# BSD MAGAZINE

WHAT'S NEW

NOW 84 PAGES

## INFINITY. FREEDOM. FREEBSD.

Vol.3No.1
Price USD14.99
Issue 1/2010(7)
1898-9144

# Orion iX-N4224

Call 1-800-820-BSDi Today!

*The* **Orion iX-N4224 SYSTEM** *is an* **excellent** *platform for ZFS Implementation!*

**The Orion features unrivaled cooling efficiency, power efficiency and excellent storage density!**

## Superior Savings On Energy Costs

**Designed for storage-intensive applications,** ZFS implementation, and virtualization, the iX-N4224 4U storage server series delivers incredible performance, storage capacity, and energy savings with adaptability and superior hardware. Each iX-N4224 comes with a Gold Level power supply, boasting a high 93% energy efficiency. Powerful Intel® Xeon® 5500 series quad core processors intelligently save power during low-use periods and increase performance when systems require it. Intel® Xeon® 5500 series processors include virtualization technologies to lead the way in performance, scalability, and simplified server management and migration.

**The iX-N4224 supports up to 144GB of DDR3** 1333 energy efficient RAM and utilizes three 5000 RPM cooling PWM fans and two 5000 RPM rear exhaust PWM fans. iX-N4224 servers offer up to 48 terabytes of storage with 24 hot-swappable SAS/SATA drive bays in a 4U configuration. Storage sizes for the iX-N4224 are customizable, with 250MB, 500MB, 750MB, 1TB, and 2TB hard drives available.

**The iX-N4224 provides the ideal solution** for applications requiring maximum storage capacity and power savings. For particularly storage-hungry applications, Western Digital® offers 2TB WD™ RE4-GP hard drives, which offer lower power use during idle times, a 64 megabyte cache, up to 25% increased performance, and a savings of up to $10 per drive on yearly power costs. Each hard drive is equipped with improvements to rotary vibration tolerance and calculates optimum seek speeds to lower power consumption, noise, and vibration. These drives also require less power and time to start up, allowing more drives to start spinning simultaneously due to the decrease in the current each drive requires. Equipping the Orion iX-N4224 4U storage server with the WD™ RE4-GP drives provides unparalleled storage capacity and power efficiency.

For more information about the Orion Series visit *http://www.iXsystems.com/Orion*.

## Features

- Dual 64-Bit Socket 1366 Quad-Core or Dual-Core Intel® Xeon® Processor 5500 Series
- 24 x 3.5" SAS/SATA Hot-swappable Drive Bays
- 1200W high-efficiency (1+1) redundant power supply (Gold Level 93%)
- 100% Cooling Redundancy
- Dual Intel® 5520 chipsets with QuickPath Interconnect (QPI)
- Up to 144GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (18 DIMM Slots)
- 2 PCI-E 2.0 x16, 4 PCI-E x8 (1 in x16 slot), and 1 PCI-E x4 Expansion Slots
- Intel® 82576 Dual Port Gigabit Ethernet Controller
- Optional 2x Internal Fixed 3.5" HDD or 2x fixed 2.5" HDD + DVD
- Matrox G200eW Graphics
- Remote Management - IPMI 2.0 + IP-KVM with dedicated LAN

iX systems™

intel Xeon inside™

Powerful. Intelligent.

**800-820-BSDi**
http://www.iXsystems.com
Enterprise Servers for Open Source

Dear Readers!

During this special time, I would like to wish you love, piece, happines and everything you wish for.

In short – Merry Christmas!

Thank you for your support, enjoy your presents and keep the emails coming in – I love hearing from you!

[illegible]

Karolina Lesinska

MAGAZINE BSD

**Editor in Chief:** Karolina Lesińska
*karolina.lesinska@bsdmag.org*

**Contributing:** Remko Lodder, Richard Bejtlich, Edd Barrett, Eric Vintimilla, Jacques Manukyan, Jan Stedenhouder, Diego Montalvo, Petr Topiarz, Donald T. Hayford, Francisco Reyes, Matthias Pfeifer, Richard C. Batka, Dru Lavigne, Mikel King, Brian D'Arcangelo

