Mastering Linux, part 15

Just because you're paranoid doesn't mean they're not after you. *Jarrod Spiga* demonstrates how to protect your Linux system.

JARROD SPIGA

Jarrod Spiga is an infrastructure and network engineer who specialises in getting Linux and Windows systems happily coexisting with each other. He currently holds CCNA and numerous MCP certifications, and dabbles in Web application development in his spare time.

apc on the DVD

BONUS DVD SOFTWARE

- PDFs of every instalment in the Mastering Linux series.

SKILL LEVEL

- Advanced

REQUIREMENTS

- An installation of Linux (Fedora Core 4 and Red Hat Enterprise Linux ES 3.0 were used in the writing of this Workshop).

TIME TO COMPLETE

- 3 hours

IN THIS SERIES

- **Part 14 – January '06** Remote X Windows.
- **Part 15 – February '06** Security and system protection.
- **Part 16 – March '06** Network file services.

One of the reasons many people switch to Linux is because they mistakenly think it's a more secure operating system. While it's true that there are not nearly as many security vulnerabilities for malicious users to take advantage of under Linux, an unpatched, unprotected Linux system can still be compromised, leading to data loss and leaving it open for malicious purposes. Regardless of what operating system you use, you have to patch every hole — but a malicious user only has to find one.

FIREWALLS

When people think about protecting their system from malicious users, the first thing that springs to mind for many is a firewall. Last month (*APC* January, page 118), the basic firewall configuration tool was shown. While this tool doesn't supply the best level of security, it does a reasonable job for users who don't want to learn about the iptables functions of the Linux kernel.

On the other end of the spectrum is the hardcore power user. Instead of using the Security Level Configuration tool, this type of user will handcraft their own firewall script using a large number of iptables commands (not for the faint-hearted though).

Personally, I'm more concerned about efficiency. I look after a large number of Linux systems, and I want to be able to easily change the firewall ruleset on any given server with a minimum of fuss — especially when troubleshooting connectivity issues. In cases such as this, a well-designed firewall script is a solid protection option. While there are dozens of suitable scripts available, Advanced Policy Firewall (APF) from R-fx Networks (**www.r-fx.ca**)has a large number of features and numerous add-ons that provide additional system protection (including brute force detection). Downloads and updates of APF are freely available from **www.r-fx.ca/apf.php**.

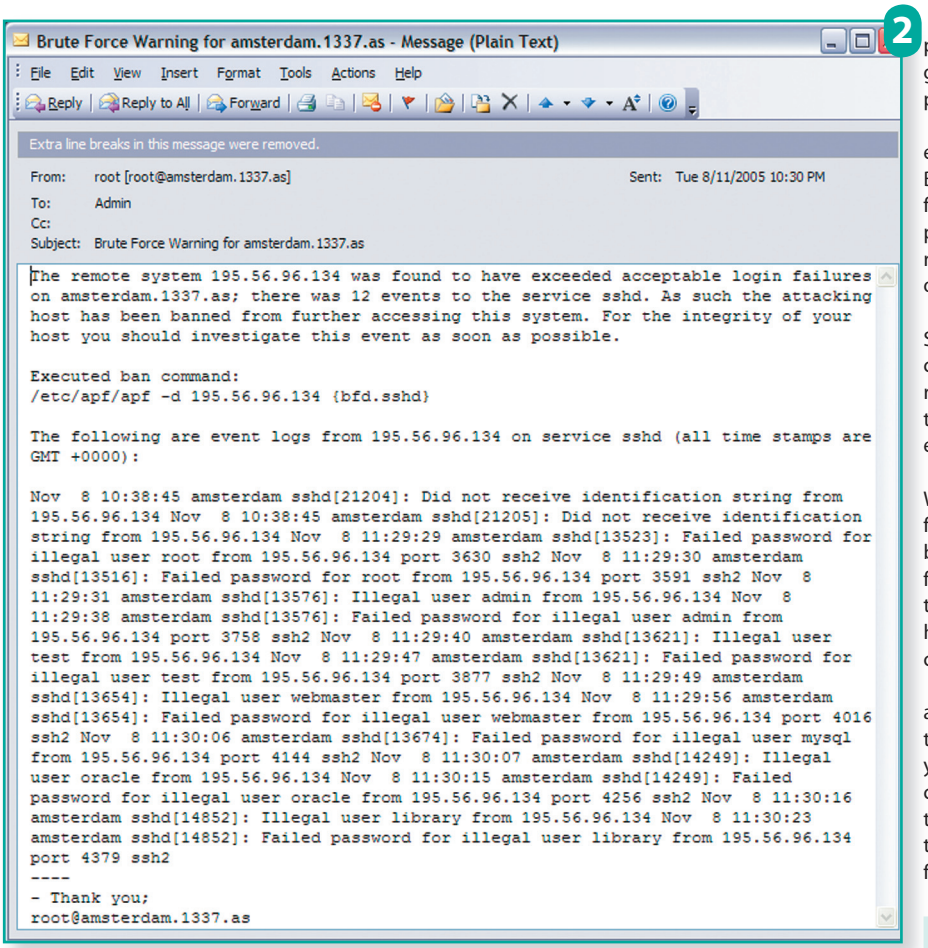
Installing APF is as simple as untarring the tar.gz file and running an install script located in the archive (all as the root user):

```
mkdir /root/apf
tar –zxvf apf-current.tar.gz /root/apf
```

