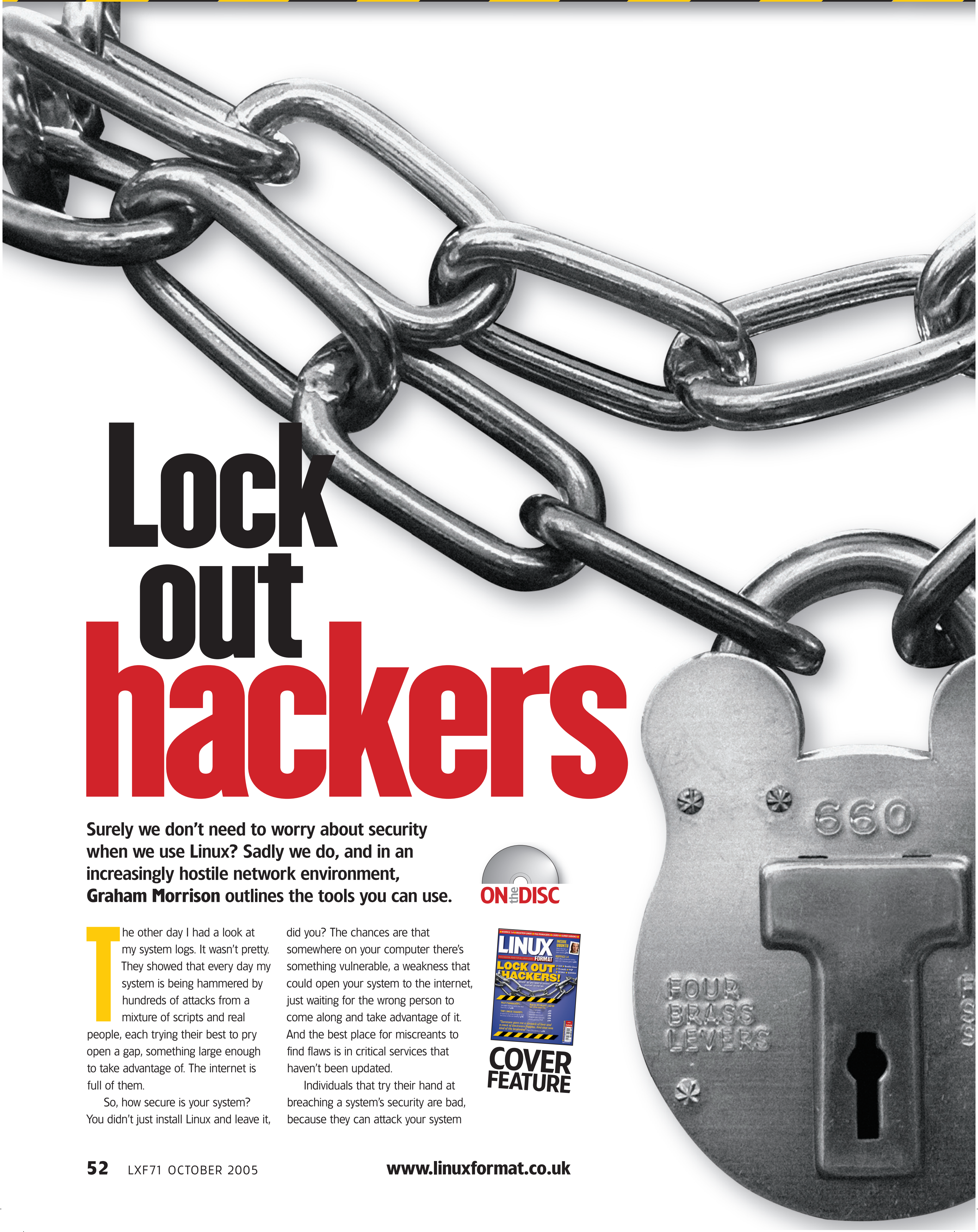


Lock out hackers

Surely we don't need to worry about security when we use Linux? Sadly we do, and in an increasingly hostile network environment, **Graham Morrison** outlines the tools you can use.

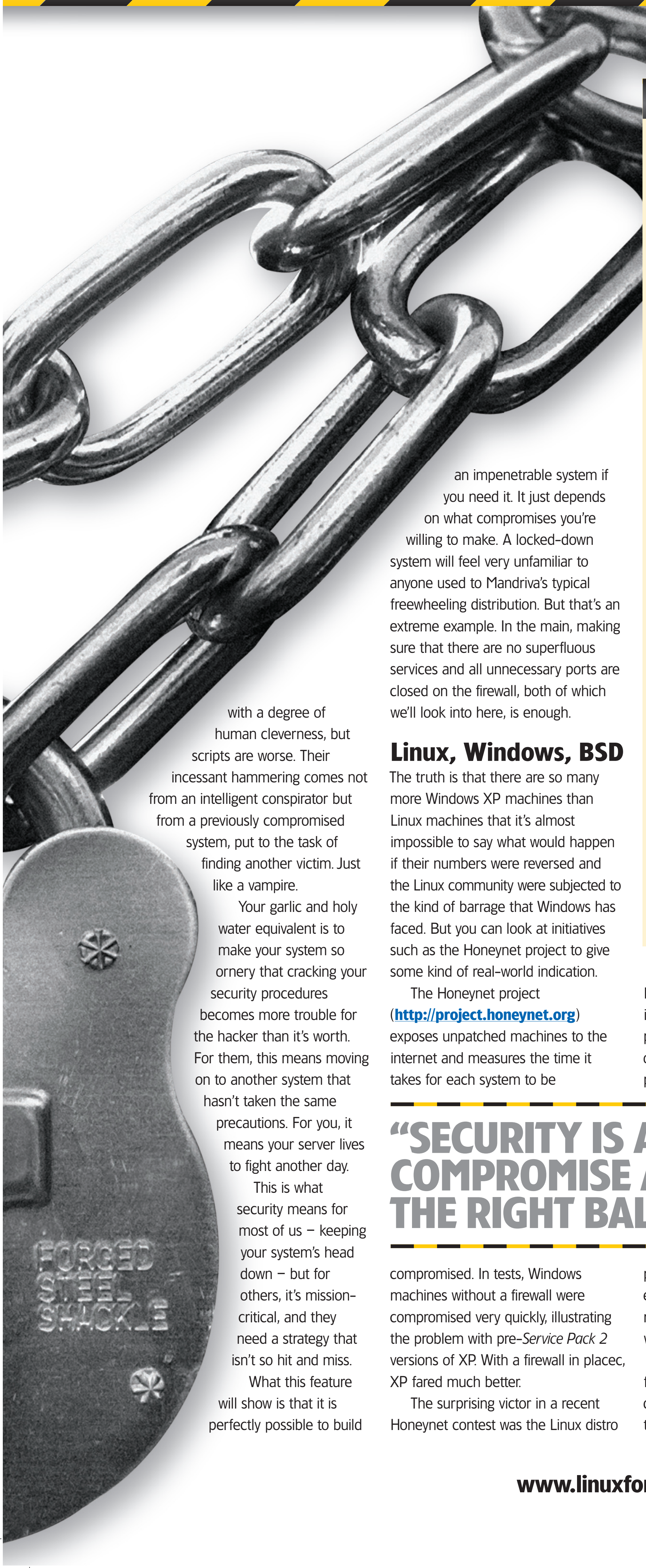
ON the DISC

COVER FEATURE

The other day I had a look at my system logs. It wasn't pretty. They showed that every day my system is being hammered by hundreds of attacks from a mixture of scripts and real people, each trying their best to pry open a gap, something large enough to take advantage of. The internet is full of them.

So, how secure is your system? You didn't just install Linux and leave it, did you? The chances are that somewhere on your computer there's something vulnerable, a weakness that could open your system to the internet, just waiting for the wrong person to come along and take advantage of it. And the best place for miscreants to find flaws is in critical services that haven't been updated.

Individuals that try their hand at breaching a system's security are bad, because they can attack your system



with a degree of human cleverness, but scripts are worse. Their incessant hammering comes not from an intelligent conspirator but from a previously compromised system, put to the task of finding another victim. Just like a vampire.

Your garlic and holy water equivalent is to make your system so ornery that cracking your security procedures becomes more trouble for the hacker than it's worth. For them, this means moving on to another system that hasn't taken the same precautions. For you, it means your server lives to fight another day.

This is what security means for most of us – keeping your system's head down – but for others, it's mission-critical, and they need a strategy that isn't so hit and miss.

