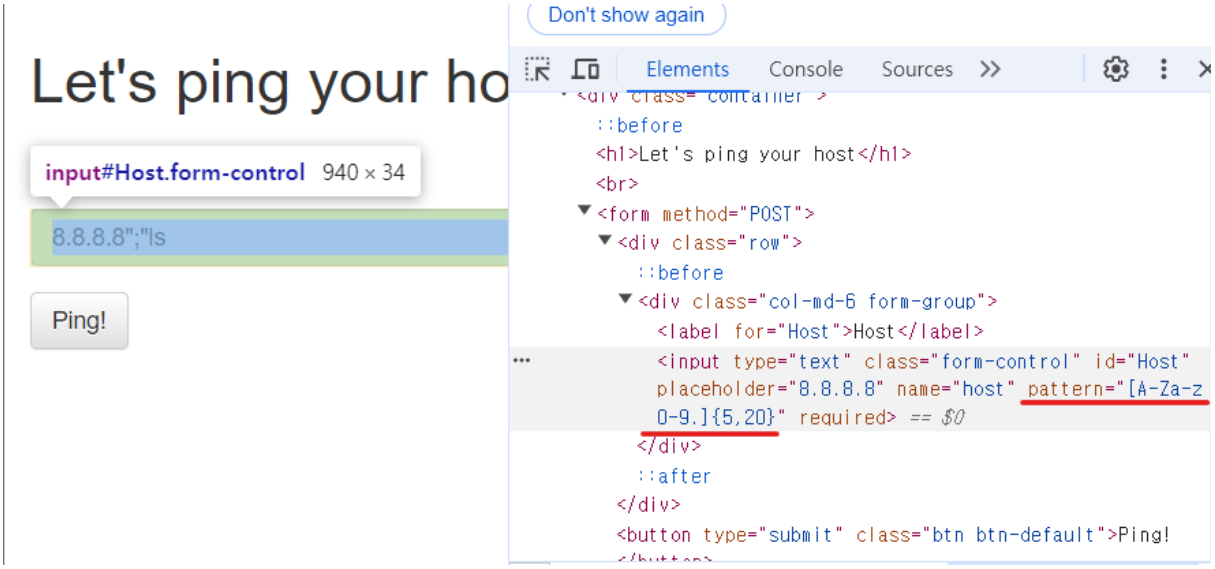
직접확인해보자.