**Special thanks to Douglas Broome and Randi Harper**

**Art Director:** Agnieszka Marchocka
**DTP Technician:** Ireneusz Pogroszewski
Przemysław Banasiewicz

**Senior Consultant/Publisher:**
Paweł Marciniak *pawel@software.com.pl*

**National Sales Manager:** Ewa Dudzic *ewa.dudzic@bsdmag.org*
**Marketing Director:** Ewa Dudzic *ewa.dudzic@bsdmag.org*
**Executive Ad Consultant:**
Karolina Lesińska
*karolina.lesinska@bsdmag.org*
**Advertising Sales:** Karolina Lesińska
*karolina.lesinska@bsdmag.org*

**Publisher :**
**Software Press Sp. z o.o. SK**
**ul. Bokserska 1, 02-682 Warszawa**
**Poland**
worldwide publishing

tel: 1 917 338 36 31
*www.bsdmag.org*

Software Press Sp z o.o. SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: *editors@bsdmag.org*

Print: ArtDruk www.artdruk.com

Distributed in the USA by: Source Interlink Fulfillment Division, 27500 Riverview Centre Boulevard, Suite 400, Bonita Springs, FL 34134 Tel: 239-949-4450.

All trade marks presented in the magazine were used only for informative purposes. All rights to trade marks presented in the magazine are reserved by the companies which own them.

The editors use automatic DTP system AURUS

Mathematical formulas created by Design Science MathType™.
DVDs tested by AntiVirenKit GDATA Software Sp. z o.o.

**Subscription:**
email: *subscription_support@bsdmag.org*
phone: 1 917 338 36 31

Fax + 48 (0) 22 244 24 59

# Keeping FreeBSD Up-To-Date: OS Essentials

Richard Bejtlich

An important system administration task, and a principle of running a defensible network, is keeping operating systems and applications up-to-date.

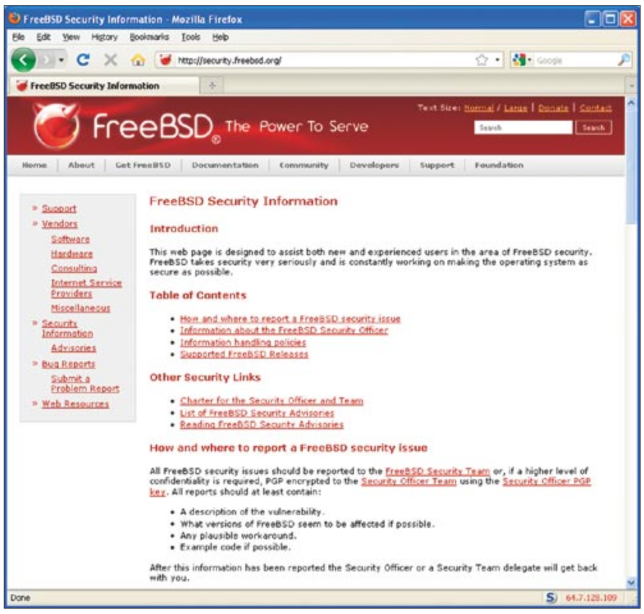
Running current software is critical when older services are vulnerable to exploitation. Obtaining new features not found in older applications is another reason to run current software. Fortunately, open source software offers a variety of means to give users a secure, capable computing environment.

This article presents multiple ways to keep the FreeBSD operating system up-to-date. I take a FreeBSD 7.1 RELEASE system through a subset of security advisories to explain the different sorts of patches an administrator might apply. It is important to realize that this article discusses the OS only; it does not discuss applications. FreeBSD does not have a unified update mechanism for the OS and applications. By applications I mean software outside of the kernel and userland. For example, Debian systems can use the apt tool to keep the distribution and packaged applications up-to-date. FreeBSD does not have a single equivalent tool, so this article only addresses keeping the OS up-to-date.

Note that there is a difference between an update and an upgrade. I use the term update to refer to keeping a certain version of FreeBSD up-to-date. For example, keeping a FreeBSD 7.1 system at version 7.1, but having the appropriate security and critical patches applied, qualifies as an update process. I use the term upgrade to refer to changing the FreeBSD version, either within a minor version or to a new major version. For example, migrating from FreeBSD 7.1 to 7.2, or from 7.2 to 8.0, qualify as upgrade processes.

I chose FreeBSD 7.1, released in January 2009, as my starting point because it offers a security history suitable for describing multiple update cases. At the time of writing FreeBSD 7.2 is the latest STABLE release and 8.0 is in BETA. Readers wondering why someone might want to install an *old* OS version can imagine that there might be an application supported only on FreeBSD 7.1 and not yet officially ready for 7.2 or 8.0, prompting an administrator to run a 7.1 box.

