Protecting Linux systems from attacks: Part 1

BATTEN DOWN THE HATCHES!

MIRKO DÖLLE

When asked about potential attacks on the free operating system, many people simply reply "Linux is secure". But even with the standard installation of a distribution, you could be leaving loopholes the size of barn doors.

Many users employ the standard installation of the respective distribution in the implicit belief that all the necessary programs are installed and set up here. But some packets are neither necessary in normal operation nor are they anything like completely safe, and you ought to be aware of their risks and side effects before letting them loose on the Internet. These are the so-called **daemons**, utilities that either run constantly in the background or are started completely automatically when certain computer resources are accessed. After a standard installation these frolic about by the dozen on your home PC. Figure 1 shows the process list of a computer with SuSE 7.0.

In the last column you will find the file name of the respective daemon, in the first is the name of the user with whose rights this utility is running. In the case of httpd this will sometimes be root and sometimes wwwrun. A process running as root is potentially problematic, because it has all the rights of the administrator. In the process list shown (which you get with *ps auxww*) though, only programs currently running are listed. Daemons that are only started on demand from outside are not found here. These are started from **inetd**, the Internet Super-Server Daemon, using

```
grep -v "^#" /etc/inetd.conf
```

We risk a look into the configuration file of *inetd*, where comments and deactivated utilities – their lines always start with a hash – are not displayed. In Figure 2 you can see the enabled utilities of the SuSE standard installation.

In the first column you can find the name of the utility, in the last the associated command to start the daemon. The fifth column is especially important and extremely critical: Here is the name of the user with whose rights the respective

***Daemons**: Utility programs started either when the system is booted up or by a specific event and are not to be confused with the dark powers, or demons. They are usually recognisable by the 'd' on the end of their name, for example in httpd, the Apache Web server. Linux has dozens of daemons, which perform a wide variety of tasks. Thus cron makes it possible to execute constantly recurring actions according to a schedule, while the ftpd allows you to log onto a computer from outside and download any waiting files. System-related events are also taken care of by daemons, such as for example changing PC cards in notebooks by cardmgr.*

***inetd**: The Internet Super-Server Daemon is responsible for starting additional daemons when specific queries come from the Internet. Port number and desired protocol of the query are then compared with the entries in /etc/inetd.conf and the utility registered there is invoked.*

daemon is started. In principle all utilities, especially those with root privileges, are a potential security risk. There is no such thing as a totally secure computer, no matter how desirable this may be, and there is no prospect of there ever being one. Operating systems alone have now become much too complex (the current kernel 2.4.1 includes around 3.5 million lines, printed out on 55,000 pages of A4 or around 16 kilometres of continuous paper!), for an individual or a small group to be able to get in and sound it out for security risks. The fact is that now only especially suspect points are tested, which means in principle that after every change (patch) a complete retest ought to be necessary.

Evil Internet

A certain amount of paranoia is fully justified at this point. **Crackers** used to keep mostly to server-operators, to accommodate their attack tools, so-called **root-kits**, and to spy out data from there or simply mount a **DoS** attack to cripple other machines. Most administrators got wise after a few losses and have increased the security on their computers.

With the expansion of the flat rate, the crackers have discovered a new playing field: Many computers are on line permanently, or with only short breaks, so are almost always accessible. The sole drawback in comparison with true Internet servers is the changing addresses (IP) and lower bandwidth – but both are more than made up for by the considerably larger number of private computers. Nor is it a problem to find a computer ready to go: flat rate offers mean the user is bound to one provider and so are relatively easy to identify. The greatest advantage from the point of view of the attacker is that many computers are not protected at all, or have totally inadequate protection, users don't really know their way around the system and often use totally outdated software with long-since discovered security loopholes. Nor are the distributors completely blameless here, as obsolescent packets are found here, there and everywhere, for which attack scenarios (**Exploits**) have already been published and which anyone can find.