What this feature will show is that it is perfectly possible to build an impenetrable system if you need it. It just depends on what compromises you're willing to make. A locked-down system will feel very unfamiliar to anyone used to Mandriva's typical freewheeling distribution. But that's an extreme example. In the main, making sure that there are no superfluous services and all unnecessary ports are closed on the firewall, both of which we'll look into here, is enough.

Linux, Windows, BSD

The truth is that there are so many more Windows XP machines than Linux machines that it's almost impossible to say what would happen if their numbers were reversed and the Linux community were subjected to the kind of barrage that Windows has faced. But you can look at initiatives such as the Honeynet project to give some kind of real-world indication.

The Honeynet project (**http://project.honeynet.org**) exposes unpatched machines to the internet and measures the time it takes for each system to be compromised. In tests, Windows machines without a firewall were compromised very quickly, illustrating the problem with pre-*Service Pack 2* versions of XP. With a firewall in placec, XP fared much better.

The surprising victor in a recent Honeynet contest was the Linux distro Linspire, a system that usually comes in for a lot of criticism because it provides root access by default. But, out of the box, it opens just a single port on the firewall, and that's only for ping requests. No other services, not even SSH, are provided, and that means no other vulnerabilities. That's why Linspire survived.

But Linspire isn't designed for the front line; it's unashamedly for the desktop. OpenBSD is the opposite, a tool that stands on its security record and is constantly being audited for potential security problems. As Theo de Raadt, OpenBSD's founder, puts it, "Whenever you try to use a single method to solve a problem, it either becomes too slow or it becomes too complicated – and when it becomes complicated, it ends up having security problems."

"SECURITY IS ALL ABOUT COMPROMISE AND FINDING THE RIGHT BALANCE."

A question of balance

This is what security issues really come down to. You could disconnect your computer from the internet and know it would be safe. But the moment you make that connection, you're weighing the advantage of being connected to the whole world against the potential threat that brings to your system. Security is all about compromise, and finding the right balance. »

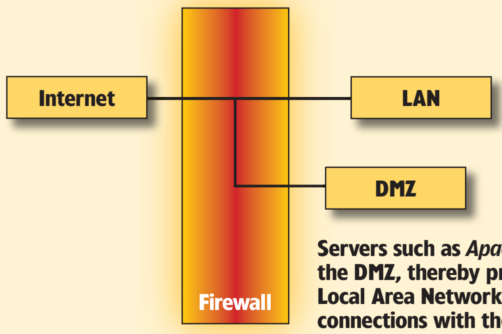
THE SECURITY RISKS TO LINUX USERS

CLASS Desktop
WEAKNESS Incoming email may contain viruses or trojans that can be inadvertently opened, and popular applications such as the *Firefox* browser can be taken advantage of.
CONSEQUENCE Spyware mines your computer for valuable information, such as cookies that contain connection information and browsing history.
CURE Keep an up-to-date virus checker and browser. Try the *ClamAV* virus checker (*see page 54*). Make sure browser cookies expire.

CLASS Home network
WEAKNESS A mis-configured firewall that lets all and sundry connect to your system.
CONSEQUENCE Personal information contained on your network may be exposed to the internet.
CURE Close every unnecessary port on your firewall – if you're not sure how, follow the instructions in this article on page 54. Restrict services to the absolutely essential.

CLASS Wireless network
WEAKNESS A network that doesn't use encryption. Using the default access point name is also a problem.
CONSEQUENCE Anyone in the immediate vicinity of your network can use your internet connection, or even worse, browse through your files.
CURE Use encryption (explained on page 58). WPA is best, but WEP is better than nothing. Change the name for the access point, or turn off broadcasting of the name completely.

CLASS LAN
WEAKNESS Insecure services that allow access to the PCs on your network.
CONSEQUENCE Machines can be controlled from anywhere in the world, then used for sending spam or as part of a Denial of Service attack.
CURE Create a demilitarised zone (DMZ, *below*), an area that sits between the LAN and the internet, removing the need for a direct connection.



Servers such as *Apache* run from the DMZ, thereby protecting your Local Area Network from direct connections with the internet.

Guard your network

Why a good firewall is the single most important security tool a Linux user can install.

In recent years, nearly every single computer connected to the internet has faced a constant bombardment from attackers, human and scripted, trying to get access to your system.The main draw for this intrusion used to be a mixture of over-zealous inquisitiveness and intellectual challenge, but those days are long gone. Today, attempts to breach your security are highly organised, and that's because it's become profitable. Whether it's for sending thousands of spam messages a day or to take control of servers for use in a Denial of Service attack, getting control of your system now has real value.

Wall of fire

As a sign of its importance, the word 'firewall' has entered the vernacular as the way to defend your computer against all the nefarious threats from the internet. This is probably because it conjures up images of an impenetrable, windowless wall, protecting your domain from the dark and threatening forest.

Of course, this is a gross oversimplification. A firewall like this would let nothing in and nothing out. In reality you need to filter, not block, and a firewall is the tool to control what to accept and what to reject.

A good example of a Linux service that shouldn't be exposed to the internet (unless you know what you're doing) is the network filesystem (NFS). It's inherently insecure, but incredibly useful over a Local Area Network. A firewall means you can use NFS within the local zone, and block access to it from any other.

The Local Area Network is your own domain where you can connect computers with a degree of trust. The transition from LAN to internet needs to be guarded by a firewall, which restricts access to secure services only. You could set it so that only SSH is allowed to communicate with the internet from within the LAN.

Blocking every external connection to your system is a little draconian. There are many occasions when you need to access services on your system from the internet, whether it's for running a web server or providing a remote desktop. When you need to leave ports open in your firewall, the security emphasis moves from prevention to damage limitation, and that means staying on top of any security flaw, and being constantly prepared to update any services you may run.

Apache is the perfect example. It's responsible for servicing 68% of all the web pages we view from the internet. Something this fundamental is always going to be a prime target and sure enough, *Apache* has more than its fair share of security alerts. But the point is, if you're going to be running a service that you know will be accessible from the internet, you absolutely have to update it whenever there's a known vulnerability.

"THERE'S NOTHING TO STOP A VIRUS FROM DELETING FILES FROM YOUR OWN HOME DIRECTORY."

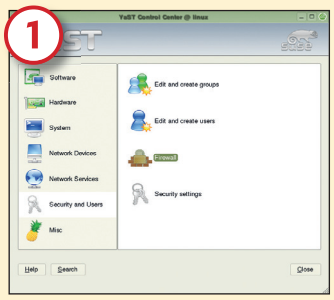
How to use ClamAV

If malware can't get in through the front door, it needs to find another way, and the best way of guarding against such an attack is to use a virus checker. There is an excellent GPL virus checker called *ClamAV*, which works either from the command line or directly from your mail reader.

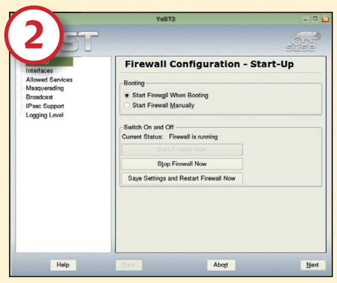
You may think that an open source virus checker, without the same commercial drive to vanquish security threats, would not be as secure as a paid-for product. In fact, the *ClamAV* team are almost as reliable as the commercial vendors, making *ClamAV* an excellent solution for the majority of Linux users.

To get it working on your machine, you need to first either install a package your distribution provides or download the latest version from *ClamAV*'s website (**www.clamav.net**). Compiling it yourself is easy, but you

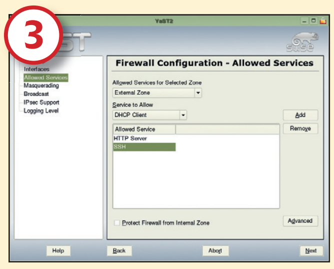
CONFIGURE YOUR FIREWALL



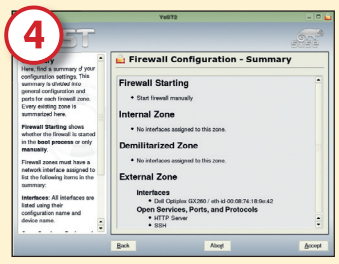
With SUSE, you can configure your firewall from *YaST*. The module can be found in Security And Users and clicking on Firewall. Mandriva uses a similar tool available from its own Control Center.



The Firewall Configuration window has seven pages. In the Interfaces page, select the device that's connected to the internet. Then click on Allowed Services, which is where you can open and close ports in the firewall.



