Getting Started: m0n0wall Development (v0.1.0)

Michael Iedema <iedemam@pluto.dsu.edu>

Introduction

Getting started developing for a new architecture, language, or even using a new environment can be a daunting task. After embarking on a semester long wireless network security project I purchased two Soekris Engineering[1] 4511 embedded PC's to use for research and testing. I received the equipment in March of 2003 and began investigating how other people have been using these boards.

Soekris's technical support section included a link to the m0n0wall[2] project. m0n0wall is an all-in-one firewall software package based on FreeBSD[3] 4.9 designed for embedded PC's. At that point in time m0nwall was in its infancy. The software was stable but lacking many of the advanced features it has today.

People adopting this software began looking for ways to aid in its development. The first hurdle in this process is breaking apart the binary image files that m0n0wall is distributed in.[4] The m0n0wall hacker's guide[5] written by Rudi van Drunen thoroughly outlines the m0n0wall development process but hasn't been updated since July of 2003. The depth of the hacker's guide may be daunting for some people to dive into. In a recent project for Linux Administration class at Dakota State University[6] I attempted to summarize my experiences in customizing

[1] Soekris Engineering (http://www.soekris.com)
[2] m0n0wall (http://m0n0.ch/wall)
[3] FreeBSD Project (http://www.freebsd.org)
[4] **note:** m0n0wall is also distributed in source but directly modifying the images is, in my opinion, less of a headache.
[5] m0n0wall hacker's guide (http://m0n0.ch/wall/hack)
[6] Dakota State University (http://www.dsu.edu)

m0n0wall images. This document expands on the material presented in that project.

Reasons for Development

If you're reading this document chances are you have the "why" question answered. If that is the case feel free to skip this section. On the other hand you may be investigating m0n0wall as firewall solution for your business, school, home office, or personal network. Read on to see if this software is what you're looking for and then how you can customize it to suit your needs.

Version 1.0 of m0n0wall contains a feature rich firewall rule interface, numerous NAT configurations, PPTP VPN with RADIUS support, IPSEC with roaming client support, traffic shaping, wireless AP/client support, SNMP support, caching DNS forwarder, remote Syslog, a highly configurable DHCP server, and many other features. These features encompass the majority of configuration scenarios but your situation may call for a small piece of software to be added. You may also wish to customize the look and feel of the m0n0wall webGUI if you are deploying this solution to clients.[7]

Another, often overlooked, reason to customize the m0n0wall installation image would be to replace the default XML configuration file with one pre-built for your specific conditions. This way every installation you make with the image will follow your custom configuration with no need for any further intervention.

[7] **note:** see the m0n0wall license for details (http://m0n0.ch/wall/license.php)

Assembling Your Toolbox

You will need a few tools to get your development environment setup properly. A brief rundown of each component follows.

→ Development System – A computer which you can install FreeBSD 4.9 onto or an existing system running FreeBSD 4.9 which you have root access on.[8] If you are using an existing system I would advice against using anything 'production' in nature. Development work of any kind can be unpredictable.

→ Test Firewall – An embedded or regular pc to run m0n0wall on which you can simulate any scenarios you may be developing for. Simple conversions can be made for generic-pc -> embedded m0n0wall images if your development system is not the same as your target system.[9]

→ Test Environment – You will also most likely need a couple of client test computers and possibly an extra switch to ensure the features you are adding are working correctly.

→ Scripts – m0n0image[10] is a php[11] shell script to aid in the development process. It simply automates the rather large number of commands needed to extract and compress m0n0wall images. Nothing fancy, just handy. It's usage follows.

```
usage: ./m0n0image.php [-x,-c] imagename.img
  -x  extract contents of image into DEVEL_imagename
  -c  compress DEVEL_imagename folder into a bootable image
```

Development Environment Installation

Your development system must have FreeBSD 4.9 installed on it as mentioned earlier. In addition to the installation several software packages must be installed and the kernel source must also be updated. These additional processes are outlined in detail but the installation procedure is treated with a fair amount of generality. I'm assuming a little familiarity with installing UNIX like

[8] **note:** this document will assume a fresh computer needing an OS install
[9] **note:** The practice of developing on one architecture and deploying on another is pretty common. You will most likely not
[10] m0n0image (http://michael-i.com/files/projects/m0n0image)
[11] PHP (http://www.php.net)

operating systems. Please consult the FreeBSD handbook[12] if you have any questions along the way.

