The Sysadmin's Daily Grind: Portfwd

NEW DIRECTIONS

Incoming TCP connections do not always end up where they are supposed to. A freely configurable redirector points digital debris in the direction of a new and better place.

BY CHARLY KÜHNAST

Just a little while back we discussed Rinetd [1], a TCP redirector. Rinetd is lean, reliable, and easy to configure; on the downside, it lacks some advanced features. This month I'll examine Portfwd, a tool that includes some of those features missing from Rinetd. Portfwd (Port Forwarding Daemon, [2]) takes the form of a 116K tarball, which you can build and install in the normal way:

```
./configure; make; make install
```

or almost. The all-important binary went into hiding in the *src* directory after I typed *make install*. A symlink *(ln -s /usr/local/portfwd-0.27/src/portfwd/usr/bin/portfwd)* took care of that.

No Sweat

Let's first check how Portfwd handles simple redirection. We want to forward any TCP packets that arrive via port 80 to the server at 10.20.30.40. The con figuration file looks like this:

```
user nobody
group nobody
tcp { 80 { => 10.20.30.40:80 } }
```

The external curly braces contain the port where Portfwd receives the incoming connection, and the internal braces take care of the redirection target. Of course, good old Rinetd could have handled this too. But, I found I was able to configure multiple targets to use Portfwd as a simple round-robin load balancer. I changed the first example to forward incoming connections on port 80 to two servers, 10.20.30.40 and 10.20.30.41, like this:

```
tcp { 80 { =>
10.20.30.40:80,

10.20.30.41:80 } }
```

If my machine is multi-homed, I can use *bind-adress* to specify the interface I mean. For my third experiment, I assigned the addresses 192.168.1.1 and 192.168.1.2 to my machine. I want to beam any incoming connections for port 25 to two other mail servers:

```
bind-address 192.168.1.1
tcp { 25 { => 10.20.30.40:25 } }

bind-address 192.168.1.2
tcp { 25 { => 10.20.30.41:25 } }
```

In contrast to Rinetd, Portfwd not only forwards TCP connections but also UDP datagrams. The following rule tells Portfwd to accept DNS requests and forward those requests to the server at *dns.example.com*:

```
udp { 53 { => ↩
dns.example.com:53 } }
```

Fragile

Portfwd also gives me the ability to define a special case for servers with connections that tend to break quite often. The *fragile* keyword tells Portfwd to be lenient in case of connection errors – and to retry more often than usual:

```
fragile tcp { 80 { => ↩
dialup.example.com:8080 } }
```

This is the kind of setup I like to use when I want a central machine to look like a function-loaded monster from the outside, although I actually offload much of this functionality onto separate machines as needs dictate. ■

SYSADMIN

INFO

[1] Charly Kühnast, "The Sysadmin's Daily Grind: Rinetd": Linux Magazine 11/02, p. 51.
[2] Portfwd: *http://portfwd.sf.net*

THE AUTHOR

Charly Kühnast is a Unix System Manager at the data-center in Moers, near Germany's famous River Rhine. His tasks include ensuring firewall security and availability and taking care of the DMZ (demilitarized zone).