By default, SUSE allows access to only the HTTP server from the internet. To add more, such as SSH, make sure the zone is set to External and choose the service you require from Service To Allow list.



Here, find a summary of your configuration settings. This summary is divided into general configuration and parts for each firewall zone. Every existing zone is summarized here.

Firewall Starting shows whether the firewall is started in the **boot process** or only **manually**.

Firewall zones must have a network interface assigned to list the following items in the summary:

Interfaces: All interfaces are listed using their configuration name and device name.

Firewall Configuration - Summary

Firewall Starting
- Start firewall manually

Internal Zone
- No interfaces assigned to this zone.

Demilitarized Zone
- No interfaces assigned to this zone.

External Zone

Interfaces
- Dell Optiplex GX260 / eth-id-00:08:74:18:9e:42

Open Services, Ports, and Protocols
- HTTP Server
- SSH

When you've finished setting up the firewall, finish the process by clicking on Next. *YaST* will then display an overview of which services are going to be accessible from where. If you're happy, click Accept to make it active.

need to create a 'clamav' user first, and make the data files accessible to this user after installation:

```
useradd clamav
groupadd clamav
cd clamav-0.86.1/
./configure --prefix=/usr
make install
chown -R clamav /usr/share/clamav
```

You also need to edit a couple of config files: **/usr/etc/clamd.conf** and **/usr/etc/freshclam.conf**. The most important thing is to remove or delete the Example line in both. **clamd.conf** needs Adjust User, LocalSocket, LogFile, LogFileMaxSize and ArchiveMaxFileSize to be checked over; while **freshclam.conf** needs Adjust UpdateLogFile, DatabaseOwner, DatabaseMirror, and Checks in.

You will find that the default values should work in most cases, except for DataBase Mirror, which needs to have your country code inserted (UK for example).

Next, create a couple of the required files and directories, and make sure they're accessible by the clamav user:

```
mkdir /var/lib/clamav
chown -R clamav:clamav /var/lib/
clamav
touch  /var/log/freshclam.log
chown clamav:clamav  /var/log/
freshclam
```

You should then be able to download the latest virus database by executing the **freshclam** command. If it ran successfully, you should see something like:

```
ClamAV update process started at Fri
Jul 22 09:39:37 2005
Downloading main.cvd [*]
```

Finally, start both the *ClamAV* daemons by running **freshclam -d** and **clamd**.

Scanning individual files is accomplished using the **clamscan** command, and you can make sure everything is working correctly by scanning the **test** directory contained in the downloaded **clamscan** tarball. If it's all working correctly, you should see something like this:

```
# clamscan clamav-0.86.1/test
---------- SCAN SUMMARY --------
Known viruses: 36088
Engine version: 0.86.1
Scanned directories: 1
Scanned files: 7
Infected files: 5
Data scanned: 0.00 MB
Time: 1.101 sec (0 m 1 s)
```

It doesn't take long, and it's well worth the effort.

SECURITY TIP APACHE

Some hits to your website will come from programs called robots. Many of these gather data for search engines and are known as spiders.

A well-behaved robot is supposed to read and OK the robots.txt file in your site's home directory. This file tells it which files and directories may be scheduled. You should have a robots.txt file in the top directory of each website.

Exclude all directories with CGI scripts (anything marked as ScriptAlias, such as /cgi-bin), images, access-controlled content or any other content that should not be exposed to the world.

Here's a simple example:

```
User-agent: *
Disallow: /image_dir
Disallow: /cgi-bin
```

»

MARK COX ON APACHE SECURITY

Mark J Cox runs the Red Hat security response team and is part of the security teams for both *Apache* and OpenSSL. We asked him just how vulnerable *Apache* is.

"I believe many people have a perception that the *Apache* web server has a poor security history, but this is really a myth. The Apache Software Foundation defines a critical security issue as one that could be exploited remotely to gain access to a server; something that a worm could exploit. This also matches the Microsoft definition of a critical flaw. In the history of released versions of *Apache* since 1.3.0 in 1998 to July 2005 there have only been five critical flaws:

Windows batch files
March 2002, *Apache1.3*.
This could be exploited on Windows systems that had batch file CGI scripts.

Chunked encoding flaw
June 2002, *Apache1.3* and *2.0*.
An attacker could send malicious requests that on some BSD systems could execute arbitrary code.

MS-DOS device filtering
January 2003, *Apache 2.0*.
On Windows platforms only a remote attacker could execute arbitrary code by sending carefully crafted requests.

APR vulnerability
May 2003, *Apache 2.0*.
This could allow an attacker to execute arbitrary code if a server is using modules such as mod_dav.

IPv6 heap overflow
September 2004, *Apache 2.0*.
Could allow remote code execution on some BSD systems.

You can see that running *Apache* on Linux you'd only have been vulnerable to one of those critical issues. However, as a server administrator it's likely that you'll be using other software in conjunction with the *Apache* web server; for example if you are running a secure server you'll be using *OpenSSL*. This has had critical security issues in the past; the last was in July 2002 and was subsequently exploited by the *Slapper* worm.

To protect against future worms some Linux vendors have started shipping with enhanced protection. Fedora Core has a number of layered security technologies designed to prevent common exploits for flaws such as buffer overflows and double-frees, and randomisation to try to reduce the risk from worms by adding diversity. Fedora Core even uses an SELinux-targeted policy by default to protect *Apache*. Even so it's important to keep systems up to date and the best way to avoid being caught out by flaws in the future is to subscribe to the security announcement lists for the software you are using. Linux distribution vendors have their own lists and these explain how each issue uniquely impacts the particular distribution."

"THE BEST WAY TO AVOID BEING CAUGHT OUT IS TO SUBSCRIBE TO THE SECURITY ANNOUNCEMENT LISTS FOR THE SOFTWARE YOU ARE USING."

Harden your system

Learn about *Bastille*, *Nessus* and *John The Ripper* – tough-sounding tools that do a dirty job.

If you're not happy with the way a typical distribution automatically configures your system, there are several well-trodden paths to making it more secure. Hardening your system in this way lets you get it just right by removing users and services that are superfluous to your requirements.

How far you take the process is entirely down to how critical security is to your system. The average *Apache* server isn't going to need a system that can survive a nuclear attack but it will benefit from some hardening, whereas a back-office payroll machine will have to be almost invisible. The most important part of hardening your system is assessing the risk, and putting a plan into action that reduces that risk to a minimum.

Many distributions tailor the packages they include to their target audience. You only need to compare Red Hat Enterprise Linux with Fedora, or look at some of the kernel components removed from the typical enterprise distribution, to realise that the 'less is more' approach is paramount for server security.

As Red Hat's Mark Cox pointed out when asked which part of a standard Linux distro is the most vulnerable: "That depends on what you consider to be vulnerable. If you're running a web server, then all you care about are your web applications, and if you're running a mail server you care about *Sendmail*. With regards to what's likely to be most vulnerable, it's probably the kernel."

A storming script

Bastille is a Perl-based hardening script that takes the system administrator by the hand on a security-based tour of their system. *Bastille* intends to educate as well as harden, and each stage goes to the trouble of explaining much of the rationale behind each procedure. This includes changing file permissions, forcing passwords to be renewed and locking down unused services. These are good tactics with or without *Bastille*, but using a script such as this makes it much easier, especially when you can save your configuration for use on other similar systems.

Bastille can be downloaded from **www.bastille-linux.org**, and there are packages available for most popular distributions, including Red Hat, Mandriva and SUSE. The only requirement is that you have Perl installed along with Perl/Tk for the graphical interface.

To execute the script, just type **bastille** as root. You should then be asked to accept a disclaimer before presenting the user with the simple Tk user interface.

For a guide to *Bastille*, follow our *Fortify With Bastille* steps, below.

Another popular tool for securing your system is *Nessus*. This is based on the same principle that hackers use to find security holes, and that's scanning your computer for services and known vulnerabilities.

These may be an old version of *Apache* or OpenSSH, but they could equally be a badly-chosen password or an open port in your firewall. It's this all-encompassing approach that has made *Nessus* the most popular open

SECURITY TIP USING SUDO

The *su* command is part of every flavour of Linux – indeed, every flavour of Unix, period. But it's a little limited; to run a shell or command as another user, *su* requires you to enter that user's password and essentially become that user (albeit temporarily). But there's an even better command you can use: *sudo*, the 'superuser do'. This lets you run a specific privileged command without actually becoming root, even temporarily. Unlike with su –c, authority can thus be delegated without having to share the root password, thus:

```
sudo ifconfig eth0
```

THE SELINUX SOLUTION

The National Security Agency is responsible for more than Jack Bauer.