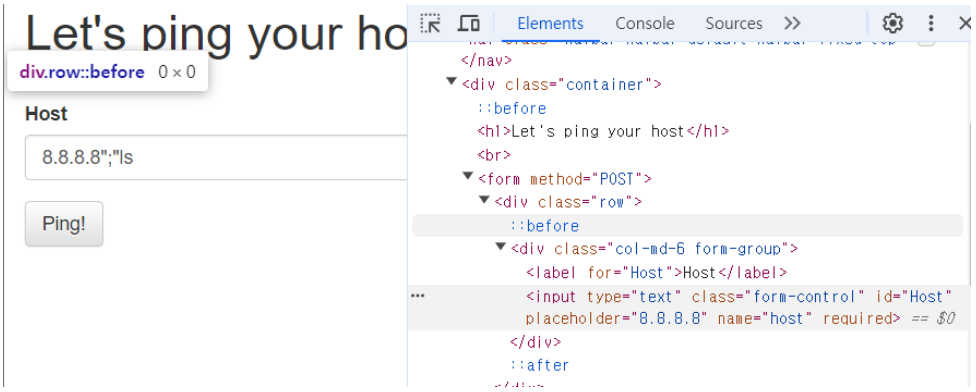


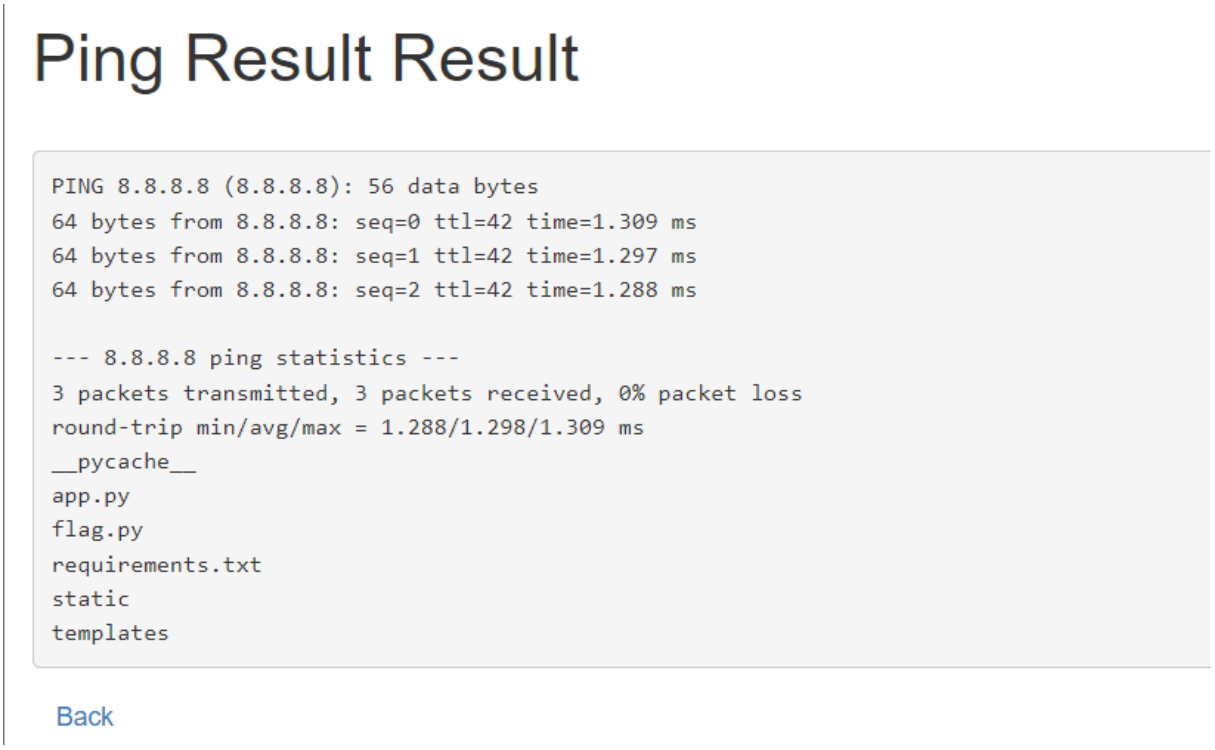
[요청한 형식과 일치시키세요.] 라는 경고 문구가 뜬다.

문구를 필터링하는 조건이 있는가 보다. (어렵다🥹)



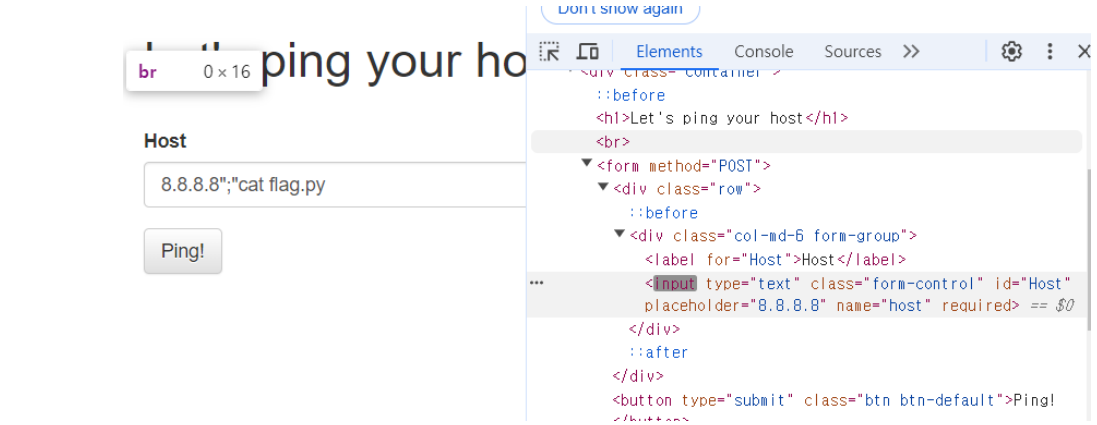
개발자 도구[F12]를 통해 html 소스코드를 확인해보니 `input` 태그의 `pattern` 속성이 설정되어 있음을 확인할 수 있었다. `pattern` 속성이란 폼 제출 시 `<input>` 요소의 값을 검사할 때 사용될 `정규 표현식(regular expression)`을 명시하며, 허용된 정규 표현식은 영어, 숫자, `"."`, 글자수 5~20이다.

