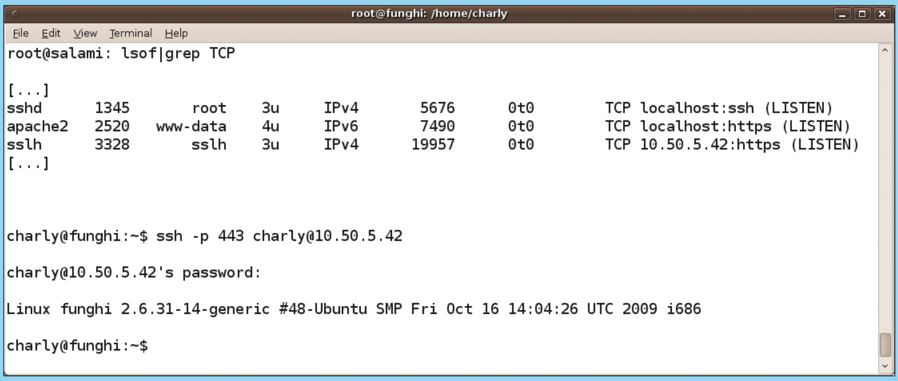
The sys admin's daily grind: sslh

THE DAEMON'S IN THE DETAILS

Some of Charly's servers run the SSH daemon on port 443 rather than on the standard port 22. If an SSL-capable Apache web server starts causing trouble, his method of settling the dispute is sslh.

BY CHARLY KÜHNAST



```
root@salami: lsof|grep TCP

[...]
sshd      1345       root    3u    IPv4      5676     0t0       TCP localhost:ssh (LISTEN)
apache2   2520   www-data    4u    IPv6      7490     0t0       TCP localhost:https (LISTEN)
sslh      3328       sslh    3u    IPv4     19957     0t0       TCP 10.50.5.42:https (LISTEN)
[...]


charly@funghi:~$ ssh -p 443 charly@10.50.5.42

charly@10.50.5.42's password:

Linux funghi 2.6.31-14-generic #48-Ubuntu SMP Fri Oct 16 14:04:26 UTC 2009 i686

charly@funghi:~$
```

Figure 1: Instead of Apache and SSH fighting over port 443, the sslh daemon upstream identifies the request type - SSH in this case - and passes it on to the daemon responsible for it.

Whether I happen to be in an Internet café, using the wireless LAN at a hotel, or using a public hotspot at an airport, I continually find myself locked up behind a firewall that refuses connections to target port 22. Of course, any firewall will generously let traffic to ports 80 and 443 pass.

In other words, it's a good idea to bind the SSH daemon to the HTTPS port on my servers. This saves me digging a tunnel, and I can simply log in to my server with *ssh -p 443 <user>@<host>*. But if the HTTPS port is already occupied by an SSL-capable web server, I have to put my thinking cap back on. Enter *sslh* [1].

The makers of this tool call it an SSL/SSH multiplexer. The underlying idea is that the multiplexer listens on port 443 and discovers for incoming connections whether the client wants to speak HTTPS with SSH to the host. The services themselves are bound to localhost:443 and localhost:22, respectively (Figure 1). Sslh retrieves this information from the */etc/defaults/sslh* file, which looks like the following in a simple setup:

```
RUN=yes
DAEMON_OPTS="-u sslh ⮐
  -p 10.50.5.42:443 ⮐
  -s 127.0.0.1:22 ⮐
  -l 127.0.0.1:443 ⮐
  -P /var/run/sslh.pid"
```

To find out which protocol is currently required, sshl analyzes the client's behavior. In the case of an incoming HTTPS connection, the client waits for the server to signal that it is ready to receive. A client requesting an SSH connection will not wait but will open the dialog itself, and sslh will just wait for a short time, typically two seconds. If the client does not send any data in this time, sslh assumes that it is an HTTPS connection and forwards it to the web server at 127.0.0.1:443.

Apache's Domain

To restrict the Apache server to localhost, I have to change the list parameter in its configuration that sets SSL to *443* by default to *127.0.0.1:443*. Strictly speaking, you do not necessarily have to bind the SSH port to localhost because there is no conflict with another daemon on port 22. However, I did this for two reasons: First, it protects me and my *auth.log* from a whole bunch of scans that keep on turning up at that address. Second, I can reach the server via a serial console if SSH or sslh should fail for some reason. ■

INFO

[1] sslh: *http://www.rutschle.net/tech/sslh*

THE AUTHOR

Charly Kühnast is a Unix operating system administrator at the Data Center in Moers, Germany. His tasks include firewall and DMZ security and availability. He divides his leisure time into hot, wet, and eastern sectors, where he enjoys cooking, fresh water aquariums, and learning Japanese, respectively.