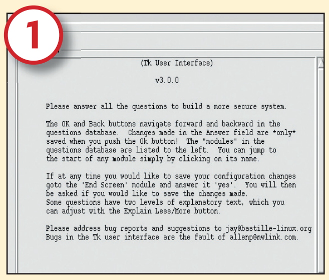
If you need the ultimate in security solutions, one of the main contenders has a pedigree quite unlike any other security product, hailing from the US National Security Agency (NSA), that bastion of secrecy and 24-hour television fiction.

This is SELinux, a series of patches for the Linux kernel with some supporting utilities that can enforce mandatory access control, isolating processes and their memory space from other processes and users. There are SELinux packages available for Fedora Core and Gentoo, but it can also be built from source code provided by the NSA. Interestingly, Novell has taken a different route, by acquiring a company that provides an alternative to SELinux called Immunix.

FORTIFY WITH BASTILLE

Forget making changes manually. With the help of *Bastille Linux* you can feign expert knowledge and build a virtually impenetrable system.

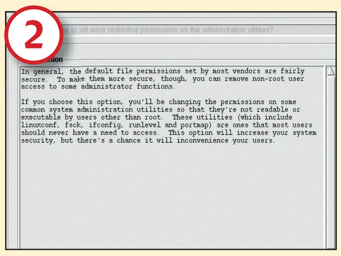
1



```
(Tk User Interface)

      v3.0.0

Please answer all the questions to build a more secure system.

The OK and Back buttons navigate forward and backward in the
questions database.  Changes made in the Answer field are *only*
saved when you push the Ok button!  The "modules" in the
questions database are listed to the left.  You can jump to
the start of any module simply by clicking on its name.

If at any time you would like to save your configuration changes
goto the 'End Screen' module and answer it 'yes'.  You will then
be asked if you would like to save the changes made.
Some questions have two levels of explanatory text, which you
can adjust with the Explain Less/More button.

Please address bug reports and suggestions to jay@bastille-linux.org
Bugs in the Tk user interface are the fault of allenp@nwlink.com.
```

In the Title screen you'll notice that one of the main features of *Bastille* is that each step in the configuration process is accompanied by a detailed description of what is about to change. You can decrease the verbosity of these explanations by clicking on the Explain Less button. Clicking on OK steps you through each *Bastille* security module in turn, putting a small tick to the left of each module when complete.

2



```
...ke to set more restrictive permissions on the administration utilities?

In general, the default file permissions set by most vendors are fairly
secure.  To make them more secure, though, you can remove non-root user
access to some administrator functions.

If you choose this option, you'll be changing the permissions on some
common system administration utilities so that they're not readable or
executable by users other than root.  These utilities (which include
linuxconf, fsck, ifconfig, runlevel and portmap) are ones that most users
should never have a need to access.  This option will increase your system
security, but there's a chance it will inconvenience your users.
```

The first module, called FilePermissions, is one of the more important, as it allows you to restrict access to your system's most important files, and limit the use of 'SUID root'. Both are vitally important to reducing your system's vulnerability. Other modules to look out for are AccountSecurity and MiscellaneousDaemons. The former gives you more control over users' accounts and how their password expire, while the latter lets you safely disable some unnecessary services. Firewall is the only module to be wary of, as it will interfere with your current settings. Make sure you select No for configuring the firewall, unless you know what you're doing.

"NESSUS IS THE MOST POPULAR OPEN SOURCE VULNERABILITY SCANNER IN THE WORLD."

source vulnerability scanner in the world. *Nessus* uses a plug-in architecture, but some of the more important checks include:

- Backdoors.
- Firewalls.
- FTP.
- Gain a shell remotely.
- Peer-to-peer file sharing.
- Remote file access.

Find out more about *Nessus* at **www.nessus.org**. While we're on the subject of obtaining security software, the coverdisc contains many of the applications mentioned in this feature.

Safe passwords

If passwords are the keys that unlock access to your system, it's amazing that these keys are so often badly chosen. So how do you go about selecting a secure password?

It's well known that hackers use so-called dictionary attacks, with scripts pouring through dozens of different word and letter combinations a second. This is most effective if the hacker knows something about the user or the system to which they're trying to gain access. It might sound like fiction, but the art of prising access details away from an inadvertent user (a kind of social engineering) is surprisingly well practised. How many people would genuinely question an internal telephone call at work from someone who claims to be from the IT support desk? Often this is all it takes.

You can use tools such as *John The Ripper* to check for vulnerable passwords on your system, but selecting a good password in the first place avoids many problems. Recent advice has suggested that a tough password written down is preferable to a poor memorable password. The best passwords make no sense, and are a mixture of random characters, including upper and lower case, numerals and non-alphanumeric symbols. The next best thing are words or phrases that have certain characters replaced with others, such as 'cH@7acT3rs' – the more character groups you can fit in the better.

Ripper in action

Despite the fact that *John The Ripper* may appear to be geared for the dark side, it is useful for testing your own security, and is also easy to use. To download and compile the source code, enter the following into the shell:

```
wget 'http://www.openwall.com/john/c/john-1.6.tar.gz'
tar xvzf john-1.6.tar.gz
cd john-1.6/src/
make linux-x86-any-elf
```

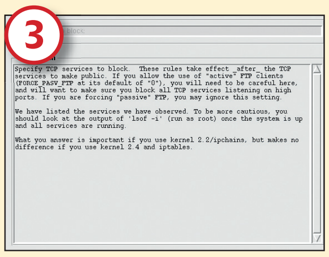
The binary is located in the **run** directory, and is executed by issuing the **john** command followed by the location of your shadow password file, which stores all the encrypted password your system users. Obviously, **john** needs to be run with root privileges to be able to access the shadow password file:

```
./john /etc/shadow
Loaded 3 passwords with 3 different salts (OpenBSD Blowfish [32/32])
test99999        (test)
```

JTR outputs any passwords it manages to crack. In the above example, it has found the password 'test99999' for the user called 'test'. This took *JTR* less than a second. By pressing any key, you can see what stage of the dictionary attack has reached. *JTR* stores the details for each cracked account in the **john.pot** file, which also ensures it doesn't attempt to crack the same passwords next time it's run. You can check which passwords it's successfully managed to crack using the **show** command:

```
./john -show /etc/shadow
test:test99999:12986:0:99999:7:::
1 password cracked, 2 left
```

It's an illuminating program – try it! »



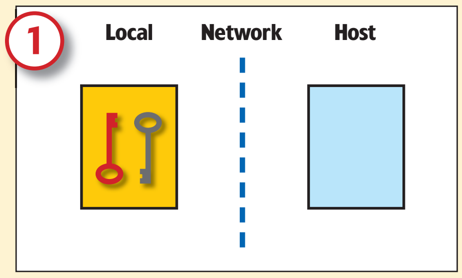
```
Specify TCP services to block.  These rules take effect _after_ the TCP
services to make public. If you allow the use of "active" FTP clients
(FORCE_PASV_FTP at its default of "0"), you will need to be careful here,
and will want to make sure you block all TCP services listening on high
ports. If you are forcing "passive" FTP, you may ignore this setting.

We have listed the services we have observed. To be more cautious, you
should look at the output of 'lsof -i' (run as root) once the system is up
and all services are running.

What you answer is important if you use kernel 2.2/ipchains, but makes no
difference if you use kernel 2.4 and iptables.
```

From the End Screen module you can save the configuration, and as long as you've visited each of the modules, you can also apply the changes. *Bastille* will then go ahead and make all the changes, after which you need to reboot your system for the changes to take effect. You can check all the changes that *Bastille* makes by looking through its excellent logs. Every change *Bastille* attempts to make is logged into /var/log/Bastille/action-log next to a subheading of ACTION, WARNING or NOTE. From here it's easy to tell if things have gone wrong.

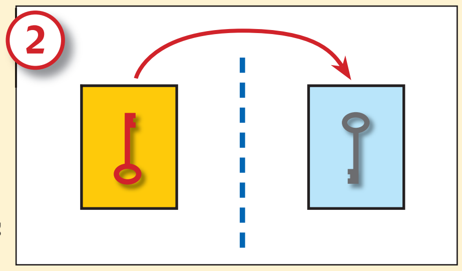
HARDENING SSH

Someone guessing an SSH password is a big threat. Remove this possibility by using private/public key authentication instead.

