Insider Tips: Locating and disabling network services

Stop Chatting

Unix systems have a huge range of functions and services. Sometimes it is hard for some administrators to know exactly what entities are hiding out on their machines. This article helps you locate and specifically disable unwanted communication services. BY MARC ANDRÉ SELIG

Once upon a time there was a manufacturer of a widespread windowing system that was notorious for its promiscuous default settings. It wanted every user to be able to use a computer out of the box, no matter what. Because no manufacturer can exactly anticipate a user's needs, the manufacturer in our story simply enabled every available network service, just in case. That was a good thing in one sense. Everything the users tried worked straight away – apart from a few broken services and functions, but that is another story altogether.

There is just one thing wrong with that manufacturer's approach; it allows malevolent hackers easy access to other people's computers. Of course, it is convenient to allow networked access to any service by default, but it does play into the hands of the evildoers. If they find an exploitable bug, then the regular user is in for trouble.

Most programs that provide networking services are not required by most machines. Thus, it makes sense to disable them for security reasons. The manufacturer from our story uses security updates to (hopefully) take care of that. On Unix-type operating systems, this job is the responsibility of the administrator, as unfortunately not all distributions disable network services by default.

Displaying Processes

The first thing you should do is to check what services are running on your machine. *ps* (process status) is the easiest way to do this. The detailed version *ps ax* (BSD style) or *ps -ef* (Posix style) creates a neat list of all the processes running on the local Unix machine (see Figure 1).

In an ideal world you would recognize all the processes you see, and know that they are all doing something really useful on your machine. This said, even a more or less idle computer with a single user will have about a hundred processes running. It takes a really experienced admin to tell what they are doing. So let's take a closer look.

Finding Open Doors

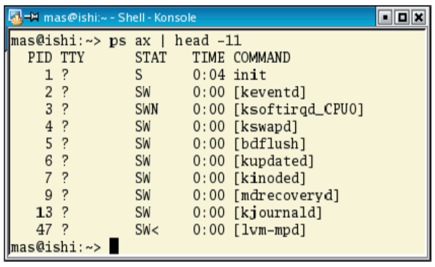
The biggest danger arises from services that provide, or open up, network connection. Unix wouldn't be Unix if it didn't have a command to display these services: *netstat -a* shows the active or listening sockets, that is, connections that allow active programs to talk to the outside world.

On Linux, you can specify the type of socket you are interested in by typing *netstat -tuan* (see Figure 2). The additional options restrict the output to TCP and UDP sockets, that is they ignore the Unix domain sockets, which are only used for exchanging data locally between programs running on the same computer. The *-n* option prevents name resolution and thus saves a lot of time. However, you only get to see the IP addresses of the communication partners, and not their DNS names.

For each socket, the output indicates the protocol (TCP or UDP), both endpoints, and the status of the connection. The status is not shown for connectionless UDP sockets. An active TCP socket (that is a socket in the *ESTABLISHED* state) represents a connection between two defined ports, at two IP addresses. States such as *TIME_WAIT* or *CLOSING* indicate sockets which are in the process of closing, or have already done so. UDP sockets are connectionless, and thus stateless. A UDP socket simply accepts any packets thrown at it.

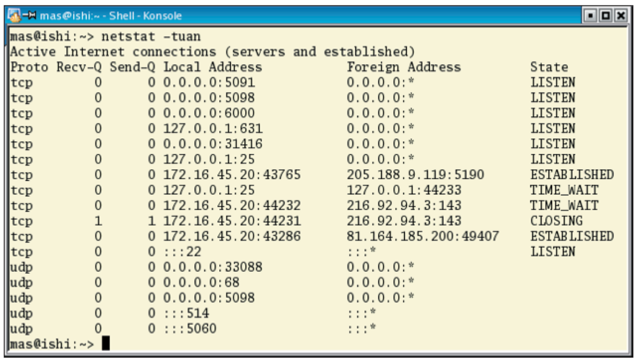
Caution, Server

Sockets in a *LISTEN* state, and any UDP



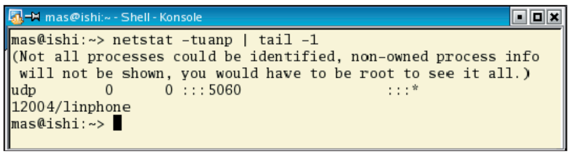
```
mas@ishi:~> ps ax | head -11
  PID TTY      STAT   TIME COMMAND
    1 ?        S      0:04 init
    2 ?        SW     0:00 [keventd]
    3 ?        SWN    0:00 [ksoftirqd_CPU0]
    4 ?        SW     0:00 [kswapd]
    5 ?        SW     0:00 [bdflush]
    6 ?        SW     0:00 [kupdated]
    7 ?        SW     0:00 [kinoded]
    9 ?        SW     0:00 [mdrecoveryd]
   13 ?        SW     0:00 [kjournald]
   47 ?        SW<    0:00 [lvm-mpd]
mas@ishi:~>
```