All of the work done in this article was done remotely via OpenSSH. One danger of performing remote upgrades is losing connection during a critical phase of the process. One software-based way to deal with this issue is to conduct all remote upgrades within a screen(1) session. (*http://www.freshports.org/misc/screen*) Should you lose connectivity during the upgrade while running screen, your session will continue uninterrupted. The screen(1) program has suffered security problems in the past, so balance its features against the possible risks.


FreeBSD Security Information - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://security.freebsd.org/
FreeBSD Security Information
Text Size: Normal / Large | Donate | Contact
FreeBSD The Power To Serve
Search
Home About Get FreeBSD Documentation Community Developers Support Foundation
Support
Vendors
Software
Hardware
Consulting
Internet Service Providers
Miscellaneous
Security Information
Advisories
Bug Reports
Submit a Problem Report
Web Resources
Done
64.7.128.109

FreeBSD Security Information

Introduction

This web page is designed to assist both new and experienced users in the area of FreeBSD security. FreeBSD takes security very seriously and is constantly working on making the operating system as secure as possible.

Table of Contents

- How and where to report a FreeBSD security issue
- Information about the FreeBSD Security Officer
- Information handling policies
- Supported FreeBSD Releases

Other Security Links

- Charter for the Security Officer and Team
- List of FreeBSD Security Advisories
- Reading FreeBSD Security Advisories

How and where to report a FreeBSD security issue

All FreeBSD security issues should be reported to the FreeBSD Security Team or, if a higher level of confidentiality is required, PGP encrypted to the Security Officer Team using the Security Officer PGP key. All reports should at least contain:

- A description of the vulnerability.
- What versions of FreeBSD seem to be affected if possible.
- Any plausible workaround.
- Example code if possible.

After this information has been reported the Security Officer or a Security Team delegate will get back with you.

**Figure 1.** FreeBSD security page

My advice on administering this reference platform is based on deploying FreeBSD on servers, workstations, and laptops since 2000. The article represents a mix of my interpretations of official FreeBSD documentation, inputs from mentors, and the result of my own experimentation and deployment strategies. This guide cannot be anywhere near a complete reference on keeping FreeBSD up-to-date or maintaining a secure system. I strongly recommend reading the excellent FreeBSD Handbook as well as the multiple helpful published books on FreeBSD.

## FreeBSD Handbook and Absolute FreeBSD, 2nd Ed

Please note that Chapter 24, Updating and Upgrading FreeBSD, is the authoritative source for information on keeping the FreeBSD OS up-to-date (*http://www.freebsd.org/doc/en/books/handbook/updating-upgrading.html*). The reason I wrote this article was to show how these various mechanisms apply in practice, and which I prefer in production. I must also recommend Michael W. Lucas' excellent book *Absolute FreeBSD*, 2nd Ed (No Starch, 2008). Several other talented FreeBSD writers have produced books, but Michael's is my favorite. For deeper coverage on the topics in this article, please see the Handbook or Michael's book.

## The Short Answer: Updating FreeBSD with Binary Upgrades

If you want to jump straight to the easiest way to keep the FreeBSD OS up-to-date, without changing major or minor versions, and you are a standard user who has not customized his or her kernel and userland, follow these instructions. I present this first and with little introduction because it is the most basic and important step for keeping the FreeBSD OS up-to-date for the majority of users.

- Set proxy, if necessary using `setenv HTTP_PROXY http://myproxy:myport`.
- Run `freebsd-update fetch`.
- Run `freebsd-update install`.
- Reboot.

These steps are demonstrated on a FreeBSD 7.2 system installed from CD (see Listing 1).

Following those four steps will keep a generic FreeBSD system up-to-date.

Colin Percival's FreeBSD Update tool is one of the best new aspects of FreeBSD, in my opinion. Prior to applying binary updates, FreeBSD administrators had to rely on recompiling source code whenever updates needed to be applied. This included casual users operating standard systems as well as power users operating custom systems. With FreeBSD Update, casual users who are

