rinetd - a TCP port redirector

by Lenz Grimmer 30 June, 1999

rinetd is a simple, but effective tool for redirecting TCP connections. This article describes its basic features and how it can be used for Linux firewalls.

Contents

1 Introduction

Linux has become very popular as the "swiss army knife" for all kinds of networking purposes. One area where Linux is a viable alternative to commercial solutions is firewalling. The Linux kernel offers some very flexible packet filtering features, which enable you to setup highly effective firewalling rules. However, it is sometimes necessary to open a defined hole to the inside of your net, for example if you need to connect to an internal mail- or WWW-Server from outside your firewalling host. This is the part where rinetd kicks in. This program is used to efficiently redirect connections from one IP address/port combination to another.

From the manual page: "Rinetd is a single-process server which handles any number of connections to the address/port pairs specified in the configuration file `etc/rinetd.conf`. Since rinetd runs as a single process using nonblocking I/O, it is able to redirect a large number of connections without putting much additional load to the machine." Apart from Linux, rinetd should run on any major Unix flavor without any trouble. But since the tool is free software, I would like to recommend using a free operating system as well :^).

It is useful when you are operating virtual servers, firewalls and the like. However, there's one catch: rinetd is not able to redirect FTP connections, because FTP requires more than one socket. Moreover, rinetd can only redirect TCP connections, there's no support for UDP.

Rinetd is Open Source under the terms of the GNU General Public License (see *http://www.gnu.org* for further info). The author is Thomas Boutell, a famous open source developer. He also created the "gd" library for manipulation of GIF images and several other nice programs. See his WWW-Site *http://www.boutell.com* for more information about his work.

2 Installation

There are two methods on how to install rinetd on your system. You could either download the source and compile rinetd on your own, or you can use a precompiled binary package for your linux distribution.

2.1 Installation from source

Download via ftp from Tom Boutell's ftp-Server: *ftp://ftp.boutell.com/pub/boutell/rinetd/rinetd.tar.gz* Surprisingly, there is also a 32-bit Windows Version (95/98/NT) available under *ftp://ftp.boutell.com/pub/boutell/rinetd/rinetd.zip*, but this won't be covered in this article. The windows port has been added recently, in former times rinetd used to run on Unices only.

Extract the source archive using `tar` and `gzip`. You should first check the Makefile for platform-specific details. Type "make" to start the compilation afterwards. This should invoke the compiler and create the binary for your platform. To install the compiled binary, type "make install" as user "root". If you want to start up the process at boot time, you should create a separate init script or add rinetd to an already existing startup script.

2.2 Binary packages

Precompiled packages are available for most Linux distributions. For example, SuSE Linux has a preconfigured package, which can be directly installed from the CD. You just have to set "**START_RINETD**" in `/etc/rc.config` to "**yes**" after installing the package via the SuSE configuration tool "YaST".

Before starting rinetd, you have to create a configuration file `/etc/rinetd.conf`. There's a well commented sample config file in the source package, I would recommend to use this for starters. The manual (man 8 rinetd) page covers all the possible configuration options.

3 Configuration examples

The configuration file `/etc/rinetd.conf` contains rules, that define how and if rinetd forwards incoming TCP connections. You need to restart rinetd, if you make changes to this file while rinetd is running. The kill -1 signal (SIGHUP) can be used to cause rinetd to reload its configuration file without interrupting existing connections.

3.1 Forwarding rules

These rules define the main functionality of rinetd: forwarding connections from one address/port-pair to another. The format is "bindaddress bindport connectaddress connectport" For example,

```
192.168.0.1 80 10.10.0.4 80
```

would forward all requests on IP address 192.168.0.1 (which could be an external network interface) on port 80 (www) to an internal WWW-Server (10.10.0.4 in this example) with no direct route to the other net. Instead of port numbers, you can also use service names as defined in `/etc/services`. Therefore, the above mentioned example could also be written like this:

```
192.168.0.1 www 10.10.0.4 www
```

In addition to that, host names instead of IP addresses are also allowed, if you have a working DNS setup. The special IP address `0.0.0.0` allows rinetd to respond on all IP addresses that the server has been configured for (The Linux kernel allows you to set up more than one IP address for an interface, this is called "IP aliasing"). The forwarding can either take place on one interface, or you can use two separate network cards. This depends on the desired functionality.

3.2 ALLOW and DENY rules

By using ALLOW and DENY rules, you can effectively define access control to certain rules. Allow and deny rules that appear before the first forwarding rule are applied globally. If the address of a new connection matches one of the DENY rules or does not match any of the ALLOW rules, the connection is immediately rejected.

If ALLOW or DENY rules appear after a certain forwarding rule, they only apply to this specific rule.

