Understanding Privilege Escalation

By Marcin Teodorczyk

Local privilege escalation happens when one user acquires the system rights of another user. Network intruders have many techniques for increasing privileges once they have gained a foothold on a system. The initial intrusion could start from anywhere. A guest account? A local user who has carelessly written a username and password on a Post It note? Everyday users typically operate at a relatively low privilege level – specifically to prevent someone who obtains their credentials from gaining control of the system. Once inside, the intruder employs privilege escalation techniques to increase the level of control over the system. In this article, I'll describe some techniques malicious users employ to escalate their privileges on a Linux system.

I'll start with a low-privilege user account with SSH access and try to escalate the privileges. Of course, this article can only cover a small fraction of the privilege escalation techniques in use today, but it should give some indication of how an intruder thinks and acts in a typical attack session.

For purposes of this exercise, I'll launch this test attack on a system running Metasploitable. Metasploitable is a purposely old and vulnerable Ubuntu 8.04 system maintained by the Metasploit project.

Information Gathering

Once the intruder logs in with a low-end user account, the first step is to get some information about the system and rights.

First, get the distribution name:

```
$ cat /etc/issue
Ubuntu 8.04 \n \l
```

An intruder typically executes some basic commands to get to know the system. Of particular interest are the \etc directory and the package manager.

One really obvious (but rarely successful) trick is to look for writable SUID/GUID files. SUID (Set User ID) files execute with the privileges of file owner, rather than the user, and a writable SUID file would essentially allow the user to execute any commands the owner of the file could execute simply by modifying the file.

```
find / -o -group `id -g` -perm \
 -g=w -perm -u=s -o -perm -o=w\
 -perm -u=s -o -perm -o=w \
 -perm -g=s -ls
```

SUID files are naturally used to escalate privileges when needed and should not be writable. If they are, a malicious user can simply change their contents before executing them, thus effectively escalating his privileges system wide.

Other questions the intruder asks are:

- Are there any unmounted filesystems?
- What development tools are available? Is there access to a compiler? Can I execute compiled files?
- Is there access to a package manager to download and configure new applications?

Also, it is worth getting information about scheduled cron jobs, checking out network configuration, and searching for files writable outside the home directory.

Most of this information gathering can be automated with at least a simple shell script, though there is always some part that has to be done manually. Listing 1 shows an example of a shell script an intruder might use to gather basic information about the system. The results are written to the standard output, which is typically redirected to a text file. A single output file lets the intruder concentrate on analyzing the gathered information, instead of typing a succession of interactive *find* commands (and *man* commands to look up *find* parameters).

In this case, since the initial user account has a low privilege level, most of *find* commands will generate Permission denied warnings. The redirection *2>/dev/null* attached to each of the commands in Listing 1 removes those warnings from the output, making the report much easier to read and analyze.

Listing 1: Example of basic information gathering script

```
#!/bin/sh
echo Distribution and kernel version
cat /etc/issue
uname -a

echo Mounted filesystems
mount -l

echo Network configuration
ifconfig -a
cat /etc/hosts
arp

echo Development tools availability
which gcc
which g++
which python

echo Installed packages (Ubuntu)
```

```
dpkg -l

echo Services
netstat -tulnpe

echo Processes
ps -aux

echo Scheduled jobs
find /etc/cron* -ls 2>/dev/null
find /var/spool/cron* -ls 2>/dev/null

echo Readable files in /etc
find /etc -user `id -u` -perm -u=r \
 -o -group `id -g` -perm -g=r \
 -o -perm -o=r \
 -ls 2>/dev/null

echo SUID and GUID writable files
find / -o -group `id -g` -perm -g=w -perm -u=s \
 -o -perm -o=w -perm -u=s \
 -o -perm -o=w -perm -g=s \
 -ls 2>/dev/null

echo SUID and GUID files
find / -type f -perm -u=s -o -type f -perm -g=s \
 -ls 2>/dev/null

echo Writable files outside HOME
mount -l find / -path "$HOME" -prune -o -path "/proc" -prune -o \( ! -type l \) \( -user `id -u` -perm -u=w  -o -group `id -g` -perm -g=w  -o -perm -o=w
\) -ls 2>/dev/null
```

Running the script from Listing 1 on Metasploitable produces quite a large amount of output.