Figure 1: *ps* shows the running processes. The *head -11* command restricts the output to the first 11 lines.



```
mas@ishi:~> netstat -tuan
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:5091            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:5098            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:6000            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:31416           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 172.16.45.20:43765      205.188.9.119:5190      ESTABLISHED
tcp        0      0 127.0.0.1:25            127.0.0.1:44233         TIME_WAIT
tcp        0      0 172.16.45.20:44232      216.92.94.3:143         TIME_WAIT
tcp        1      1 172.16.45.20:44231      216.92.94.3:143         CLOSING
tcp        0      0 172.16.45.20:43286      81.164.185.200:49407    ESTABLISHED
tcp        0      0 :::22                   :::*                    LISTEN
udp        0      0 0.0.0.0:33088           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:5098            0.0.0.0:*
udp        0      0 :::514                  :::*
udp        0      0 :::5060                 :::*
mas@ishi:~>
```

Figure 2: The *netstat -tuan* command displays all the TCP and UDP sockets used by the local machine to exchange data with the outside world.



```
mas@ishi:~> netstat -tuanp | tail -1
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
udp        0      0 :::5060                 :::*
12004/linphone
mas@ishi:~> █
```

Figure 3: In contrast to the command in Figure 2, *netstat -tuanp* additionally shows you the process assigned to each socket. In our example, linphone is listening on UDP port 5060, and its process ID is 12004.

sockets, should set the alarm bells ringing. These are servers that will accept data from external systems. It definitely pays to investigate them closely, and to check the local IP address and port number.

The port number of a server often tells you a lot about the program and the functionality listening on the socket. For example, a TCP socket with a port number of 25 will typically be assigned to a mail server, and a UDP socket at port 68 will probably be a DHCP client. You can check the */etc/services* file for port assignments. If you want to know which TCP service is listening on port 6000, just type *grep 6000/tcp /etc/services* to display the details on your terminal:

```
x11   6000/tcp   # X Window System
quake  26000/tcp  # quake
```

The IP address of a listening socket tells us if the socket really does accept data from the whole Internet, which is what the ICQ server at 0.0.0.0:5091 (IP address 0.0.0.0, port number 5091) in Figure 2 does. In contrast to this, the mail server socket is listening on 127.0.0.1:25. The 127.0.0.1 address is the loopback device. No need to worry here: this SMTP socket (Simple Mail Transfer Protocol) will only accept connections from loopback, and loopback is only available on the local machine.

The *-p* option sheds a lot of light on the subject, especially if you happen to be the root user (see Figure 3). This option tells netstat to locate the process keeping the socket open, and it might just point out a few services that look a bit fishy.

Tricks and Traps

Although a socket might be wide open, that does not automatically mean that it will be accessible from the Internet. Intermediate packet filters, especially ones with NAT functionality, can reliably prevent access attempts, no matter whether they are malevolent or benevolent.

More or less any Linux system will have an integrated packet filter. The filter has been known as netfilter [1] ever since the later kernel 2.3 versions, and the *iptables* command is used to configure it. Many distributions enable a basic netfilter configuration, but this does not necessarily mean that the rules make a lot of sense. You can type *iptables -L -n -v* to display the current ruleset. *iptables* output is somewhat cryptic, but we will be returning to that subject in another column.

Closing Doors

If *netstat* and *ps* reveal services that should not be running on your computer, you will probably want to disable them, being a conscientious admin type of person. If the service has its own daemon, you will typically need to disable an init script in */etc/rc*.d/* [2]. Some services are called on demand via a centralized control daemon. In this case, you need to modify the configuration in */etc/inetd.conf* or in the */etc/xinetd.d* directory [3].

If you think you have finished hardening your system, you should perform a few simple checks just to make sure. A port scanner like nmap [4] is the perfect tool to rattle the blinds and doors, checking for open ports. Finally, a word of warning. Do not simply disable any services you find without prior investigation. There are many basic daemons required to keep a Unix system running. If in doubt, set up a new system, and compare it with your machine to be on the safe side. ■

INFO

[1] Netfilter: *http://www.netfilter.org*

[2] Marc André Selig, "The Sys V Init Process": Linux Magazine Issue 44, July 2004, page 61

[3] Marc André Selig, "Finger Pointing": Linux Magazine, Issue 37, December 2003, *http://www.linux-magazine.com/issue/37/Admin_Workshop_Finger_Server.pdf*

[4] Nmap: *http://www.insecure.org/nmap/*