By using patterns, you can easily match a broader range of IP addresses. Patterns can contain the following characters: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, . (period), ?, and *. The "?" wildcard matches any single character. The "*" wildcard matches any number of characters, including zero. For example, the line

```
deny 141.73.21.*
```

matches all IP addresses in the 141.73.21 class C domain and would deny certain forwarding rules to take action.

Host names are not permitted in allow and deny rules, since host name lookups would dramatically decrease the performance. Because rinetd is a single process server, all other connections would be forced to pause during the address lookup. However, this should not be a serious shortcoming.

3.3 LOGGING

It is possible to create a log file about rinetd's forwarding activity. Moreover, it is able to produce a log file in two different formats: tab-delimited and web server-style "common log format." The latter is easier to parse by regular log-statistic tools.

By default, rinetd logs in a simple tab-delimited format containing the following information:

- Date and time
- Client address
- Listening host
- Listening port
- Forwarded-to host
- Forwarded-to port
- Bytes received from client
- Bytes sent to client
- Result message

The following is a sample of rinetd's output:

```
30/Jun/1999:22:21:03    197.10.0.1    197.10.0.5    80    10.0.0.3    8080    0      0    not-allowed
30/Jun/1999:22:27:56    197.10.0.1    197.10.0.5    80    10.0.0.3    8080    262    0    done-remote-closed
```

If you prefer a web server-stylish "common log file format", you can add the following keyword to the configuration file:

```
logcommon
```

This would produce the following output:

```
197.10.0.1 - - [30/Jun/1999:22:32:14  +0200] "GET /rinetd-services/197.10.0.5/80/10.0.0.3/80/done-local-closed HTTP/1.0" 200 1207 - - - 2
197.10.0.1 - - [30/Jun/1999:22:32:38  +0200] "GET /rinetd-services/197.10.0.5/80/10.0.0.3/80/done-local-closed HTTP/1.0" 200 1207 - - - 2
```

By default, rinetd does not produce a log file. To activate logging, you need to add the following line to `/etc/rinetd.conf`:

```
logfile <filename>
```

Please take into account, that writing log files does also require resources on the host. Especially on a heavy-loaded server, the log file can grow significantly and the logging can slow down the forwarding of connections. You should also consider an automated log-rotating mechanism for easier maintenance.

4 Example

Now let's go for a real-world example. A customer is running a firewall, which is supposed to protect his LAN from the outside world. It has an official IP address 131.23.2.4, which is assigned to one of two NICs. The internal network is using the IP address range 10.10.0.0/16. The customer is running a Web-Server in his internal network (10.10.0.8, Port 8080) and a mail server (10.10.0.2), where the employees can read their mail on the host by logging into the server via the secure shell (SSH). The customer would now like to give access to his WWW-server to a partner company, that is also connected to the Internet using the class C IP network 141.73.4.0/24. In addition to that, he wants to enable his employees to read mail while they are away from their offices. To make this connection a bit more "secure", the connect should take place on the usual telnet port (adding some obscurity does not help a lot, but it may confuse the unexperienced hacker). He also wishes to have a log file about the ongoing connection in the common log file format.

Here's the respective configuration file `/etc/rinetd.conf`:

```
# rinetd configuration file (/etc/rinetd.conf)
# Allow http connects for all hosts from the partner company
131.23.2.4 www 10.10.0.8 8080
allow 141.73.4.*
# Allow ssh connects on telnet port
131.23.2.4 telnet 10.10.0.2 ssh
# Log the connections
logfile /var/log/rinetd.log
logcommon
```

This solution has one shortcoming: the internal web server does not recognize, which external host initiated the http connection, it only "sees" the firewall's internal IP address. If you need to have that information, you have to match the web server logfiles with the rinetd logfile on the firewall.

Please note, that this is just a simple example to explain the basic functionality of rinetd. It's really up to you and your imagination, how you can effectively utilize this helpful daemon for your purposes. It's possibilities are virtually unlimited!

5 About the author

Lenz Grimmer uses Linux since the 0.99.xx kernel era. After finishing his studies in computer science, he worked as a system administrator for an internet consulting agency, where he learned even more about his favourite operating system. Since April 1998, he works for SuSE GmbH in Nuremberg, Germany, Europe's biggest Linux distributor. Apart from his daily work as a supporter and developer, he is the maintainer of the SuSE FAQ (*http://www.suse.com/Support/Doku/FAQ/*) and is also responsible for some packages of the distribution (rinetd is one of them). You can reach him via E-Mail at *Lenz.Grimmer@suse.de*, or just follow the SuSE Linux mailinglist `suse-linux-e@suse.com`, where he tries to be present, as his time permits. To subscribe to this list, have a look at *http://www.suse.com/Mailinglists/index.html*.