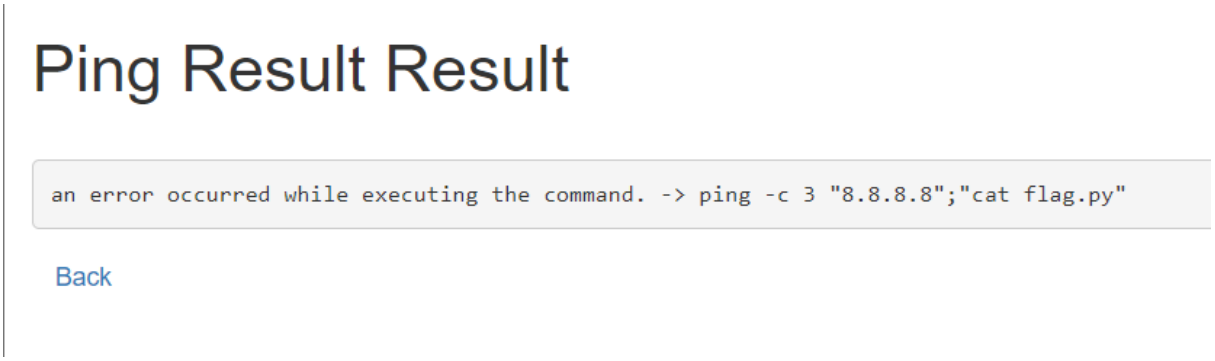




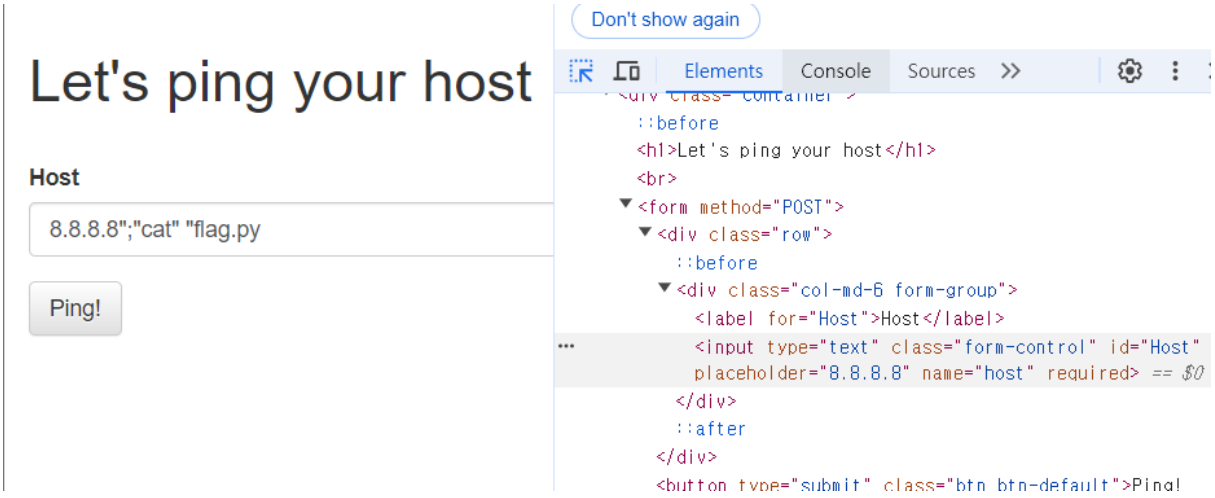
`pattern` 속성을 없애고 다시 한번 **ping**을 하였더니 **8.8.8.8 주소의 ping 값**과 **ls명령어**의 값이 함께 나온 것을 확인할 수 있었다.