In the case of Metasploitable, the script did not turn up any unmounted filesystems, as the *cfdisk* and *mount -l* commands don't show anything suspicious. If any unmounted filesystems existed, the next step would be to try to mount them, look around, and execute all the information-gathering steps again.

The script outputs some details about the network configuration, but nothing looks interesting – no hosts defined in */etc/hosts* , one network interface, and an almost-empty arp table.

Several development tools are available, including *gcc* and *g++* . I can run compiled programs from the home directory, which means I can compile exploits if we need to.

The account has access to a package manager. I can't install or remove software, but I can list the software installed with its detailed version information. This information will be very useful later.

Also, I can list all system processes with *ps aux* and listen on ports with *netstat -tulnp* , although I can't see for sure which process listens on what port (root rights required). Nonetheless I can infer that, among others, Apache, Distcc, and Tomcat running.

There are also a few scheduled cron jobs, including PHP- and Tomcat-related jobs. I can't use them directly, but they give me a clue about what's running on the system. Also, if I can read their contents, I can try to control their input (if they have any).

And finally, great news – I can read virtually the whole content of */etc* and */var* directories.

Access to the */etc* and */var* directories gives me a great starting position for searching configuration flaws, though the administrator could easily have prevented access to this configuration information. Different use of traditional Unix file rights, enhanced by grsecurity, would stop me from reading configuration files, services, processes, network configuration, and information on installed software without making the life of the typical user any harder.

Configuration Flaws

I didn't find any writable SUID/GUID files, which is not surprising, but I found 884 SUID/GUID files without permission to write. This is an unusually high number, which increases the chances that one or more will be vulnerable to privilege escalation. Most of these files are GUID files owned by user *msfadmin* and group *www-data* . The remaining 11 GUID files are owned by *root* and various groups. These files do not seem useful at first glance, so I won't analyze them deeper now, but I will remember to get back to them if I'm not successful with anything else.

Searching for writable files and directories outside home directory might provide a means for escalating privileges. Metasploitable revealed 1070 such entries. Among these writable files and directories, two directories are particularly interesting.

First is the directory */var/lib/php5* , owned by user *root* and group *root* , with write/execute permissions but without reading permission for my user directory. I could try to use this directory for overwriting some PHP lib with my own version, but that seems like a lot of work, so I pass on that option for now and look for lower-hanging fruit. I can always get back to this if I don't find anything better.

Second are 473 files and directories owned by the user *msfadmin* and group *msfadmin* , placed in */var/www/tkwiki/data* and */var/www/twiki/pub/Main* . Most of these files are executable, in spite of the fact, that they are mainly image and text files. At first glance, these files don't seem to be of much use for us, but wait, doesn't this look like it might be website data? Invoking a simple *grep -r '/var/www/twiki' /etc/apache2* and looking up the collected earlier process list and services list reveals that I have happened onto the wiki called Twiki, which should be available through HTTP.

Indeed, I can even open a wiki website using the URL: *http://metasploitable/twiki/* . This means I can easily run code with the rights of the web server, that is, as the previous *ps* showed, with *www-data* rights. A simple proof of concept is a PHP script with the following contents:

```
echo 'User: ' . exec('whoami');
```

I place the script in */var/www/twiki/pub/Main/* . After navigating with a web browser to *http://metasploitable/twiki/pub/Main/whoami.php* , I get a web page saying *User: www-data* . Unfortunately (if you're the attacker) or fortunately (if you're the administrator), the web server doesn't run with root privileges. Nonetheless, I can escalated my rights to *www-data* , and I can use this account for further escalation later.

1 2 3 Next »

Related content

Share / Save

Pen Test Tips

The powerful Metasploit framework helps you see your network as an intruder would see it. You might discover it is all too easy to get past your own defenses.

more »