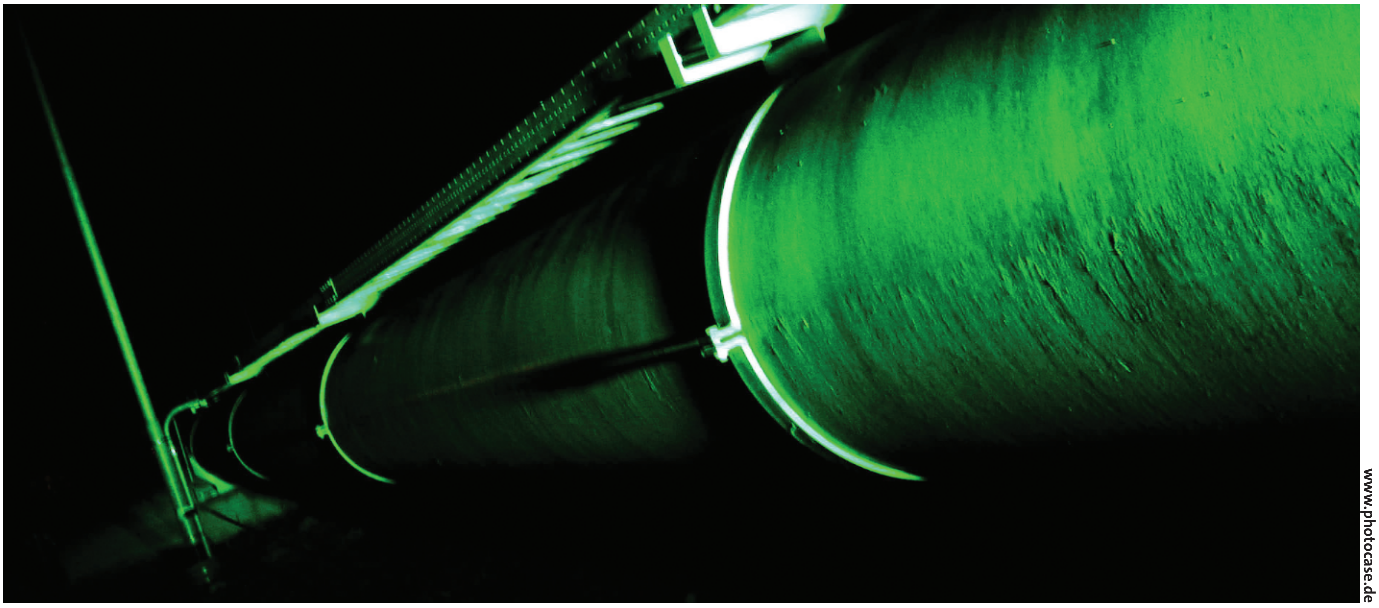
Restricting SSH Access with SCPonly

Safety Line



www.photocase.de

Most administrators are only too happy to replace the venerable FTP protocol with a secure alternative, such as SSH with SCP or SFTP. In its default setup, this program suite allows complete shell access to the system. Enter SCPonly, which adds the ability of confining users to a chroot jail. BY MARTIN WERTHMÖLLER

FTP can look back on thirty years of up and downloading files to and from servers. Amazingly, FTP is still the protocol of choice, although it is extremely insecure. The whole transfer process takes place in the clear, providing would-be attackers with a simple vector for grabbing passwords.

Also, the way FTP works complicates firewall configurations. The protocol needs two TCP connections. Depending on the mode, either the server or the client opens up a connection to an arbitrary target port on the other side of the connection. When the firewall is confronted with a connect request of this kind, it needs to know for security reasons that the request originated from a FTP connection.

FTPS is a secure and compatible alternative to FTP. FTPS is specified in RFC 2228 [1]; it uses two TCP connections, just like FTP, and implements data encryption in a similar way to HTTPS. Unfortunately, FTPS never made it past the RFC draft stage, and not many clients or servers actually implement the protocol.

No Alternatives in Sight

The WebDAV HTTP protocol extension (RFC 2518, [2]) allows files to be loaded and modified directly on a server. A TLS/SSL communication link can provide encryption for the data. Unfortunately, WebDAV has a few drawbacks. On the Apache Web server, the module does not support user ID changes. In other words, all WebDAV users have the same user ID on the filesystem. Thus, you can rule out granular permissions with Apache and WebDAV. Most production Web servers do not have a WebDAV module installed, or the module is not configured.

Safe Replacement

On Unix systems, the SSH package tools, SCP (secure copy) and SFTP typically provide an alternative to FTP. SCP provides encrypted transport for files via a SSH tunnel. Users simply specify the files to be copied on the command line. The SFTP protocol is not to be confused with FTPS; the two are not related. SFTP only needs a single TCP connection, as it uses SSH, just like SCP. However, in contrast to SCP, you can use it interactively, just like an FTP client.