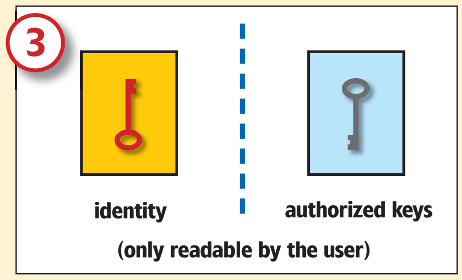
ssh-keygen -t rsa
This generates the key pair to be used for authentication. By default, both keys are written to the user's .ssh directory. If you leave the pass phrase blank, SSH logins will not require user intervention.



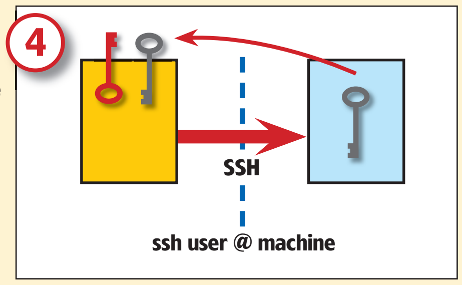
ssh-copy-id -i ~/.ssh/key.pub user@hostname
Copy the public key on to the host machine. Using ssh-copy-id ensures that the public key has the right permissions on the remote host, but there's nothing to stop you doing this manually.



chmod –R 700 ~/.ssh
All keys can only be user-readable. The local private key needs to be renamed as ~/.ssh/identity. The remote public key needs to be renamed as ~/.ssh/authorized_keys.



ssh user@hostname
The remote host uses the public key to encrypt a message that can only be decrypted by your local private key, thus authorising your connection. Finally, enter PasswordAuthentication no into the /etc/ssh/sshd_conf file.



Encrypt your files

How to use keys to keep your data from prying eyes.

Encryption has been around for as long as people have needed to communicate. It can be an incredibly complex subject, with modern cryptography involving advanced mathematics and now moving into the world of quantum physics. The motivation for creating ever more complex methods has been the same for centuries, and that's to ensure that only authorised individuals get access to the data.

On a PC, files can be encrypted individually for securing personal information or to send through an insecure network, such as via email. The next step down is to encrypt a whole filesystem. This makes particular sense for portable devices such as laptops or memory sticks. The bottom level is restricting access to a system via a hardware device, such as a smartcard reader.

Encryption basics

As with a lock, decoding encrypted data always needs a key. The key doesn't need to be a totally separate entity, like the key that unlocks the front door; it could equally be a pass phrase or number that opens a combination lock. It's important to make the distinction between using a pass phrase to access something like your personal accounts and a pass phrase for unlocking a decrypted file. With the former, you're authenticating your identity for a server to grant you access. If you forget your logon details, it's just a matter of getting the administrator to change them. If you lose the key to encrypted data that's a different thing entirely, as it's the key that actually decrypts the data.

One of the first encryption utilities to get a lot of media attention was *Pretty Good Privacy* (*PGP*). It was developed by Philip Zimmermann in 1991 to enable him to communicate securely over a BBS bulletin board, and he made the then unusual step of making the application available for free for non-commercial use, and even included the source code.

Despite problems with the US government classifying the export of encryption software as 'munitions', *PGP* became relied upon for securing all kinds of data, from files to email.

In session

PGP uses a rather clever approach to get around the CPU-intensive nature of asymmetric keys. It encrypts the data using a session key. This is a symmetric key that's randomly generated at the point of encryption. The session key is then encrypted using the recipient's public key, and grouped together as a single package. To decrypt the message, the asymmetric secret key is used to decrypt the session key, which in turn is used to decrypt the main body of data.

PGP was dogged by patent issues in the 1990s, which led to the creation of an open standard called OpenPGP to help other encryption packages interoperate with *PGP*, and the way it stored its various keys.

Probably the most significant development from the OpenPGP standard, especially where Linux is concerned, is the *GNU Privacy Guard*. This is commonly known as *GnuPG*, and has become the standard encryption tool for files and email communication when using open software. You'll find it embedded in desktop stalwarts such as *Evolution* and *KMail*, as well as various front-ends to what is basically a command line utility.

How to use GnuPG

Creating keys with *GnuPG* is as easy as typing the following from the command line:

```
gpg --gen-key
```

You may prefer to use a graphical interface such as *KGpg*, but either way you will need to decide which algorithm to use. You have a choice between DSA and RSA, which are only suitable for signing data, or a combination of DSA and Elgamal. For most uses, the latter option is best.

The next consideration is for the size of the key. On the command line, the default is 2048 bits, but you may find that your software prefers to select a more modest 1024 bits.

The biggest effect of using a larger key is the time it takes to decompress a message, and the *GnuPG* documentation itself suggests that the law of diminishing returns works against too large a key. At a certain point, it will become easier for a potential eavesdropper to break into your house than attack your key.

You then need to provide some personal information that will personalise your key pair. This is usually just your name and email address, but you can also add a comment if you want.

The final stage is the most important, and that's the pass phrase.

"THERE'S NO POINT USING AN UNCRACKABLE ENCRYPTION ALGORITHM IF YOU'RE GOING TO SECURE IT WITH YOUR MOTHER'S MAIDEN NAME."

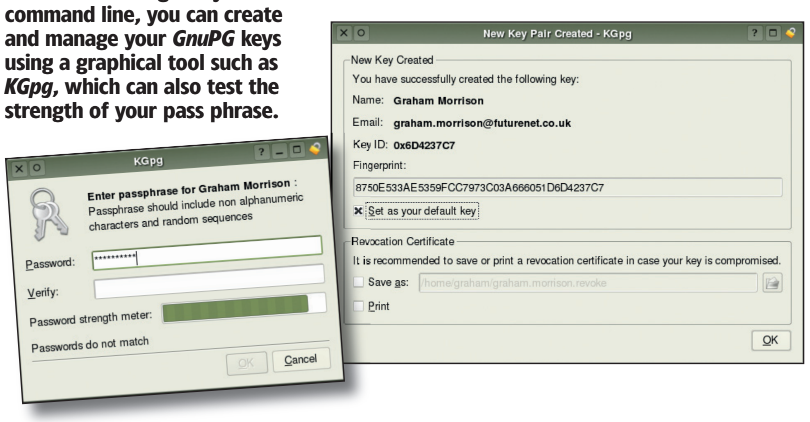
SECURITY TIP
CONFIGURING SYSLOG

Whenever syslogd, the syslog daemon, receives a log message, it acts based on the message type (or 'facility') and its priority. Syslog's mapping of actions to facilities and priorities is specified in /etc/syslog.conf. Each line in this file specifies one or more facility/priority selectors followed by an action; a selector consists of a facility or facilities and a (single) priority.

An example line from syslog.conf is:

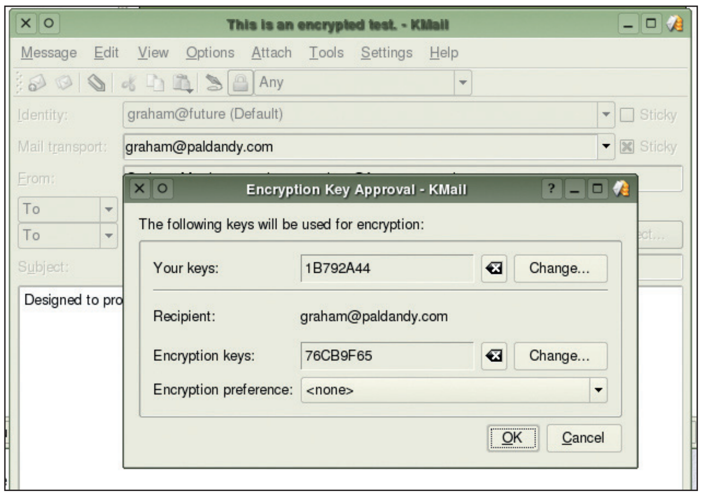
```
Mail.notice                /var/log/mail
```

Within the selector, mail is the facility (message category) and notice is the level of priority.


Instead of slaving away at the command line, you can create and manage your *GnuPG* keys using a graphical tool such as *KGpg*, which can also test the strength of your pass phrase.