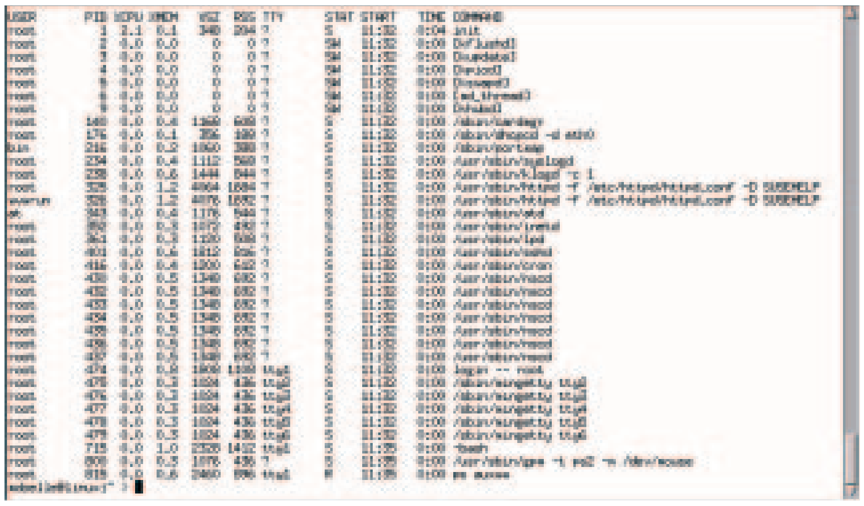
Unauthorised users

Especially dangerous are utilities which either provide information about their own system or enable an access. And all other daemons, in particular those with administrator privileges, represent a high security risk. An attacker always uses the same approach: Find out which operating system is running, look for insecure or wrongly installed utilities and finally take advantage of weak points. The worst-case scenario for the user in this event is certainly not the loss of personal data – but that it is now almost impossible for normal users, even to trace good root-kits on their own computer. The system thus turns into a ticking time bomb, which, together with other primed computers, can cripple entire networks at an unnoticed command. Most security loopholes in the daemons are due to programmer errors or ignored boundary conditions. Often the programmer expects the user to be too good-natured and fails to take account of the fact that instead of giving him one file

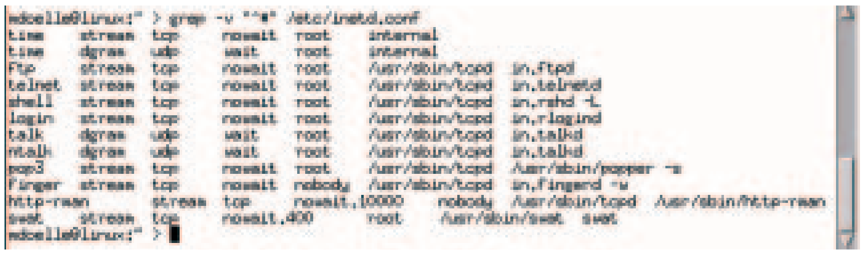
***Crackers**: Computer experts, who, unlike hackers, do not break into computers to highlight security loopholes, but to loot data, cripple computers or to install programs to collect passwords or sabotage other computers.*
***root-kit**: A collection of programs with which a cracker keeps control of other computers. Usually coupled with other sabotage programs.*
***DoS** and **DDoS**: Short for denial of service and distributed denial of service. Not to be confused with the operating system, DOS. In a DoS an attempt is made to prevent another system from performing its regular work – for example by bombarding an Internet search engine with thousands of meaningless queries. By now, the administrators responsible for the system under attack are reacting very quickly and simply ignoring all the data packets from a certain computer or group of computers. Attackers can get round this by using a Distributed DoS, in which not only individual computers attack, but hundreds of thousands, scattered all over the world. Often this uses the computers of Internet users who have no idea what is going on – so-called root-kits have been installed on their computers, and these attack one or more targets by remote control, without the owner knowing anything about it. Since the queries are now coming from all over the world, the administrators have to batten down all their hatches – and thus take their computer off the Net or shut it down.*
***Exploit**: Program which takes advantage of a security loophole in a utility or a computer and breaks in. They serve as a demonstration and proof that an attack in a certain way is possible and will succeed. Most exploits are generally available to the public.*