These two variants do not require extensive configuration of the server or client. Many machines already have a SSH server set up, and most firewalls allow SSH connections to pass. Unfortunately, this secure approach also has a downside:

- Not all operating systems provide a GUI-based SCP client software solution.
- The SCP and SFTP programs in the (Open)SSH packages are too complicated for many users to cope with in their daily work patterns.

- SCP requires an interactive SSH session. This means that SCP users are also allowed to execute shell commands.
- The SSH daemon does not have native support for a chroot configuration, in contrast to many FTP servers.

OpenSSH development is mainly geared toward supporting OpenBSD, although there are ports to many other systems, including Linux, Cygwin, and Mac OS X. Thanks to the Cygwin port, you can even install OpenSSH on Windows. However, Putty [3] is a better choice for Windows. Besides the OpenSSH client programs, *scp* and *sftp*, GUI-based front-ends are available for Linux, Windows, and Mac OS X (see box "GUI-based SCP clients"). These GUI clients often have a free license.

There are numerous scenarios where users do not need, or should not be allowed a login shell. Large Web space providers host thousands of websites on a single server, for example. Full shell access could be fatal and a system of this kind is also quite complex to maintain.

Restricting Access

As SCP and SFTP assume an interactive shell, a useful solution will need to restrict shell access, while at the same time allowing full SCP and SFTP functionality.

There are two possible approaches to this: a customized login shell could allow users to run only the executables needed for SCP and SFTP operations. The second approach would be to confine users to a chroot jail when they log on, and only allowing them access to the required programs within the jail. This would then help security by preventing the clients from accessing other people's files.

Switching to a chroot jail in the login shell is quite dangerous, as users could interrupt the process before the chroot command has executed, and run arbitrary commands in an unsecured environment. Thus, the login shell needs to ensure that the chroot command runs before the user can do anything about it. There is a patch at [8] that tells the SSH daemon to call *chroot()*.