Operating System Installation

1. Install FreeBSD 4.9 on your development computer using a standard installation.
2. When asked which distributions to install select all except 'X' related items.
3. Choose 'YES' when the option to install the ports collection appears.
4. Ensure to add another user besides root with wheel access granted.

Software Package Installation

After the Operating System has been installed we now must add some software packages to aid in the development process.

1. Launch the system installer by executing:

```
$ /stand/sysinstall
```

2. Select **configuration -> packages -> ftp -> freebsd main site**
3. Select the following packages:
   → **lang/php4-cli-4.3.3**
   → **net/cvsup-without-gui-16.1h**
   → **www/links-0.98,1**
   → **ftp/wget-1.8.2_4**

5. After selecting these packages select Install.
6. If you haven't downloaded the m0n0image script yet you can do by executing:

```
$ wget http://michael-i.com/files/projects/m0n0image/m0n0image_current.php
```

7. You will also need a recent m0n0wall build for your desired platform. So open the m0n0wall homepage in links.
8. Download an image for your target platform.

[12] FreeBSD Handbook (http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/index.html)

Update Your System's Kernel Source Tree

m0n0wall is built upon the latest stable source tree from FreeBSD. It also utilizes some patches specific to the m0n0wall project. First we need to setup a program used to ease the source update process called CVSup.

1. Execute the following command to create a new directory in etc:

```
$ mkdir /etc/cvsup
```

2. Create a new file to hold CVSup's configuration:

```
$ touch cvsup_releng_4_9
```

3. Use your favorite editor to add the following information into this config file.[13]

```
*default host=cvsup5.FreeBSD.org
*default tag=RELENG_4_9
*default prefix=/usr
*default base=/etc/cvsup
*default release=cvs delete use-rel-suffix
src-all
```

4. All that's left is to execute CVSup to synchronize your source tree with FreeBSD 4.9-Stable:

```
$ /usr/local/bin/cvsup /etc/cvsup/cvsup_releng_4_9
```

[13] **note:** The first line of this file can contain a number of addresses, visit this page to find the server closest to you. The listing is under "CVSup Sites." (http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvsup.html)

Patching the Kernel Source

Now we must download and apply the custom kernel patches that m0n0wall utilizes to our updated source tree.

1. Use wget to retrieve the kernel patches from the m0n0wall website:

```
$ wget http://m0n0.ch/wall/downloads/kernel-patches.tgz
```

2. Move and extract this tarball into a new directory in **`/usr/src/`** named **`m0n0`**.

```
$ mkdir /usr/src/m0n0
$ mv kernel-pathes.tgz /usr/src/m0n0
$ tar zxf kernel-patches.tgz
```

3. For each file in this tarball execute the following command to apply them to the source tree.

```
$ patch < filename.patch
```

This will bring your kernel source up to date allowing compatible kernel binaries to be built from it.

Extracting the m0n0wall Kernel Configuration File

Now we move onto extracting a kernel configuration file to begin working from. The m0n0wall kernel configuration file is available on the website but an interesting post to the user mail list outlines an alternate method.

1. Change into the directory where you downloaded the m0n0wall binary install image and run the following command as root to extract it with m0n0image.

```
 $ ./m0n0image.php -x imagename.img
```

2. Change into the newly created directory of the image's contents and navigate to where the kernel is stored.

```
 $ cd DEVEL_imagename/kern
```

3. Unzip the kernel.

```
$ gunzip kernel.gz
```

4. Run this fancy command to parse out the kernel configuration file used to generate the binary image and redirect the output into a file. You may wish to use a more appropriate filename for the output. Appending the intended platform and current date is usually good practice.

```
$ strings -n 3 kernel_tmp | sed -n 's/^___//p' > M0N0_GENERIC
```

5. Move this new file into the i386 configuration directory and clean up after ourselves.

```
$ mv M0N0_GENERIC /usr/src/sys/i386/conf
$ cd ../../
$ rm –rf DEVEL_imagename
```

6. Finally, we may have to do a bit of clean-up on the generated file so open it up in your favorite editor. Sometimes the top of the file can have a line or two of garbled text. Delete these and save the file.

```
$ vi /usr/src/sys/i386/conf/M0N0_GENERIC
```

At this stage in the document your environment is ready to be used for m0n0wall development. If you already have a project in mind and are familiar with m0n0wall and FreeBSD feel free to go ahead and begin developing. The only other piece of information you need (how to compress your DEVEL directory into a bootable .img file) is trivial.

```
$ ./m0n0image.php –c DEVEL_imagename
```

The following section outlines an example modification to give you an idea of the process and hopefully spark your interest in creating some new features for m0n0wall. They're always welcome!