1

```
root@berlin:/etc/apf
# [Ingress]
# Configure ingress (inbound) accepted services. This is an optional
# feature; services and customized entries may be made directly to an ip's
# virtual net file located in the vnet/ directory. Format is comma seperated
# and underscore seperator for ranges.
#
# Example:
# IG_TCP_CPORTS="21,22,25,53,80,443,110,143,6000_7000"
# IG_UDP_CPORTS="20,21,53,123"
# IG_ICMP_TYPES="3,5,11,0,30,8"
##

# Common ingress (inbound) TCP ports
IG_TCP_CPORTS="20,21,22,25,53,80,110,143,443,993,995,19638"

# Common ingress (inbound) UDP ports
IG_UDP_CPORTS="37,53,873"

# Common ICMP (inbound) types
# 'internals/icmp.types' for type definition; 'all' is wildcard for any
IG_ICMP_TYPES="3,5,11,0,30,8"

##
# [Egress]
# Configure egress (outbound) accepted services. This is an optional
# feature; services and customized entries may be made directly to an ip's
# virtual net file located in the vnet/ directory.
#
# Egress filtering is not required but makes your firewall setup complete
# by providing full inbound and outbound packet filtering. You can toggle
# egress filtering on or off with the EGF variable. Format is comma seperated
# and underscore seperator for ranges.
#
# Example:
EG_TCP_CPORTS="21,25,80,443,43"
                                                    290,1         65%
```

Opening doors: by default, APF blocks all inbound requests to your Linux system. You'll need to open the ports for services running on your system.



Instant notification: the Brute Force Detection extension to APF provides proactive blocking of hacking attempts on your Linux system.

```
/root/apf/apf-0.9.6-1/install.sh
```

```
service apf start
```

❶ By default, APF will block all inbound access attempts to your system bar SSH (which runs on port 22). Egress filtering is disabled, allowing you to establish any outbound connection from your system. This default configuration offers a reasonable level of protection and, depending on the services you have running on your Linux box, you may have to edit this configuration.

To make such changes, edit the /etc/apf/conf.apf file. This file lists all of the configuration options for APF as well as detailed descriptions of what each setting does.

If you're testing firewall rulesets from a remote system, ensure that you enable development mode while testing. This mode ensures against implementing a ruleset that blocks your access to the system. If you mistakenly enable such a ruleset, the firewall is disabled after five minutes, restoring your access in the process.

APF runs as a service on Fedora, CentOS and Red Hat Linux distributions. To start the firewall, type:

By default, the firewall will not load up on boot. Once you've created a configuration you're happy with (and have tested it thoroughly), the following commands will set your system to start APF on system boot:

```
chkconfig –add apf
chkconfig –level 345 apf on
```

❷ Once you have APF installed and running, take a look at the Brute Force Detection add-on to APF (**www.r-fx.ca/bfd.php**). ❸ This add-on is especially handy if your system is directly connected to the Internet, as it will dynamically block most brute-force access attempts to your system (BFD running on one of my servers detects on average six different brute-force hacking attempts daily, and dramatically reduces the length and risk of each attack).

ADVANCED FILE PERMISSIONS

Way back in part three (*APC* February 2005, page 98), we covered the basics of file system permissions and how to change the owner and group of files. However, there are a few other permissions settings available for use.

Files can be marked as Set User ID (SUID) executables or Set Group ID (SGID) executables. Both markings achieve similar results — when a file is executed by a user who has executable permissions, it is granted access to system resources as if it was run by the owner user or owner group of the file.

You can tell a file has been marked as SUID or SGID executables by looking at the execute character in a file's user or group field respectively. A regular executable will contain the letter "x" in this field, while an SUID or SGID executable will contain an "s".