[top]
Figure 1: Colourful activities: Most daemons can be recognised by the 'd' on the end, but also include portmap, cardmgr and cron

[above]
Figure 2: Utilities which (almost) no-one needs: In */etc/inetd.conf* too, there are hidden daemons, which are started completely automatically



```
USER       PID %CPU %MEM   VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  2.1  0.1   340  204 ?        S    11:32   0:04 init
root         2  0.0  0.0     0    0 ?        SW   11:32   0:00 [kflushd]
root         3  0.0  0.0     0    0 ?        SW   11:32   0:00 [kupdate]
root         4  0.0  0.0     0    0 ?        SW   11:32   0:00 [kpiod]
root         5  0.0  0.0     0    0 ?        SW   11:32   0:00 [kswapd]
root         6  0.0  0.0     0    0 ?        SW   11:32   0:00 [md_thread]
root         9  0.0  0.0     0    0 ?        SW   11:32   0:00 [khubd]
root       140  0.0  0.4  1368  608 ?        S    11:32   0:00 /sbin/cardmgr
root       176  0.0  0.1   356  188 ?        S    11:32   0:00 /sbin/dhcpcd -d eth0
bin        216  0.0  0.2  1060  388 ?        S    11:32   0:00 /sbin/portmap
root       234  0.0  0.4  1112  568 ?        S    11:32   0:00 /usr/sbin/syslogd
root       238  0.0  0.6  1444  844 ?        S    11:32   0:00 /usr/sbin/klogd -c 1
root       335  0.0  1.2  4064 1684 ?        S    11:32   0:00 /usr/sbin/httpd -f /etc/httpd/httpd.conf -D SUSEHELP
wwwrun     338  0.0  1.2  4076 1692 ?        S    11:32   0:00 /usr/sbin/httpd -f /etc/httpd/httpd.conf -D SUSEHELP
at         343  0.0  0.4  1176  544 ?        S    11:32   0:00 /usr/sbin/atd
root       352  0.0  0.3  1072  432 ?        S    11:32   0:00 /usr/sbin/inetd
root       361  0.0  0.3  1120  508 ?        S    11:32   0:00 /usr/sbin/lpd
root       401  0.0  0.4  1212  556 ?        S    11:32   0:00 /usr/sbin/sshd
root       416  0.0  0.4  1200  620 ?        S    11:32   0:00 /usr/sbin/cron
root       430  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       432  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       433  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       434  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       435  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       436  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       438  0.0  0.5  1348  692 ?        S    11:32   0:00 /usr/sbin/nscd
root       474  0.0  0.3  1900 1108 tty1     S    11:32   0:00 login -- root
root       475  0.0  0.3  1024  436 tty2     S    11:32   0:00 /sbin/mingetty tty2
root       476  0.0  0.3  1024  436 tty3     S    11:32   0:00 /sbin/mingetty tty3
root       477  0.0  0.3  1024  436 tty4     S    11:32   0:00 /sbin/mingetty tty4
root       478  0.0  0.3  1024  436 tty5     S    11:32   0:00 /sbin/mingetty tty5
root       479  0.0  0.3  1024  436 tty6     S    11:32   0:00 /sbin/mingetty tty6
root       715  0.0  1.0  2320 1432 tty1     S    11:35   0:00 -bash
root       808  0.0  0.5  1076  436 ?        S    11:35   0:00 /usr/sbin/gpm -t ps2 -m /dev/mouse
root       819  0.0  0.6  2460  896 tty1     R    11:35   0:00 ps auxww
mdoelle@linux:~ >
```



```
mdoelle@linux:~ > grep -v "^#" /etc/inetd.conf
time     stream  tcp     nowait  root    internal
time     dgram   udp     wait    root    internal
ftp      stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd
telnet   stream  tcp     nowait  root    /usr/sbin/tcpd  in.telnetd
shell    stream  tcp     nowait  root    /usr/sbin/tcpd  in.rshd -L
login    stream  tcp     nowait  root    /usr/sbin/tcpd  in.rlogind
talk     dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
ntalk    dgram   udp     wait    root    /usr/sbin/tcpd  in.talkd
pop3     stream  tcp     nowait  root    /usr/sbin/tcpd  /usr/sbin/popper -s
finger   stream  tcp     nowait  nobody  /usr/sbin/tcpd  in.fingerd -w
http-rman       stream  tcp     nowait.10000    nobody  /usr/sbin/tcpd  /usr/sbin/http-rman
swat     stream  tcp     nowait.400      root    /usr/sbin/swat  swat
mdoelle@linux:~ >
```