KGpg
Enter passphrase for Graham Morrison :
Passphrase should include non alphanumeric characters and random sequences
Password: **********
Verify:
Password strength meter:
Passwords do not match
OK
Cancel

New Key Pair Created - KGpg
New Key Created
You have successfully created the following key:
Name: Graham Morrison
Email: graham.morrison@futurenet.co.uk
Key ID: 0x6D4237C7
Fingerprint:
8750E533AE5359FCC7973C03A666051D6D4237C7
✓ Set as your default key
Revocation Certificate
It is recommended to save or print a revocation certificate in case your key is compromised.
Save as: /home/graham/graham.morrison.revoke
Print
OK


This is an encrypted test. - KMail
Message Edit View Options Attach Tools Settings Help
Any
Sticky
Sticky
To
To
Subject:
Encryption Key Approval - KMail
Change...
Change...
OK
Cancel

Identity: graham@future (Default)
Mail transport: graham@paldandy.com
From:

Designed to pro

The following keys will be used for encryption:

Your keys: 1B792A44

Recipient: graham@paldandy.com

Encryption keys: 76CB9F65

Encryption preference: <none>

Sending an encrypted message requires your key and your recipient's.

This is the key that will unlock files that either you've encrypted yourself or others have sent to you using your public key. It's usually the weakest link in the chain. There's obviously no point in using an uncrackable encryption algorithm if you're going to secure it with your mother's maiden name, at least not if that kind of information is easily available.

Encrypt with KMail

It's all very well creating a public/private key pair, but what do you do with them? *KMail* can manage your keys for you (as can *Evolution*), and makes it easy to sign messages and decrypt any you may receive.

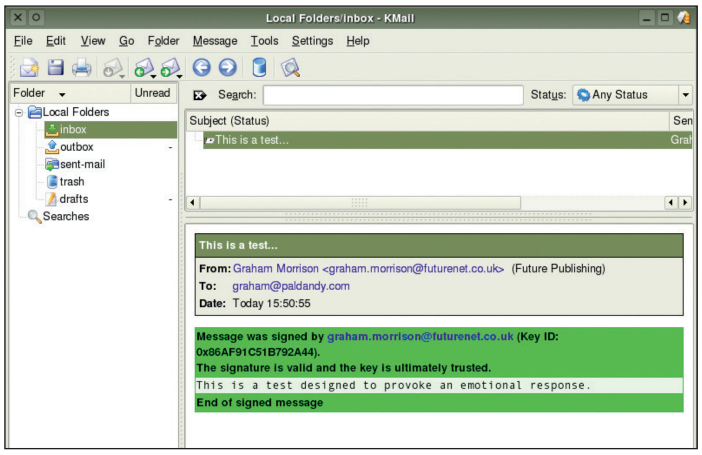
To get all this wonderful functionality, you first need to tell *KMail* which keys to use. Set this by selecting Settings > Configure KMail > Identities. When you open the Identity page, click on the Cryptography tab.

From here you can decide which keys to use by selecting Change. This needs to be done for both OpenPGP signing and encryption.

There are two ways of using encryption from *KMail*. You can sign a message, so that the recipient can verify your identity using your public key, or you can encrypt a message using your recipient's public key to ensure only they can decrypt it.

Both of these tools are available from *KMail*'s Composer window, either via the Options menu or via the stylus and padlock icons on the toolbar. Sending a signed message doesn't require any user interaction, but with an encrypted message you need to select both your own key and that of the recipient *(see screenshot above)*.

When you receive a signed message, as long as the sender's public key is in your keyring, it will be verified automatically. When you receive an encrypted message you will be prompted for your own passphrase. *KMail* uses colours to classify messages, as the screenshot illustrates below. It then stores emails as encrypted messages even after you've decrypted them so they're safely stored on your hard drive. »



This is a test...

From: Graham Morrison <graham.morrison@futurenet.co.uk> (Future Publishing)
To: graham@paldandy.com
Date: Today 15:50:55

Message was signed by graham.morrison@futurenet.co.uk (Key ID: 0x86AF91C51B792A44).
The signature is valid and the key is ultimately trusted.

```
This is a test designed to provoke an emotional response.
```

End of signed message

Thanks to *KMail*'s garish colours, you always know the status of a message.

BRUCE SCHNEIER ON THE THREAT TO ENCRYPTION

Bruce Schneier is a security guru. That might sound like hyperbole, but Bruce has often put his credentials to the test. He developed one of the main contenders for the AES encryption standard, Twofish. It lost out to the Rijndael method, but many consider Twofish to be just as relevant. His interests in security range from network to national. Right now he's working on a diversity of projects including airline passenger profiling, FRID passports, national ID cards and secure voting. We asked Bruce to comment on a few of the problems facing modern encryption techniques, and he replied using a mixture of links to his wonderful security-based web log (www.schneier.com/blog) and his own prose.

***LXF:* Do you think a disproportionate amount of time is spent discussing the strength of one encryption method over another one?**
BS: Security is as strong as the weakest link. Rarely is the encryption algorithm the weak link in a security system. So choose a good algorithm. Choose a good key length. But don't get hung up on the details of the crypto. There are undoubtedly much worse security problems in any system.

***LXF:* How vulnerable is the SHA-1 hashing algorithm and how straightforward will it be to find collisions (data other than the original that creates the same hash)? How should we be using SHA-1 with a view to migrating to a better solution?**
BS: They can find collisions in SHA-1 in 2^{69} calculations, about 2,000 times faster than brute force. Right now, that is just on the far edge of feasibility with current technology.

For the average internet user, this news is not a cause for panic. No one is going to be breaking digital signatures or reading encrypted messages anytime soon. The electronic world is no less secure after these announcements than it was before.

But there's an old saying inside the NSA: "Attacks always get better; they never get worse." Just as this attack builds on other papers describing attacks against simplified versions of SHA-1, SHA-0, MD4 and MD5, other researchers will build on this result. The attack against SHA-1 will continue to improve, as others read about it and develop faster tricks, optimisations, etc. And Moore's Law will continue to march forward, making even the existing attack faster and more affordable.

***LXF:* How can we guard against poorly-implemented encryption, such as with *WinZip's* Advanced Encryption scheme?**
BS: Basically, we can't. We have to trust the vendors.

***LXF:* Do you think passwords should be stuck to the side of your monitor?**
BS: This is good advice, and I've been saying it for years. Simply, people can no longer remember passwords good enough to reliably defend against dictionary attacks, and are much more secure if they choose a password too complicated to remember and then write it down. I recommend that people write their passwords down on a small piece of paper, and keep it with their other valuable small pieces of paper: in their wallet.

"PEOPLE ARE MORE SECURE IF THEY CHOOSE A PASSWORD TOO COMPLICATED TO REMEMBER AND THEN WRITE IT DOWN."

“4 proactive security tools

Essential applications for securing your network.

Ethereal

Network protocol analyser

- **DEVELOPER** Ethereal Software
- **WEB** www.ethereal.com
- **PRICE** Free under GPL

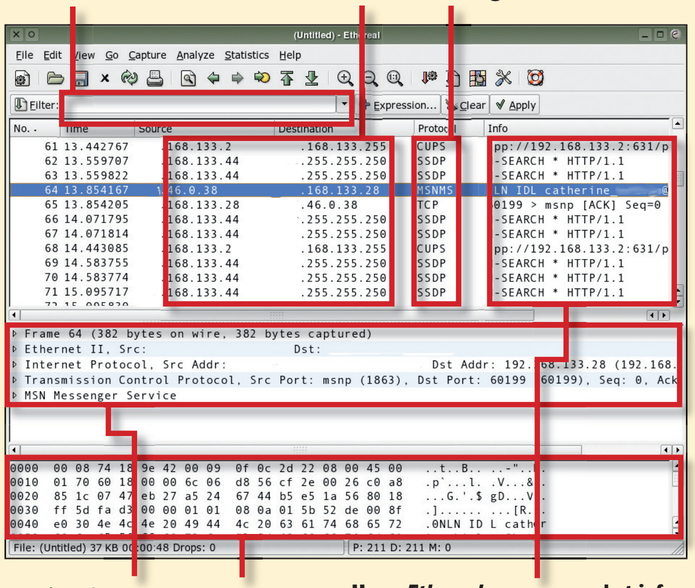
HOW IT WORKS *Ethereal* is the network equivalent of a disassembler, capturing network data to disk and decoding instructions sent across the network. It's just as useful for learning how your network is behaving as it is for tracking rogue packets on the PC.

From the main window you just need to capture the output from your network card, after which it's displayed in the main list view. You can easily see both the source and destination addresses for each packet, as well as the protocol used. After using a tool like this you really start to appreciate how vulnerable you really are.