**Listing 1.** Uname output for FreeBSD 7.2

```
freebsd7a# uname -a
FreeBSD freebsd7a.localdomain 7.2-RELEASE FreeBSD 7.2-RELEASE #0: Fri May
1 08:49:13 UTC 2009     root@walker.cse.buffalo.edu:/usr/obj/usr/src/sys/
GENERIC  i386

freebsd7a# setenv HTTP_PROXY http://172.16.2.1:3128
freebsd7a# freebsd-update fetch
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
Fetching public key from update5.FreeBSD.org... done.
Fetching metadata signature for 7.2-RELEASE from update5.FreeBSD.org...
done.
Fetching metadata index... done.
Fetching 2 metadata files... done.
Inspecting system... done.
Preparing to download files... done.
Fetching 26 patches.....10....20... done.
Applying patches... done.

The following files will be updated as part of updating to 7.2-RELEASE-p3:
/boot/kernel/if_bce.ko
/boot/kernel/if_bce.ko.symbols
/boot/kernel/if_fxp.ko
/boot/kernel/if_fxp.ko.symbols
/boot/kernel/kernel
/boot/kernel/kernel.symbols
/lib/libc.so.7
/lib/libthr.so.3
...edited...
/usr/sbin/named
/usr/sbin/nologin
/usr/sbin/ntpd

freebsd7a# freebsd-update install
Installing updates... done.
freebsd7a# reboot

freebsd7a# uname -a
FreeBSD freebsd7a.localdomain 7.2-RELEASE-p2 FreeBSD 7.2-RELEASE-p2 #0:
Wed Jun 24 00:57:44 UTC 2009     root@i386-builder.daemonology.net:/usr/
obj/usr/src/sys/GENERIC  i386
```

**Listing 2.** Uname output for FreeBSD 7.1

```
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.1-RELEASE FreeBSD 7.1-RELEASE #0: Thu Jan
1 14:37:25 UTC 2009     root@logan.cse.buffalo.edu:/usr/obj/usr/src/sys/
GENERIC  i386
```

not making changes to the standard kernel and userland can quickly and easily keep the FreeBSD OS up-to-date. With some careful use, even power users can benefit from binary updates.

The rest of the article demonstrates additional methods and details, depending on the administrator's needs.

## Understanding FreeBSD Versions

Before explaining ways to keep the FreeBSD OS up-to-date, I must briefly expand on the idea of the term `up-to-date`. Thanks to FreeBSD's open source development methodology, any version of FreeBSD is available via check out from the *Concurrent Versions System* (CVS). (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/anoncvs.html*) These versions can be represented by CVS revision tags. (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/cvs-tags.html*) The following examples begin with 7.2 RELEASE, the most recently published version of FreeBSD:

- `RELENG_7_2_0_RELEASE` is FreeBSD 7.2 RELEASE, just as you might get on CD. `RELENG_7_2_0_RELEASE` is also known as a *release tag.*
- `RELENG_7_2` is the *security* branch for 7.2, which is FreeBSD 7.2 RELEASE with patches for security advisories and critical fixes applied. `RELENG_7_2` is known as a *branch tag.*
- `RELENG_7` is the development line of the FreeBSD 7 tree, also known as 7-STABLE. `RELENG_7` is also a *branch tag.*
- `. ("dot")`, also known as HEAD, is the development line of the next version of FreeBSD, 8.0, also known as 8-CURRENT or simply CURRENT.

At the time of writing, the FreeBSD project was working the release process for FreeBSD 8.0. Creating FreeBSD 8.0 means declaring that, as of a certain date, FreeBSD 8-CURRENT will now be designated FreeBSD 8.0. From that point forward, CURRENT will be the future FreeBSD 9.0, so *CURRENT* will be considered 9-CURRENT.