name to enter, in reality he is being handed the content of an entire hard disk. In particular via **buffer overflows**, where entries are simply considerably larger than anticipated by the programmer, access to a shell has been achieved via diverse utilities – preferably with root privileges.

Full cover

So the motto has to be: Shut down anything which is not totally necessary, and protect everything else as well as possible. This will not bring you complete security, but it should make life as difficult as possible for attackers. The most important part of this is making as little information as possible about the system that is running available to the cracker. The front line here is held by the *finger* daemon, which carries information about the user registered in the system into the outside world. The corresponding line in the */etc/inetd.conf* should therefore be decommented on all computers, by placing a hash (#) at the beginning. There are also various tools, which differ from distribution to distribution, for editing the */etc/inetd.conf*, but in most cases these cause more confusion than benefit – at this point, we recommend making changes properly using text editors such as *kedit* or *emacs*.

The same fate as finger should also await the utilities also entered in the /etc/inetd.conf, telnet, shell and login. These allow unprotected and unencrypted access from outside, where the password entered wanders, completely open and explicitly identified as such, throughout the entire Internet. In particular, access via telnet is still one of the biggest security loopholes overall. Even today, many Internet providers still offer Telnet accesses, even though they are well aware that thousands of programs filter the data traffic day after day and are just spying out such passwords (sniffing) in order to infiltrate the system at the next opportunity. Here you should insist on an encrypted access via SSH, and you can find out more on this in the SSH & Co. boxout.

The POP3 utility also allows a log-in onto your system, even if it is a bit rough and ready, but in the past this was also a point of attack. You only need this if other computers are collecting mails from you, for example because you have constructed a home network. In all other cases, the line beginning with *pop3* should be provided with a hash and thus be decommented. The *talk* daemon is entered under *talk* and *ntalk*. If this is running, a connection can be made from local and remote computers via *talk* to users on your machine. We really do feel this is not a good idea: If the output of messages has not been expressly deactivated with *mesg no* in *xterm* or on the console, the connection request of the remote station is displayed at the cursor position – completely regardless of whether you are currently editing a text or copying files in Midnight Commander. This regularly messes up the screen and also makes the computer bleep. In our view, this is an unnecessary bother, which can be countered by decommenting both lines in the */etc/inetd.conf*. Apart from this the Talk program is anything but easy to use, which is why most users prefer IRC (Internet relay chat). Last but not least, by decommenting *talk* and *ntalk* we are closing another potential security loophole – because the daemon *in.talkd* is also started with root privileges.

SuSE Help

One utility, which has been found, especially in SuSE distributions, for years, is hidden away behind *http-rman*. This is a Web attachment for the *rman* command, which converts manual pages into other file formats. It is used for the internal SuSE help system, which displays man pages with it. Unfortunately this utility can be accessed from outside. Even if it is running as user *nobody*, even if an attacker breaks in, he still has limited access to the system and can look for further opportunities to pilfer some administrator privileges. If you use the SuSE help system, you should protect it with the firewall described in the next issue. Even though no specific attacks are known, nevertheless they are still possible. Only SuSE users are affected by this peculiarity. All that remains now are the two *time* entries, *ftp* and *swat*. Since very few users take the time and date for their own computers from one of the Internet servers, the two *time* entries are only needed in exceptional cases. Even though we are not aware of any successful attack so far, one should usually shut down utilities which are not in use.