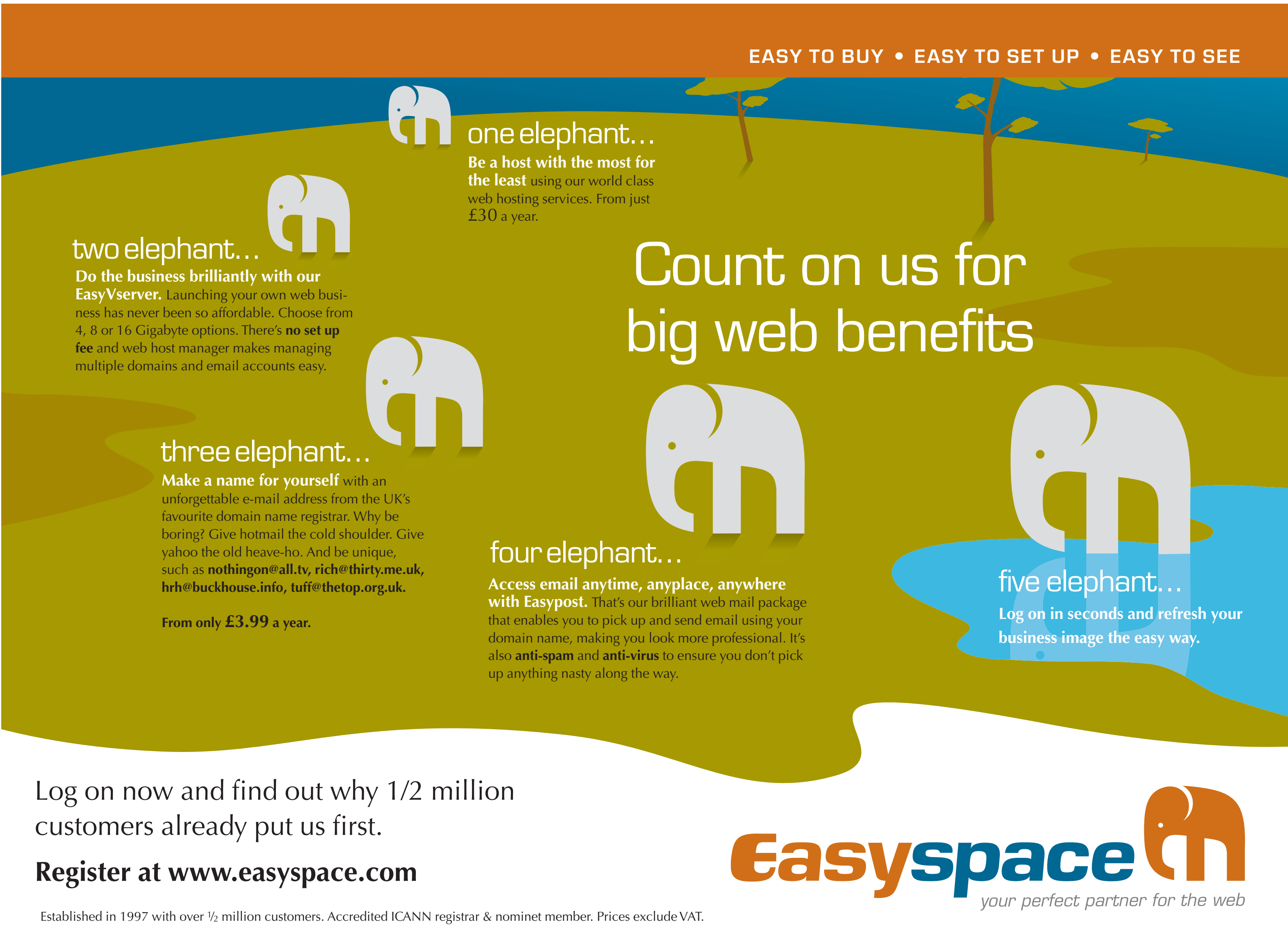
Best Option: SCPonly

A patched SSH daemon is not available as a RPM or Deb package, and thus will not be recognized by your distributor's security updates. This means manually modifying the OpenSSH installation on your server each time you perform a security update.

RSSH [9] and SCPonly [10] offer the best of both worlds. They lock the user in a chroot jail, and allow access to a restricted command set only. One advantage that SCPonly has over RSSH is the fact that it is compatible to the popular WinSCP client. Also, you can use SCPonly to run rsync via SSH, and future versions promise CVS over SSH support.

At present, only Gentoo provides a SCPonly package with the current distribution. The Debian developers do have a package in unstable and testing, however. Until they do, users have no alternative but to download the sources and build the program. A download is available from [10].

SCPonly's author develops and runs the program package on FreeBSD, which



has a different approach to user and password management than Linux. Older versions may require you to perform some manual modification of the setup scripts, particularly if you need automatic chroot jail configuration. This is no longer necessary in the current version 3.11.

The *groups* command, which Linux often implements as a shell script, can be another pitfall. The setup script, *setup_chroot.sh*, copies the */usr/bin/groups* file to the chroot jail, among other things. When you call *groups*, the kernel attempts to hand the script to the shell, */bin/sh*.

To avoid installing a full-featured command line interpreter in the jail, sysadmins have little alternative but to replace */usr/bin/groups* in the chroot environment with the command provided by the SCPonly distribution. Some GUI-based SCP clients can be a source of headaches in this respect. Check out the tip in "GUI-based SCP clients" box below.

Installation

After unpacking the SCPonly sources, you need to run the *configure* script with a few options, rsync or chroot functionality, for example. Type *./configure --help* to output a complete list of options. The following sample syntax, enables rsync compatibility, creates a chroot binary, and installs whatever is needed:

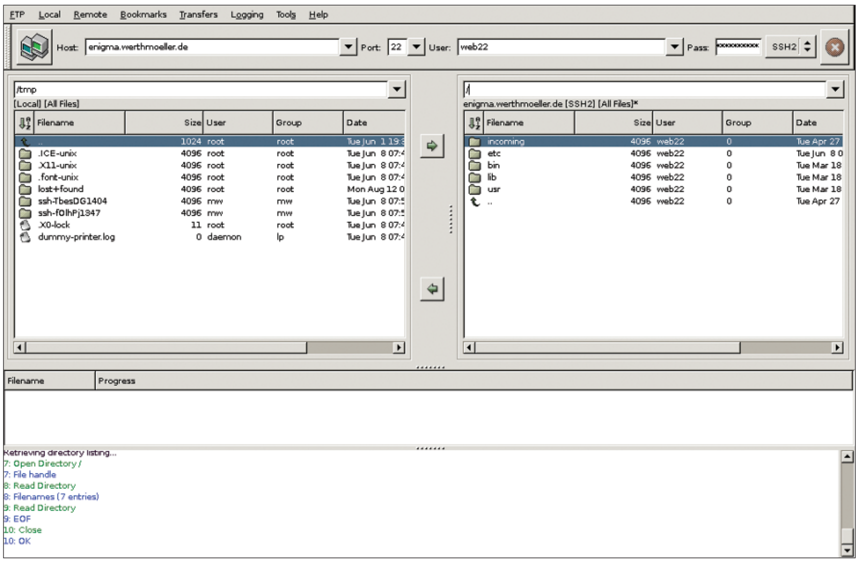
```
./configure -enable-rsync➥
-compat --enable-chrooted-binary
make
make install
```

GUI-based SCP clients

Numerous GUI-based SSH clients are available for Linux, Windows, and Mac OS X. Some of these programs, like gFTP [4] (see Figure 1), are basically FTP clients that can also handle SCP and SFTP access. Others are genuine SCP programs, such as WinSCP [5] for Microsoft Windows. The front-ends are quite similar. All of these applications have a two-panel main window (with one panel for the local machine, and another for the remote filesystem tree) where users can use drag & drop files to copy files back and forth.