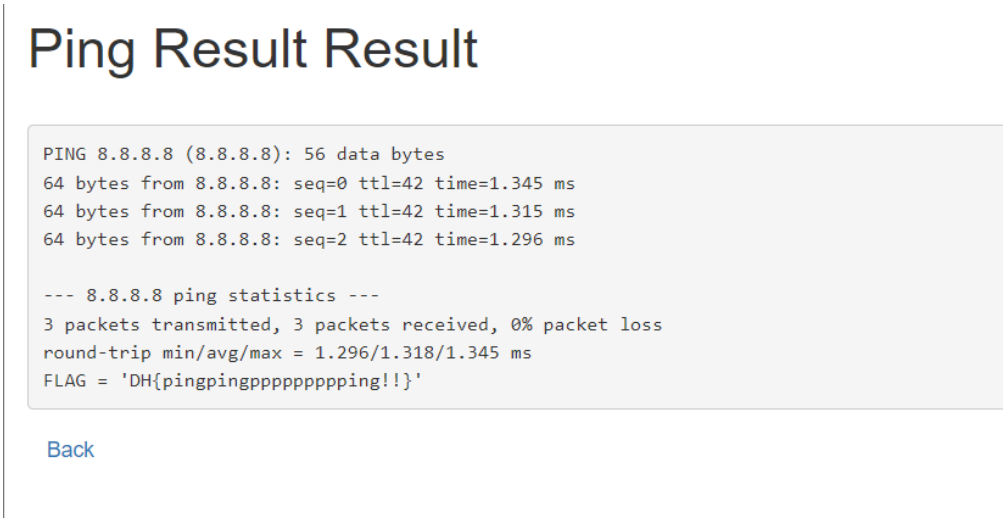
flag.py 에서 플래그 값을 찾을 수 있기 때문에 **cat 명령어**를 사용하여 flag.py의 내용을 볼 것이다.





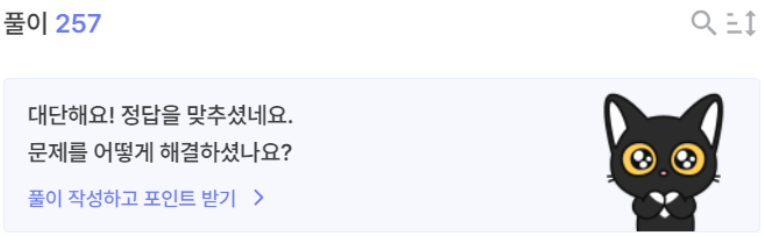
안된다. ls 명령어는 되는데 왜 cat flag.py는 안될까.





“cat flag.py는 안되고 **“cat” “flag.py”**는 된다. 왜일까!!!!!!!!

어쨌든 플래그 값을 획득했다.



어렵다 ㅎㅎ,, 아직 혼자서는 못하겠다.

Flask

위의 소스코드에서

```
from flask import Flask, request, render_template, redirect
APP = Flask(__name__)
```

부분을 보면 flask 서버로 짜여진 것을 확인할 수 있다.

Flask는 **파이썬으로 작성된 마이크로 웹 프레임워크(Micro Web Framework)** 중 하나이다. 웹 애플리케이션을 구축하는 데 필요한 핵심 기능을 제공하는데 중점을 둔 단순하고 미니멀한 디자인으로 유명하다. 간단한 웹 사이트나 혹은 간단한 API 서버를 만드는 데에 특화된 Python Web Framework 이다.

Web Framwork (웹 프레임워크)

웹 프레임워크는 **동적인 웹 페이지**나 **웹 어플리케이션**을 개발할 때 유용하게 사용되는 일종의 **‘틀(fram)’**이다.

일반적으로 데이터베이스(DB) 연동, 템플릿 형태의 표준, 세션 관리, 코드 재사용 등의 기능을 포함한다.

---

https://maker5587.tistory.com/61

https://hobbylists.tistory.com/entry/드림핵DreamHack-커맨드-인젝션-1-Command-Injection-1풀이