The thing about FTP

The *File Transfer Protocol*, FTP for short, is often used to transfer files between two computers. To call up files on a server, one uses a **Client**, an application program, with which one looks for files and can then download them. An FTP **server** is only needed by the other side which provides the data. The great advantage of FTP servers compared with other solutions is that anonymous users can be granted access. Unlike a normal user known to the server, in an anonymous access only the files made available in a special directory can be accessed. In the past few months, though, more and more errors have been found in various FTP daemons, via which in case of doubt, administrator privileges could have been obtained. This is why we recommend that whenever possible you shut down FTP access by decommenting in the */etc/inetd.conf*. When regular users want to transfer files, they can do so with the command *scp*, as described in the SSH & Co. boxout.

Big-mouth Swat

You should also deactivate the last remaining candidates, the configuration program *swat* from the Samba Project. Firstly, *swat* allows

buffer overflow*: When a storage area is filled with more data than it can accommodate, there is an overflow: The data is simply written beyond the end of the area and can overwrite other program parts. In almost all cases this is the fault of the programmer, who has forgotten to include a corresponding overflow check in his program. By continuing to write beyond the boundaries of the area, in some circumstances computer commands can infiltrate, which in turn allow access to the system. In the past, buffer overflows were found in many common Linux utilities, especially in FTP and mail servers.*

Client*: Client computers are usually workstations, which obtain data for example from a server. Client applications are programs called up and used directly by the user, in order to access data or communicate with a server. Typical client applications are browsers, FTP and mail programs.*

Server*: or assistant. Server computers often serve the workstations as a central data store or provide it with Internet and other utilities. No users normally work on it. Server utilities provide the corresponding client applications with the necessary data – so an FTP daemon allows logging in and reading out of files using the FTP-program, while a mail server passes on the user's emails. Daemons are almost without exception server utilities.*

reconfiguration from outside, which means even sensitive data areas of your hard drive could be open to the whole world. Also, an attacker can gather important information about your system by eavesdropping on your connection to *swat*. The greatest problem, though, is that when you use the tool you log on as root and have to enter your root password – this is then transferred, completely unencrypted and clearly marked as a password. This is something akin to hanging up your house key with an address label at a railway station in the hope that nobody will make use of it. If you still want to use *swat* to configure your Samba system, you should at least install a firewall to the Internet, as described below, so that *swat* can no longer be reached from outside.

Dubious linuxconf

The *linuxconf* found in Red Hat and Mandrake is also suspect on security grounds. As with *swat* the password is transferred unencrypted in *linuxconf*, which obviously opens the door to any attacker. Since *linuxconf* also manages almost the entire system, this one tool is all an attacker needs to cause a lot of damage to the computer. If you need *linuxconf*, you should at least protect it with a firewall, or again, deactivate it by placing a hash at the start of the line in the */etc/inetd.conf* . In Figure 3 you can see the fruits of our work by the example of */etc/inetd.conf* in our SuSE system with activated *http-rman* and *swat* (standing in for *linuxconf*).

Shutting down utilities

This means that in an ideal case the */etc/inetd.conf* would be empty, and we could even shut down *inetd*. Depending on the distribution being used, there are various configuration tools for this. In SuSE Linux, in the graphic YaST 2, set *Network/Utilities* switch *inetd on/off* to Off, do not start *inetd*. In Mandrake you do this via the tool *DrakConf*, found there under Startup-utilities (Figure 4). It's easier to use the command line: in (almost) all distributions the start scripts of the individual utilities are stored under */etc/rc.d* in the subdirectory *init.d* and references to the respective start scripts are stored in the subdirectories *rc0.d* to *rc5.d*. Ultimately the utility is started during boot up by means of these references. The utility runs constantly, which is why it should firstly be stopped in an orderly fashion. To do this, call up the script manually, here for example is how it is done with *inetd*:

```
/etc/rc.d/init.d/inetd stop
```

If you would now like to shut down a utility permanently, all you need is to change the name of the respective start script in */etc/rc.d/init.d*:

```
mv /etc/rc.d/init.d/inetd /etc/rc.d/init.d/i➘
netd.no
```

The ending ".no" has been selected at random. This means that the references from the other subdirectories will be running in a vacuum, and the utility will no longer be started. To reactivate, simply rename the start script again with the original name. Bear in mind that you still need root privileges for all these actions.