Limit the list of packets with search criteria.

This is where each packet originated from and where it's headed.

The protocol each packet is using.



```
No. -  Time          Source          Destination       Protocol   Info
    61 13.442767     .168.133.2      .168.133.255      CUPS       pp://192.168.133.2:631/p
    62 13.559707     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    63 13.559822     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    64 13.854167     .46.0.38        .168.133.28       MSNMS      LN IDL catherine_
    65 13.854205     .168.133.28     .46.0.38          TCP        0199 > msnp [ACK] Seq=0
    66 14.071795     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    67 14.071814     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    68 14.443085     .168.133.2      .168.133.255      CUPS       pp://192.168.133.2:631/p
    69 14.583755     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    70 14.583774     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1
    71 15.095717     .168.133.44     .255.255.250      SSDP       -SEARCH * HTTP/1.1

▷ Frame 64 (382 bytes on wire, 382 bytes captured)
▷ Ethernet II, Src:                          Dst:
▷ Internet Protocol, Src Addr:                         Dst Addr: 192.168.133.28 (192.168.
▷ Transmission Control Protocol, Src Port: msnp (1863), Dst Port: 60199 (60199), Seq: 0, Ack
▷ MSN Messenger Service

0000  00 08 74 18 9e 42 00 09  0f 0c 2d 22 08 00 45 00   ..t..B.. ..-"..E.
0010  01 70 60 18 00 00 6c 06  d8 56 cf 2e 00 26 c0 a8   .p`...l. .V...&..
0020  85 1c 07 47 eb 27 a5 24  67 44 b5 e5 1a 56 80 18   ...G.'.$ gD...V..
0030  ff 5d fa d3 00 00 01 01  08 0a 01 5b 52 de 00 8f   .]...... ...[R...
0040  e0 30 4e 4c 4e 20 49 44  4c 20 63 61 74 68 65 72   .0NLN ID L cather

File: (Untitled) 37 KB 00:00:48 Drops: 0          P: 211 D: 211 M: 0
```

Detailed feedback on the packet under scrutiny.

The packet's contents.

Here, *Ethereal* sums up packet info it thinks is important, such as who you're chatting with in *Messenger*.

KSytemLog

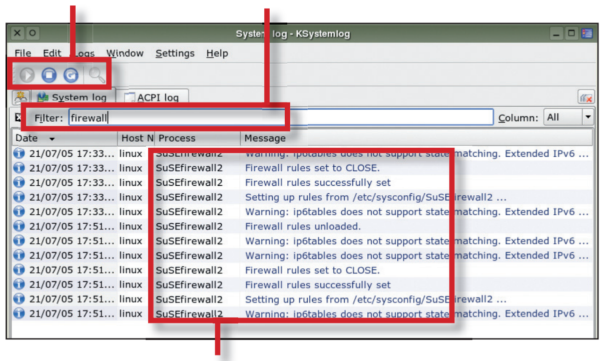
KDE system log viewer

- **DEVELOPER** Nicholas Ternisien
- **URL** http://annivernet.free.fr/ksystemlog
- **PRICE** Free under GPL

HOW IT WORKS There's not much point in your system logging everything to your system log files if you never check them, but scrolling through thousands of lines of tedious errors doesn't make the job any easer. *KSystemLog* is a new graphical front-end that makes the task bearable. It can alert you to any problems as and when they happen, and you can sort messages into specific categories.

Pause the log, or watch the events in real time.

Filter events with the search field.



```
Date ▾             Host N  Process         Message
21/07/05 17:33...  linux   SuSEfirewall2   Warning: ip6tables does not support state matching. Extended IPv6 ...
21/07/05 17:33...  linux   SuSEfirewall2   Firewall rules set to CLOSE.
21/07/05 17:33...  linux   SuSEfirewall2   Firewall rules successfully set
21/07/05 17:33...  linux   SuSEfirewall2   Setting up rules from /etc/sysconfig/SuSEfirewall2 ...
21/07/05 17:33...  linux   SuSEfirewall2   Warning: ip6tables does not support state matching. Extended IPv6 ...
21/07/05 17:51...  linux   SuSEfirewall2   Firewall rules unloaded.
21/07/05 17:51...  linux   SuSEfirewall2   Warning: ip6tables does not support state matching. Extended IPv6 ...
21/07/05 17:51...  linux   SuSEfirewall2   Warning: ip6tables does not support state matching. Extended IPv6 ...
21/07/05 17:51...  linux   SuSEfirewall2   Firewall rules set to CLOSE.
21/07/05 17:51...  linux   SuSEfirewall2   Firewall rules successfully set
21/07/05 17:51...  linux   SuSEfirewall2   Setting up rules from /etc/sysconfig/SuSEfirewall2 ...
21/07/05 17:51...  linux   SuSEfirewall2   Warning: ip6tables does not support state matching. Extended IPv6 ...
```

Log events can be sorted and coloured.

Open Source Tripwire

Network auditer

- **DEVELOPERS** Tripwire Inc, Paul Herman
- **URL** http://sourceforge.net/projects/tripwire
- **PRICE** Free under GPL

HOW IT WORKS *Tripwire* creates a hash for each file on your system and stores it in a database. Each hash can only be generated from the bit-for-bit copy of the file, which means that any change, whether it's adding of text to a script or replacing one binary utility with another, will generate a different hash and be brought to *Tripwire*'s attention.

This is an effective way of combating the use of a rootkit. A rootkit contains replacement files for some core Linux utilities, enabling a hacker to track passwords or edit configuration files. *Tripwire* alerts the system administrator when such vital files have changed.

Installation is simple. You need to provide a pass phrase for signing the configuration files before you start, after which it's just a case of initialising and filling the database using the **tripwire --init** command. **tripwire --check** is used to test the integrity of the system.

Nmap

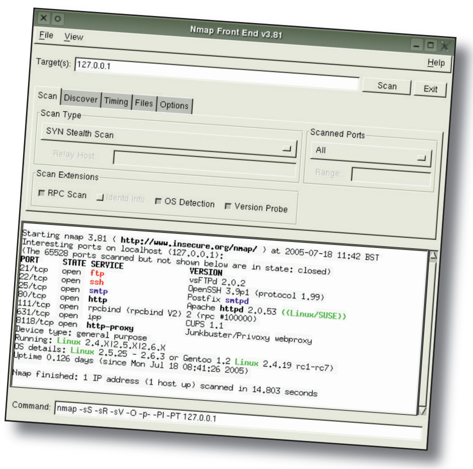
Network auditer

- **DEVELOPER** Fyodor
- **WEB** www.insecure.org/nmap
- **PRICE** Free under GPL

HOW IT WORKS: This is the tool you turn to when you want to test the security of your system. It's what *Nessus* uses, and in all likelihood it's what the hackers use as well. Though it was designed for trailing through large networks, it will work on single systems just as well. *Nmap* scans your computer for open ports, and by analysing network packets is often able to determine services and their versions running on your system, as well as the operating system itself.

There is a user interface, but *Nmap* is a command-line tool at heart, and various options allow you to scan a system with varying degrees of stealth.

For example, **nmap -v -sS -O 127.0.0.1** would scan your own computer using the TCP SYN scan, and output a list of open ports. The thing to be aware of is that not only can other computers tell they've been scanned using a tool like *Nmap*, it's also considered an aggressive act by many people. You should only use tools like *Nmap* to scan machines that you are responsible for.

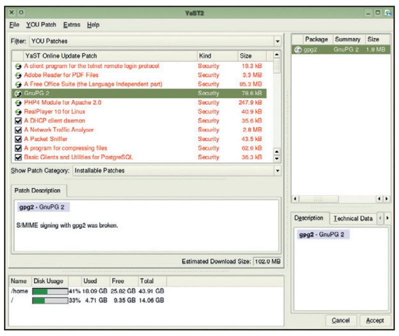


```
Starting nmap 3.81 ( http://www.insecure.org/nmap/ ) at 2005-07-18 11:42 BST
Interesting ports on localhost (127.0.0.1):
(The 65528 ports scanned but not shown below are in state: closed)
PORT      STATE SERVICE              VERSION
21/tcp    open  ftp                  vsFTPd 2.0.2
22/tcp    open  ssh                  OpenSSH 3.9p1 (protocol 1.99)
25/tcp    open  smtp                 Postfix smtpd
80/tcp    open  http                 Apache httpd 2.0.53 ((Linux/SUSE))
111/tcp   open  rpcbind (rpcbind V2) 2 (rpc #100000)
631/tcp   open  ipp                  CUPS 1.1
8118/tcp  open  http-proxy           Junkbuster/Privoxy webproxy
Device type: general purpose
Running: Linux 2.4.X|2.5.X|2.6.X
OS details: Linux 2.5.25 - 2.6.3 or Gentoo 1.2 Linux 2.4.19 rc1-rc7)
Uptime 0.126 days (since Mon Jul 18 08:41:26 2005)

Nmap finished: 1 IP address (1 host up) scanned in 14.803 seconds

Command: nmap -sS -sR -sV -O -p- -PI -PT 127.0.0.1
```

It's built for the console, but there's also an effective optional GUI for *Nmap*.