An Example Modification

Our development environment is fully prepared for us to begin our tinkering. I will run through a brief example to show a simple customization. We are going to modify a m0n0wall image to include SMP support in the kernel. We are also going to expand the PPTP user limit from 16 to 50 and, consequently, the PPTP subnet mask. This would be handy in developing a powerful VPN solution with m0n0wall as a starting point.

Kernel Configuration

To add SMP support to this image we must build a customized kernel. This customization only makes sense for a generic-pc build of m0n0wall. Ensure this is the image you've downloaded in the previous steps and extracted the kernel configuration file. The remainder of this process is actually quite simple now that our environment is in place.

1. Make a copy of our kernel configuration file to avoid later needing to extract a fresh copy and open this new copy in your favorite editor.

```
$ cd /usr/src/sys/i386/conf
$ cp M0N0_GENERIC M0N0_GENERIC_50VPN
$ vi M0N0_GENERIC_50VPN
```

2. Add the following lines to the file after the device section in your kernel configuration file.

```
# Required To Build SMP Kernel
options SMP
options APIC_IO

# Optionally, you may specify the number of CPU's present
options NCPU=2
```

3. Execute the following command to configure your kernel based on this new configuration file

```
$ config M0N0_GENERIC_50VPN
```

4. The following commands are needed to compile your kernel binary based on this new configuration.

```
$ cd /sys/compile/M0N0_GENERIC_50VPN
$ make NO_MODULES=yes depend && make NO_MODULES=yes
```

5. Compress and move this newly created kernel into your DEVEL_imagename directory

```
$ gzip -9 kernel
$ mv kernel.gz /path/to/your/DEVEL_imagename/kern/
```

The kernel is ready to go, now we move onto the file system.

File System Configuration

The file system will require two changes in order to expand the number of PPTP users from 16 to 50. As mentioned earlier this change then results in a broken PPTP subnet mask so that must be calculated and altered as well. Both of these settings are in the same file so modifying them is a straightforward process.

6. Change into your DEVEL_imagename directory's file system subdirectory.

```
$ cd /path/to/your/DEVEL_imagename/fs/
```

7. Upon listing this directory you will discover that it contains the root file system which m0n0wall loads into RAM upon boot. Change into following directory and open globals.inc in your favorite editor.

```
$ cd etc/inc/
$ vi globals.inc
```

8. Locate the $g (global) array and change the following line.

```
"n_pptp_units" => 16,"  (change 16 to your desired user limit)
"n_pptp_units" => 50,"
```

9. Calculating the needed modification to our subnet requires a bit of research if you're unfamiliar with CIDR[14] notation. After a quick glance at the chart we see we need a /26 to accommodate 64 hosts. In the same $g array alter the following line to make this change.

```
"pptp_subnet" => 28,"  (change 28 to your desired subnet mask)
"pptp_subnet" => 26,"
```

10. Finally, recreate the image, rename it to something logical and you're done!

```
$ cd ../../../../
$ ./m0n0image.php -c DEVEL_imagename
$ mv DEVEL_imagename.img imagename.pptp50.img
```

Wrap-Up

In this document I've attempted to lower the barrier to entry on m0n0wall development. If you feel like playing around with this stuff start off with something simple like changing the font used in the webGUI. Work your way up from there as you become more familiar with the process.

Further additions to this document are in order such as how to upgrade your ports tree to add new software packages. Also, m0n0image must be expanded to take care of some of the tedium in extracting a kernel configuration file.

Please send feedback about your success or failures in using these methods. The last thing I want people doing is fighting with the development environment. Development isn't about the environment, it's about the product you are creating.

- Happy Hacking,
- Michael I.

[14] CIDR Information (http://public.pacbell.net/dedicated/cidr.html)

```
Changelog

  [+] new feature
  [*] changed or improved existing feature
  [-] bug fix
  [!] comments


Version 0.1.0 (05/06/04)
  [!] initial document release
```