Ghostbusting: portmap

Even with *inetd* shut down, you still mustn't be lulled into a false sense of security. There are still half a dozen other daemons running, some of them

SSH & Co.

Telnet is still often used to log in on an outside computer. This passes on your own keyboard inputs to the remote station and displays the data from the remote station on your own screen. It is almost as if you had laid a long keyboard and monitor cable to the remote computer. But the connection is transferred in clear text, one-to-one so anyone sitting between your own computer and the remote station (the entire Internet) can follow your progress. The main problem is that even the password necessary to log in is transferred in clear text. Anyone who snaps it up can impersonate you and cause damage at the remote station. The secure shell, SSH for short, remedies this. It works with various encryption procedures that are used right from the start. So an eavesdropper picks up nothing right from the start. In principle, it is in fact possible to encrypt the connection afterwards, but even on large computers this still takes weeks or even months. Also, SSH takes a fingerprint from each computer when a connection is first made. If this does not match at any time, this could indicate an attack on the remote station, with someone else pretending to be the server. In such cases SSH gives a corresponding warning. In addition, with SSH connections it is also possible to call up graphics programs such as a browser on the remote machine – it is still shown on your own monitor, but SSH transfers only the mouse movements and keyboard inputs. The call up from SSH is done according to the pattern:

```
ssh User@Computer
```

Secure Copy

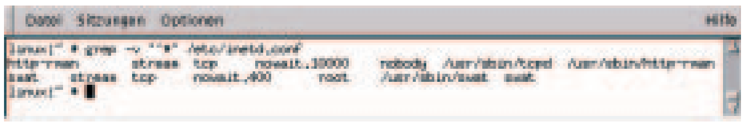
The FTP utility, with which files can be exchanged between two computers, is also very insecure. As with Telnet, the whole connection is unencrypted and can be understood by anyone. So long as it is an anonymous FTP, where one logs on as anonymous user and needs no password, this is not a big problem. But if a regular user wants to transfer his data from the server to his home, he must enter his username and the password – again leaving him wide open to eavesdroppers. Instead of FTP, you can use the command scp:

```
scp User@Computer: file file
```

With this you are copying the file from the remote computer onto your own, where it will be stored as 'file'. The whole thing also works the other way round, as in this concrete example:

```
scp Article.txt modelle@linuxmagazine.co.uk:/home/modelle/Article.txt
```

This causes Article.txt to land on our editorial server. Incidentally scp works in exactly the same way as cp, you can even copy files locally. Generally you should insist on encrypted transfer paths for all connections for which you have to enter passwords so as to make life as hard as possible for attackers.



```
linux:~ # grep -v "^#" /etc/inetd.conf
http-rman       stream  tcp     nowait.10000    nobody  /usr/sbin/tcpd  /usr/sbin/http-rman
swat    stream  tcp     nowait.400      root    /usr/sbin/swat  swat
linux:~ # █
```

Figure 3: Almost all utilities were superfluous: we need *http-rman* for the SuSE help system, *swat* stands in for the system administration program *linuxconf* of other distributions.

with root privileges. So the next thing to do is to collar the "ghosts" which run frequently. The *portmap* daemon is the first one we tackle. This is only required when you want to make parts of your own hard drive available to others over the network. In most cases, you do not. First stop *portmap* with */etc/rc.d/init.d/portmap stop*, then rename, as shown by the example of *inetd*, the file */etc/rc.d/init.d/portmap*:

```
mv /etc/rc.d/init.d/portmap /etc/rc.d/init.d↵
/portmap.no
```

Name Server Cache Daemon

nscd, the Name Server Cache Daemon, is a buffer memory for name queries, but also for user and password queries. It is interesting for workstation operation, where the directories of the user are stored in a central server and there is no user information on the local computers. As you can see from Figure 1, it has started precisely seven times on our system. *nscd* is not absolutely necessary to run a single computer and can be removed without any risk:

```
/etc/rc.d/init.d/nscd stop
mv /etc/rc.d/init.d/nscd /etc/rc.d/init.d/ns↵
cd.no
```