In most cases, users can right click a file to set that file's attributes. The context menu often has additional features such as bookmark and password management. WinSCP users should go for the current Version 3, as the known bugs in previous versions have been removed in Version 3. You can avoid error messages telling you that the *groups* command is missing on the server by disabling the *lookup user groups* option in WinSCP. As an alternative to WinSCP, you might also like to try the Windows client, Filezilla [6].

Things are a lot easier for KDE users wanting to use SCP. Simply type *fish://server* in the Konqueror address box, to copy files just as if they were on your local disk. Mac OS X has a program called Fugu [7]. Fugu is available as a source code distribution, and supports FTP, SFTP, and SCP.



Host: enigma.werthmoeller.de Port: 22 User: web22 Pass: xxxxxxxxxx

```
/tmp
[Local] [All Files]
Filename                 Size User    Group  Date
..                       1024 root    root   Tue Jun  1 19:3
.ICE-unix                4096 root    root   Tue Jun  8 07:4
.X11-unix                4096 root    root   Tue Jun  8 07:4
.font-unix               4096 root    root   Tue Jun  8 07:4
lost+found               4096 root    root   Mon Aug 12 0
ssh-TbesDG1404           4096 mw      mw     Tue Jun  8 07:5
ssh-f0lhPj1347           4096 mw      mw     Tue Jun  8 07:5
.X0-lock                   11 root    root   Tue Jun  8 07:4
dummy-printer.log           0 daemon  lp     Tue Jun  8 07:4
```

```
/
enigma.werthmoeller.de [SSH2] [All Files]*
Filename                 Size User    Group  Date
incoming                 4096 web22   0      Tue Apr 27
etc                      4096 web22   0      Tue Jun  8 0
bin                      4096 web22   0      Tue Mar 18
lib                      4096 web22   0      Tue Mar 18
usr                      4096 web22   0      Tue Mar 18
..                       4096 web22   0      Tue Apr 27
```

Filename Progress

```
Retrieving directory listing...
7: Open Directory /
7: File handle
8: Read Directory
8: Filenames (7 entries)
9: Read Directory
9: EOF
10: Close
10: OK
```

Figure 1: A SCP session with the GUI-based gFTP program. As SCPonly is running on the remote host, the user automatically lands in a chroot jail.

To set up a chroot jail, admins need to collect the required libraries, and store them at the correct position in the chroot directory tree. SCPonly provides *setup_chroot.sh*, which we referred to earlier, to take care of this. The execute bit is not set for the script by default. This is easy to change: type *make jail* instead of *make install* to resolve the problem and initiate the steps required for running the script.

The setup script needs to run with root privileges, so you will want to be root when you run *configure*, as this creates the *setup_chroot.sh* file from *setup_chroot.sh.in;* it needs a few files in privileged paths, such as */usr/sbin/*. When you create and run *setup_chroot.sh*, you are first prompted for the home directory and username for the new SCPonly user. The user will be placed in a chroot jail after logging on via SFTP. The user has write permissions in *~/incoming* (or another directory assigned by the admin).

Usability

To make things easier for your users, you can assign */home/user/incoming* as the SCPonly user's home directory. This tells SCPonly to change directory to *~/incoming* immediately after logging on. SCPonly provides server operators with a tool that could finally oust the venerable FTP protocol, and clear the way for more modern authentication and encryption techniques. ■

INFO

[1] FTPS RFC: *http://www.ietf.org/rfc/rfc2228.txt*

[2] WebDAV RFC: *http://www.ietf.org/rfc/rfc2518.txt*

[3] Putty: *http://www.chiark.greenend.org.uk/~sgtatham/putty/*

[4] gFTP: *http://www.gftp.org*

[5] WinSCP: *http://winscp.sourceforge.net/eng/*

[6] Filezilla: *http://filezilla.sourceforge.net*

[7] Fugu: *http://rsug.itd.umich.edu/software/fugu*

[8] OpenSSH-Chroot-Patch: *http://chrootssh.sourceforge.net/index.php*

[9] RSSH: *http://www.pizzashack.org/rssh/index.shtml*

[10] SCPonly: *http://sublimation.org/scponly/*