FreeBSD's Firewall Feast

By Michael W Lucas

FreeBSD is famous for all sorts of fantastic features, such as ZFS, jails, bhyve virtualization, and the Ports Collection. It's somewhat infamous, however, for having three different firewalls: PF, IPFilter, and IPFW. Where did all these firewalls come from, and why are they all still in the system?

The IT industry has repeatedly abused, stretched, and tormented the word firewall to fit all sorts of different products. When someone uses firewall, ask them exactly what they're talking about. Do they mean a caching HTTP proxy like Squid or Varnish? A generic proxy like relayd? Or a TCP/IP packet filter?

All of FreeBSD's firewalls are packet filters. They control which TCP/IP addresses and ports can connect to the host. If your FreeBSD host passes packets between interfaces, then the firewall controls which traffic gets forwarded, which gets silently dropped, and which is sent back to the source with a letter of complaint. A packet-forwarding packet filter is the original firewall.

The firewalls all have a common core feature set considered the minimum for a modern packet filter. They can track the state of a TCP/IP connection and permit traffic based on existing connections. They can all return resets or silently drop connections. All can manage non-routable addresses and perform network address translation. They all work with lists of rules defining how to respond to traffic from different IP addresses and network ports. Incoming packets are compared to the list of rules until they are permitted or rejected.

The firewalls have their own unique features, however. How do you choose between them?

IPFW is the oldest FreeBSD packet filter, dating from 1995. IPFW maintains its rules in a

numbered list. You add a rule to an existing ruleset by giving it a rule number. Rule numbers range from 1 to 65534. IPFW rules are very dynamic, and can be altered programmatically. Numbered rules do impose some limits, however. Every time I design an IPFW ruleset, I eventually wind up having to renumber the rules because, somehow, I need to cram too many rules in between two other rules. This is a limitation of my imagination when creating my ruleset, though, not any limitation in IPFW.

The most interesting unique feature in IPFW is dummynet, which lets you selectively degrade traffic. Why would you want to make a connection worse? Well, for one thing, it's kind of fun to simulate an ADSL link to the moon. But the ability to impose excess latency and bandwidth limitations on a connection can save a global organization days or weeks of time and thousands of dollars in developer travel expenses. More than once, I've set up an IPFW box to simulate a connection from the other side of the world, or with multiple profiles to simulate several different countries. A user in South Korea with a gigabit Internet line will have a very different experience than a user who's closer but has less bandwidth. Dummynet lets you simulate their experience locally.

IPFW also has the highest performance of any FreeBSD firewall, although that only becomes apparent at tens of gigabits per second.

IPFilter, or IPF, is a platform-independent firewall, and came to FreeBSD in the late 1990s. You can use IPFilter on FreeBSD, Solaris, SunOS, HP-UX, NetBSD, OpenBSD, and Linux. It has a configuration syntax similar to PF. If you must run one firewall across multiple operating systems, IPFilter is for you.

IPFilter is not in active development, but it can block, permit, and translate packets. It's largely feature-complete.

PF is the newest FreeBSD firewall, and is quite popular among younger sysadmins. It originated with the OpenBSD project, as their replacement for IPFilter. It's quite popular among younger sysadmins who didn't grow up with IPFW.

PF is perhaps the most popular FreeBSD firewall. PF has a simpler configuration syntax than IPFW. While you can alter the ruleset programmatically, you must configure the ruleset to permit such alterations. While mailing list archives discuss occasional problems, such as handling IPv6 fragmentation, those issues were solved years ago.

While FreeBSD's PF originated with OpenBSD, that import happened several years ago. FreeBSD's PF has diverged from the original import, and OpenBSD's PF has continued its natural evolution. Chances are that the two will continue to diverge.

All this information is nice, but which should **you** choose?

When you're studying PF configuration, be sure that you use documentation relevant for FreeBSD, not current OpenBSD documentation. The chances of FreeBSD taking a new import of OpenBSD's PF are very slim. PF is also incompatible with the vimage virtual networking stack used by jails.

All this information is nice, but which should you choose? That depends on your environment. Most sysadmins with small servers will find PF's simplicity a win.

If you want one firewall software across multiple Unix-like operating systems, use IPFilter.

If you need to pass tens of gigabits per second, simulate a bad connection, have unlimited ability to programmatically alter rules, or require advanced jail networking, use IPFW.

If you already have one deployed, of course, keep using it. They all work fine for the essential firewall task of filtering packets. Combined with an application proxy or cache such as Squid, relayd, or varnish, FreeBSD can go head-to-head with any commercial firewall vendor and win. ●

MICHAEL W LUCAS is the author of several books on FreeBSD, including *Absolute FreeBSD* and the *FreeBSD Mastery series*. Learn more at *www.michaelwlucas.com*.