One thing you will barely notice is atd, which unlike cron only controls one-off events. But this really useful daemon is rarely used in practice – anyone not using it can also shut it down:

```
/etc/rc.d/init.d/at stop
mv /etc/rc.d/init.d/at /etc/rc.d/init.d/at.no
```

Rascal Cron

Highly suspect, although only in the event of attacks from the local machine, is the *cron* daemon. This allows constantly recurring tasks to be automated such as for example doing a complete back-up every first Monday in the month or calling up new emails every 10 minutes. But not only root can allocate such tasks, so can any normal user. The snag is that *cron* sends the screen tasks, which a program would normally do, by mail to the respective order-giver. With this, by a few contortions, root privileges can be obtained, which is why if there are several users, this should, if possible, be shut down:

```
/etc/rc.d/init.d/cron stop
mv /etc/rc.d/init.d/cron /etc/rc.d/init.d/cr↵
on.no
```

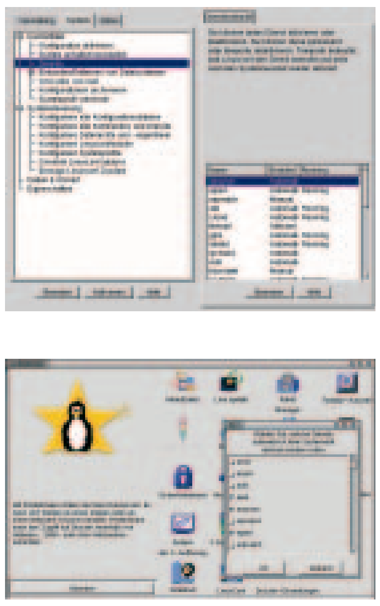


Figure 4: Activation and deactivation is done differently from one distribution to another – here for example in linuxconf under Red Hat (above) and DrakConf under Mandrake.

Cowboys and Indians

In SuSE distributions in particular an Apache Web server is installed as standard and rudimentarily configured. SuSE uses it as part of the help system in order to make help pages or examples accessible via the browser. In the process list, which can be seen in Figure 1, it is hiding behind *httpd*. We feel it is relatively uncritical to allow the Apache server to run, especially since this is also very good for appraising one's own HTML drafts. Nevertheless, an outsider rarely has to look for anything here – and since in any case you get a new address every time you connect to the Internet via modem or ISDN, nor is it that easy to find again. So the Apache can reliably be protected from outside attacks by a firewall – anything which cannot be accessed will be hard to hack. Internally, this has no effects, so the pages can still be viewed by a browser as before. Anyone who really has no use for the Apache might just as well deactivate it and so save a good chunk of memory:

```
/etc/rc.d/init.d/apache stop
mv /etc/rc.d/init.d/apache /etc/rc.d/init.d/↵
apache.no
```

Courier

The last risky utility is *sendmail*. It is still used on many systems for sending emails, although it is often not needed. The problem with *sendmail* is that in the past some security loopholes came to light, with which in some circumstances one can obtain administrator privileges. Whether or not you need *sendmail* depends among other things on the mail programs you use. In most cases emails are collected, via a POP access, from the provider and stored on the local machine. First off, *sendmail* is not necessary to do this. But when sending, many email programs such as *pine* or *mutt* simply pass on the messages to *sendmail*, which takes care of passing them on. On the other hand anyone using Netscape Messenger can pass mails on directly to the server of the provider – *sendmail* is then no longer necessary and can be shut down:

```
/etc/rc.d/init.d/sendmail stop
mv /etc/rc.d/init.d/sendmail /etc/rc.d/init.↵
d/sendmail.no
```

In all other cases you should at least prevent access from outside to *sendmail* by using a firewall.

Firewall

In the next section on the subject of system security we will look at the installation of a simple firewall for domestic purposes. Together with shutting down your utilities, you will then achieve a usable, secured system. ■