❹ Sticky-bits are also assigned to directories. When assigned, all users are allowed to write files to the directory (or create subdirectories), but they are only permitted to modify or delete files they have previously created or files that they own. A sticky-bit enabled directory will have the letter "t" appearing in the execute character of a directory's everyone field.

Much like regular permissions settings, these advanced settings can also be assigned using the chmod command. To set these permissions, you need to supply numeric arguments to chmod, adding an extra digit at the beginning of the permissions code. Binary arithmetic is used to set this extra digit properly, according to the following table:

Value	Meaning
4	Assigns the SUID property to a file
2	Assigns the SGID property to a file
1	Assigns the sticky-bit to a directory

For example, to make a file both SUID and SGID executable for all users, one would enter:

```
chmod 6755 <filename>
```

where <filename> is the name of the file that we are applying these permissions to. The above example will also set the regular permissions on the file so that the owner can edit the file, but others can only read and execute it. To preserve the existing basic permissions, you would use the + sign to add the advanced permissions as follows:

```
chmod +6000 <filename>
```

These advanced permissions can also be removed by simply replacing the + sign with a - sign. From a security perspective, it's important that you assign SUID and SGID properties with extreme caution. When enabled, both properties allow users to

transcend the restrictions placed by their current account settings.

By the same token, you should be wary of any files that are not located in bin directories (ie. /sbin, /usr/bin and /usr/sbin) that have these permissions set. The presence of such files could indicate the presence of a rootkit or trojan installed on your Linux system. The following two commands will generate a list of all SUID and SGID files on your system:

```
find / -perm +2000
find / -perm +4000
```

PROTECTING THE ROOT ACCOUNT

The root account has permission to do anything on a Linux system. If an unauthorised user managed to get root access to your system, they could erase your entire system if they chose to do so. Actually, they'd be more likely to install malicious software and hide that software with a rootkit, which is probably even worse than erasing the entire file system.

As such, you need to protect the root account in as many ways as possible, while still providing yourself with the ability to perform root-user functions.

One of the first things that you can do is to disable direct root logins via SSH. This is even more important if your Linux machine is directly connected to the Internet (and acting as a Web server, for instance).

To do so, edit your /etc/ssh/sshd_config file in the editor of your choice. Uncomment the PermitRootLogin line and ensure that this directive is followed by a single space and the word "no". While you're at it, uncomment the Protocol line, and ensure that this directive is followed by a space and the number 2. This forces you to use SSH 2 to connect to the server, a more secure update to the original SSH specification. Save your changes, exit from your editor and then restart your SSH daemon:

```
service sshd restart
```

In order to gain access to the root account via SSH, users will now need to log on to the system as a regular user and then execute the su command. This forces them to enter two passwords instead of just one.

The next thing you can do is restrict access to the su command. You do this by enabling a special group called "wheel" and ensuring that the su command can only be run by members of the wheel group.

Adding users to the wheel group is simply done by using the gpasswd command (see part 12 — *APC* November 2005, page 112):

```
gpasswd –a <user> wheel
```

where <user> is the name of the user account that you want added to the wheel group. Step two is done by using the chown command to ensure that members of the wheel group have access to the su command:

```
chown root:wheel /bin/su
```

Next, the su command needs to have the correct permissions assigned to it so that members of the wheel group (and only members of this group) can execute it (using the advanced permissions techniques mentioned previously):