Security to-do list

You've learnt the theory, now put it into practice – here are the top nine things you should do now to keep your system safe...



YaST2

File YOU Patch Extras Help

Filter: YOU Patches

YaST Online Update Patch	Kind	Size
A client program for the telnet remote login protocol	Security	19.3 kB
Adobe Reader for PDF Files	Security	3.3 MB
A Free Office Suite (the Language Independent part)	Security	85.3 MB
GnuPG 2	Security	78.6 kB
PHP4 Module for Apache 2.0	Security	247.9 kB
RealPlayer 10 for Linux	Security	40.9 kB
☑ A DHCP client daemon	Security	35.6 kB
☑ A Network Traffic Analyser	Security	2.8 MB
☑ A Packet Sniffer	Security	43.5 kB
☑ A program for compressing files	Security	62.6 kB
☑ Basic Clients and Utilities for PostgreSQL	Security	36.3 kB

Package	Summary	Size
gpg2	GnuPG 2	1.8 MB

Show Patch Category: Installable Patches

Patch Description

gpg2 - GnuPG 2

S/MIME signing with gpg2 was broken.

Estimated Download Size: 102.0 MB

Name	Disk Usage	Used	Free	Total
/home	41%	18.09 GB	25.82 GB	43.91 GB
/	33%	4.71 GB	9.35 GB	14.06 GB

Description | Technical Data

gpg2 - GnuPG 2

Cancel Accept

***YaST* allows you to pick and choose which updates you wish, although you've no choice over the ones it considers critical fixes.**

1 Get your system patched

We can't stress this enough: your Linux system includes hundreds of programs that make a cohesive whole. If any one of them has security holes, you're at risk. The solution is as simple as using your package manager to update all the components at once. If you're using Mandriva, select Updates from Mandriva Control Center. SUSE users should use Online Update from *YaST*.

2 Hack your own passwords

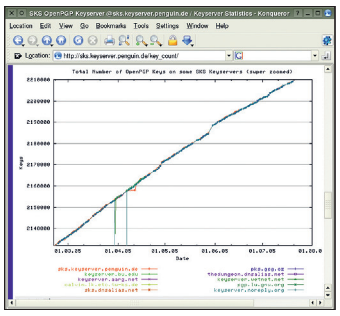
Try testing them to see how hard it would be for hackers. Follow our instructions on page 57 to test your system, keeping in mind that a hacker might have a much more powerful system than yours. We know it's annoying having a strong root password, but it's the last defence between your system and a dedicated hacker.

3 Test your firewall

Sure, it says it's protecting you, but how do you know your distro hasn't misconfigured it, or opened up holes for internet services? For the best test, you need to work from a computer outside your local network. If you have access to another machine, it's just a matter of running **nmap -sTUR -F -P0 -O hostname** as root. You could use an external web-driven tool, such as *ShieldsUP* (**www.grc.com/default.htm**), which reports on any open ports it finds.

4 Encrypt your emails

Every day people send information across the internet, totally open for anyone to copy. If you publish a public key, people can send you emails that only you can read. Type **gpg --gen-key** to create secret



More than 10,000 OpenPGP keys are submitted to the SKS Key servers each month.

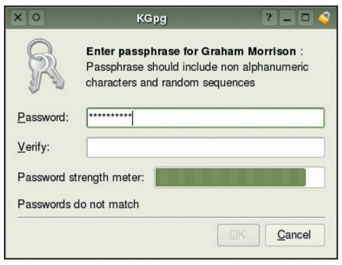
and private keys. You can export your public key to a file with **gpg --export -a -o pubkey.txt**. You then need to publish this key on a site where your friends can get it, such as **http://subkeys.pgp.net**. Import other people's public keys using **gpg --import <Filename>**.

5 Force a password on startup

Add 'password=secretword' to your **/etc/lilo.conf** file, then run **lilo** as root to make the change permanent. You can also – perhaps as an extra layer of security – add a password to your system BIOS.

6 Encrypt your Wi-Fi network

Although the standard WEP protocol is better than nothing, WPA uses dynamic keys to prevent intruders getting any footholds in your network. In a corporate environment, WPA uses a key server to provide different keys, whereas a home network generates a stream of different keys itself. At home, it's no harder to configure than WEP, requiring only a pass phrase, which itself provides a key.



KGpg

Enter passphrase for Graham Morrison :
Passphrase should include non alphanumeric characters and random sequences

Password: **********

Verify:

Password strength meter:

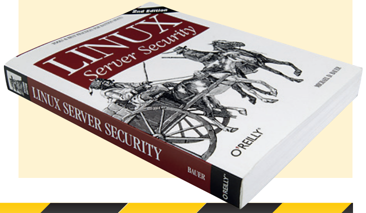
Passwords do not match

OK Cancel

Most wireless access points make switching from WEP to WPA easy.

THANKS...

Thanks to O'Reilly for letting us take some excellent advice from *Linux Server Security*, 2nd Edition, for use as our server tips. Thanks also to Mark Cox and Bruce Schneier for their help and contributions to the article.



7 Check your system log files

Security threats often appear here first. Log files live in your system's **/var/log** directory, and most system events are logged into the **Messages** file. For firewall problems, SUSE uses **firewall** to log any dropped packets, while Mandriva uses **syslog**. Display the last few lines of a log using a command such as **tail -f /var/log/syslog**.

8 Deny access to rogue IPs

Edit the **/etc/hosts.allow** and **/etc/hosts.deny** files on your server to grant or deny access from certain IP addresses to the services running on that server. Because **hosts.allow** is processed before **hosts.deny**, you need to be careful not to override instructions. To deny all connections except from those from explicitly declared, add **ALL : ALL** to the **hosts.deny** file, and **ALL : .yourhostnam.org** to **hosts.allow**.

9 Clear your shell history

If anyone does get access to your account, they can do untold damage using your private data. The shell history can give away login names and mistyped passwords, and give an indication of how you connect to external networks. Delete your *bash* history using the **history -c** command. Browser information can easily be deleted from within the browser itself. LXF

SECURITY BOOKSHELF

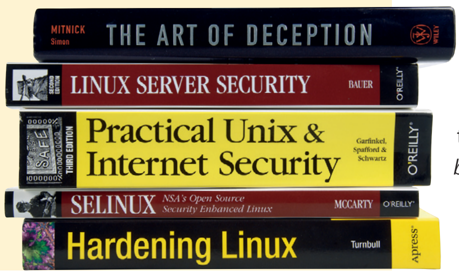
Further reading for those who want to know more...

Encryption
Applied Cryptography **(Schneier)**
Malicious Cryptography **(Young *et al*)**
Practical Cryptography **(Ferguson *et al*)**
SSH: The Definitive Guide **(Barrett *et al*)**

Internet
Building Internet Firewalls **(Zwicky *et al*)**
Managing Security with Snort **(Cox *et al*)**
Nessus, Snort, and Ethereal Power Tools **(Caswell *et al*)**
Practical UNIX & Internet Security, Third Edition **(Garfunkel *et al*)**

Servers
Hacking Linux Exposed **(Lee *et al*)**
Hardening Linux **(Turnbull)**
Linux Security Cookbook **(Barrett *et al*)**
Linux Server Security **(Bauer)**
SELinux **(McCarty)**

General
Beyond Fear **(Schneier)**
Maximum Linux Security **(Ray *et al*)**
The Art of Deception **(Mitnick)**
The Art of Intrusion **(Mitnick)**


MITNICK Simon THE ART OF DECEPTION WILEY
LINUX SERVER SECURITY BAUER O'REILLY
Practical Unix & Internet Security Garfinkel, Spafford & Schwartz O'REILLY
SELINUX NSA's Open Source Security Enhanced Linux MCCARTY O'REILLY
Hardening Linux Turnbull Apress