Using pf on OS X Mountain Lion

15 May 2013 · Filed in [Tutorial](https://blog.scottlowe.org/categories/tutorial)

I’ve [written before](https://blog.scottlowe.org/2012/04/05/setting-up-ipfw-on-mac-os-x/) about adding an extra layer of network security to your Macintosh by leveraging the BSD-level ipfw firewall, in addition to the standard GUI firewall and additional third-party firewalls (like [Little Snitch](http://www.obdev.at/products/littlesnitch/index.html)). In OS X Lion and OS X Mountain Lion, though, ipfw was deprecated in favor of pf, the powerful packet filter that I believe originated on OpenBSD. (OS X’s version of pf is ported from FreeBSD.) In this article, I’m going to show you how to use pf on OS X.

Note that this is just *one way* of leveraging pf, not necessarily the *only way* of doing it. I tested (and am currently using) this configuration on OS X Mountain Lion 10.8.3.

There are X basic pieces involved in getting pf up and running on OS X Mountain Lion:

1. Putting pf configuration files in place.
2. Creating a launchd item for pf.

Let’s look at each of these pieces in a bit more detail. We’ll start with the configuration files.

Putting Configuration Files in Place

OS X Mountain Lion comes with a barebones /etc/pf.conf preinstalled. This barebones configuration file references a single anchor, found in /etc/pf.anchors/com.apple. This anchor, however, does not contain any actual pf rules; instead, it appears to be nothing more than a placeholder.

Since there is a configuration file already in place, you have two options ahead of you:

1. You can overwrite the existing configuration file. The drawback of this approach is that a) Apple has been known to change this file during system updates, undoing your changes; and b) it could break future OS X functionality.
2. You can bypass the existing configuration file. This is the approach I took, partly due to the reasons listed above and partly because I found that pfctl (the program used to manage pf) wouldn’t activate the filter rules when the existing configuration file was used. (It complained about improper order of lines in the existing configuration file.)

Note that some tools (like [IceFloor](http://www.hanynet.com/icefloor/index.html)) take the first approach and modify the existing configuration file.

I’ll assume you’re going to use option #2. What you’ll need, then, are (at a minimum) two configuration files:

1. The pf configuration file you want it to parse on startup
2. At least one anchor file that contains the various options and rules you want to pass to pf when it starts

Since we’re bypassing the existing configuration file, all you really need is an extremely simple configuration file that points to your anchor and loads it, like this:

anchor "org.scottlowe.pf"

load anchor "org.scottlowe.pf" from "/etc/pf.anchors/org.scottlowe.pf.rules"

The other file you need has the actual options and rules that will be passed to pf when it starts. You can get fancy here and use a separate file to define macros and tables, or you can bundle the macros and tables in with the rules. Whatever approach you take, be **sure** that you have the commands in this file in the right order: options, normalization, queueing, translation, and filtering. Failure to put things in the right order will cause pf not to enable and will leave your system without this additional layer of network protection.

A *very* simple set of rules in an anchor might look something like this (click [here](https://gist.github.com/scottslowe/5581710) for an option to download this code snippet):

# Options

set block-policy drop

set fingerprints "/etc/pf.os"

set ruleset-optimization basic

set skip on lo0

# Normalization

# Scrub incoming packets

scrub in all no-df

# Queueing

# Translation

# Filtering

# Antispoof

antispoof log quick for { lo0 en0 en2 }

# Block by default

block in log

# Block to/from illegal destinations or sources

block in log quick from no-route to any

# Allow critical system traffic

pass in quick inet proto udp from any port 67 to any port 68

# Allow ICMP from home LAN

pass in log proto icmp from 192.168.254.0/24

# Allow outgoing traffic

pass out inet proto tcp from any to any keep state

pass out inet proto udp from any to any keep state

Naturally, you’d want to customize these rules to fit your environment. At the end of this article I provide some additional resources that might help with this task.

Once you have the configuration file in place and at least one anchor defined with rules (in the right order!), then you’re ready to move ahead with creating the launchd item for pf so that it starts automatically.

*However*, there is one additional thing you might want to do first—test your rules to be sure everything is correct. Use this command in a terminal window while running as an administrative user:

sudo pfctl -v -n -f <path to configuration file>

If this command reports errors, go back and fix them before proceeding.

Creating the launchd Item for pf

Creating the launchd item simply involves creating a properly-formatted XML file and placing it in /Library/LaunchDaemons. It must be owned by root, otherwise it won’t be processed at all. If you aren’t clear on how to make sure it’s owned by root, go do a bit of reading on sudo and chown.

Here’s a launchd item you might use for pf (click [here](https://gist.github.com/scottslowe/5581726) for an option to download this code snippet):

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE plist PUBLIC "-//Apple Computer/DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">

<dict>

<key>Label</key>

<string>org.scottlowe.pf.plist</string>

<key>Program</key>

<string>/sbin/pfctl</string>

<key>ProgramArguments</key>

<array>

<string>/sbin/pfctl</string>

<string>-e</string>

<string>-f</string>

<string>/etc/pf.anchors/org.scottlowe.pf.conf</string>

</array>

<key>RunAtLoad</key>

<true/>

<key>ServiceDescription</key>

<string>FreeBSD Packet Filter (pf) daemon</string>

<key>StandardErrorPath</key>

<string>/var/log/pf.log</string>

<key>StandardOutPath</key>

<string>/var/log/pf.log</string>

</dict>

</plist>

A few notes about this launchd item:

* You’ll want to change the last <string> item under the ProgramArguments key to properly reflect the path and filename of the custom configuration file you created earlier. In my case, I’m storing both the configuration file and the anchor in the /etc/pf.anchors directory.
* As I stated earlier, you *must* ensure this file is owned by root once you put it into /Library/LaunchDaemons. It won’t work otherwise.
* If you have additional parameters you want/need to pass to pfctl, add them as separate lines in the ProgramArguments array. Each individual argument on the command line must be a separate item in the array.

Once this file is in place with the right ownership, you can either use launchctl to load it or restart your computer. The robust pf firewall should now be running on your OS X Mountain Lion system. Enjoy!

Some Additional Resources

Finally, it’s important to note that I found a few different web sites helpful during my experimentations with pf on OS X. [This write-up](http://krypted.com/mac-os-x/a-cheat-sheet-for-using-pf-in-os-x-lion-and-up/) was written with Lion in mind, but applies equally well to Mountain Lion, and [this site](https://calomel.org/pf_config.html)–while clearly focused on OpenBSD and FreeBSD—was nevertheless quite helpful as well.

It should go without saying, but I’ll say it nevertheless: courteous comments are welcome! Feel free to add your thoughts, ideas, questions, or corrections below.