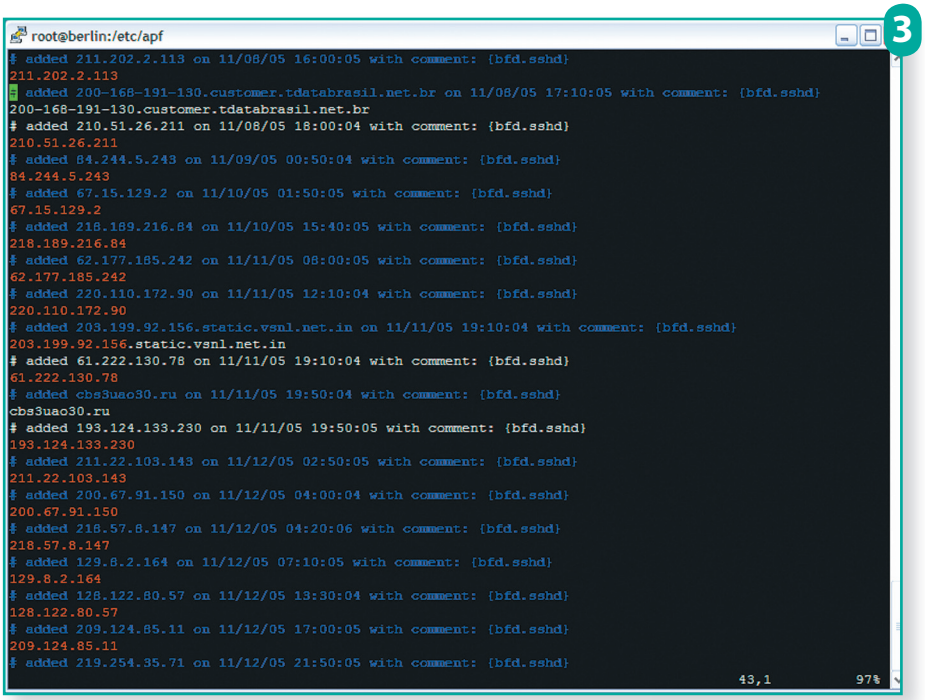
```
chmod 4750 /bin/su
```

The su command now has the following permissions:

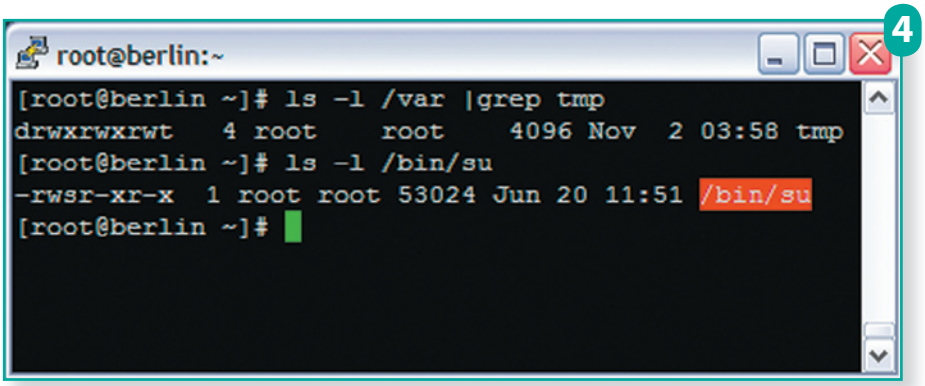
```
-rwsr-x--- root    wheel
```

In a nutshell, this means that members of the wheel group can execute the command, but root privileges are used to execute it. No other groups can even read the command and the root user is the only one who can make modifications to the file.

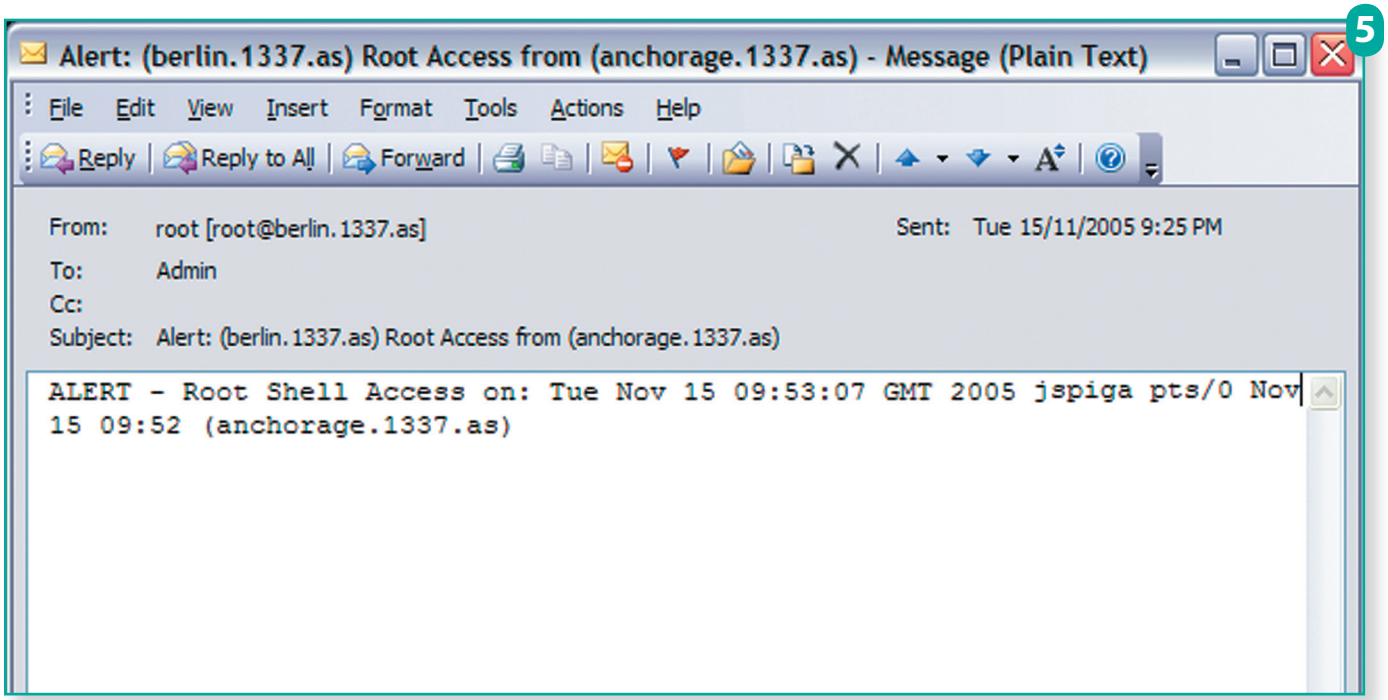
5 The last protection mechanism that will be listed here is to enable an email alert to be sent each time someone logs in as the root user. To do this, edit the /root/.bash_profile file, and add the following line to the bottom of the file:



Denied! Once BFD blocks an attack, the IP address that initiated the attack is listed in /etc/apf/deny_hosts.rules. APF blocks all communication with these IP addresses.



SUID and sticky-bits: /var/tmp is an example of a directory that usually has the sticky-bit enabled. The su command is one of a few files that should always be SUID-executable.



5

```
echo 'ALERT - Root Shell Access on:' `date` `who` | mail -s "Alert: Root Access from `who | awk '{print $6}'`" <email>
```

where <email> is the email address that you would like the email alert to be sent to. It's a good idea to get this email sent to an external email account (that is, an email account not on the system in question). That way, a malicious user cannot delete the system-generated email before you read it.

SETTING LOGIN TIMEOUTS

If your Linux system has multiple people using it at various times, it might be worthwhile implementing login timeouts on your system. With this enabled, any session that has been idle for longer than a defined period of time will be automatically logged out. This setting only applies to command shells — GUI users can still remain logged in to the system.

Each of the scripts that are located in the /etc/profile.d directory are executed each time any user logs on to the shell. Therefore, by adding a simple script that sets the timeout to this directory, the timeout will apply to everyone who logs on to the system:

```
vi /etc/profile.d/timeout.sh
```

All that this script needs to do is set the TMOUT variable as follows:

```
TMOUT=<time>
```

where <time> is the number of seconds that you wish to set the timeout to. In general, a value of 1800 (30 minutes) is a fairly generous timeout, but you may wish to change this value depending on your circumstances.

After creating the file, you should ensure that it is able to be executed by all users:

```
chmod ugo+x /etc/profile.d/timeout.sh
```

One last caveat about this script is that it will only be executed if users are using the bash shell. The other popular shell of choice is the C shell (csh, and its variants). A second profile.d script can take care of users using this shell:

```
vi /etc/profile.d/timeout.csh
```

The format of the timeout command is slightly different for this shell:

```
set autologout=<time>
```

where <time> is the login timeout in minutes. Again, you'll need to make sure that the script you've created is executable by all users:

```
chmod ugo+x /etc/profile.d/timeout.csh
```

RESTRICTING SHELLS

It's not possible to set login timeouts on most other shells. If you'd prefer to restrict the users of the system from being able to access other shells which may be installed on your Linux system,

Who's using root? This email notification tells me that I've just used the su command to login as the root user on a server.

edit the /etc/shells file. To prevent access to a shell, simply place a # in front the shell's command path to comment out the line.

OTHER SECURITY TIPS

These items are just the tip of the iceberg when it comes to Linux security. Some other things to consider in order to harden up your box are to:

- Disable Telnet from within xinetd (**www.phy.bnl.gov/cybersecurity/telnet.html**).
- Install an intrusion detection service such as Tripwire (**www.tripwire.com/**).
- Install and periodically run rootkit detection applications such as chkrootkit (**www.chkrootkit.org**) and rkhunter (**www.rootkit.nl/**).
- Mask service names and version numbers on various system services (especially PHP, Apache, MySQL, FTP and SMTP services).
- And finally, educate users on having strong and secure passwords. apc

INSIDE INFO

- APF (Advanced Policy Firewall) **www.r-fx.ca/apf.php**

NEXT MONTH

- In next month's Mastering Linux Workshop, we'll explore network file systems and Samba. At the end of this guide, you'll be able to share files more easily between Linux and Windows systems.