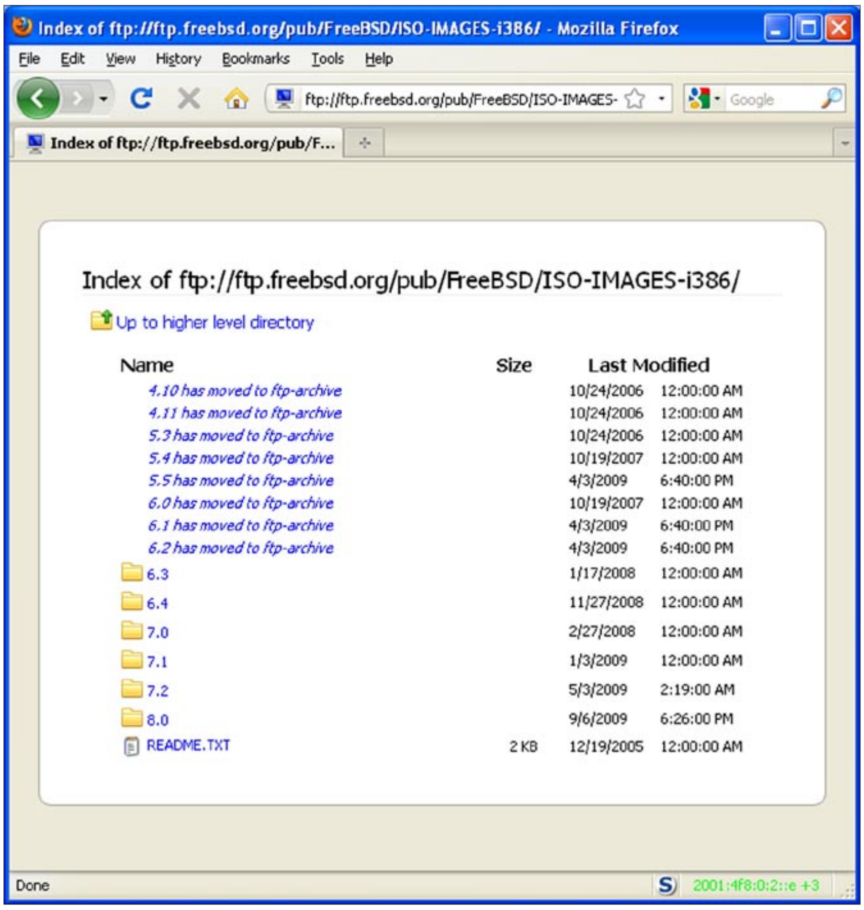
The bottom line is that CURRENT should always be thought of as the next major version of FreeBSD. When 7.2 was the newest FreeBSD version, CURRENT was being developed as FreeBSD 8.0. When FreeBSD 8.0 is released, CURRENT will be developed as FreeBSD 9.0.

Incidentally, during the release process for FreeBSD 8.0, various beta (*BETA*) and release candidate (*RC*) versions will be released to facilitate testing. In the article you will see references to FreeBSD 8.0-BETA versions, for example.

Linux users should note that these CVS revision tags do not pertain to the FreeBSD kernel alone. FreeBSD is developed as an integrated system, with a kernel matching userland tools. One should not run a kernel compiled for FreeBSD 7.2 RELEASE on

**Listing 3.** Installing GnuPG

```
freebsd7# pkg_add -vr gnupg
scheme:   [ftp]
user:     []
password: []
host:     [ftp.freebsd.org]
port:     [0]
document: [/pub/FreeBSD/ports/i386/packages-7.1-release/Latest/gnupg.tbz]
---> ftp.freebsd.org:21
looking up ftp.freebsd.org
connecting to ftp.freebsd.org:21
...edited...
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-7.1-
release/Latest/gnupg.tbz...
x +CONTENTS
...edited...
Package gnupg-2.0.9_2 registered in /var/db/pkg/gnupg-2.0.9_2
...truncated...
```


Index of ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-i386/ - Mozilla Firefox
File Edit View History Bookmarks Tools Help
ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-
Google
Index of ftp://ftp.freebsd.org/pub/F...
Done
2001:4f8:0:2::e +3

Index of ftp://ftp.freebsd.org/pub/FreeBSD/ISO-IMAGES-i386/

Up to higher level directory

| Name | Size | Last Modified | |
|---|---|---|---|
| 4.10 has moved to ftp-archive | | 10/24/2006 | 12:00:00 AM |
| 4.11 has moved to ftp-archive | | 10/24/2006 | 12:00:00 AM |
| 5.3 has moved to ftp-archive | | 10/24/2006 | 12:00:00 AM |
| 5.4 has moved to ftp-archive | | 10/19/2007 | 12:00:00 AM |
| 5.5 has moved to ftp-archive | | 4/3/2009 | 6:40:00 PM |
| 6.0 has moved to ftp-archive | | 10/19/2007 | 12:00:00 AM |
| 6.1 has moved to ftp-archive | | 4/3/2009 | 6:40:00 PM |
| 6.2 has moved to ftp-archive | | 4/3/2009 | 6:40:00 PM |
| 6.3 | | 1/17/2008 | 12:00:00 AM |
| 6.4 | | 11/27/2008 | 12:00:00 AM |
| 7.0 | | 2/27/2008 | 12:00:00 AM |
| 7.1 | | 1/3/2009 | 12:00:00 AM |
| 7.2 | | 5/3/2009 | 2:19:00 AM |
| 8.0 | | 9/6/2009 | 6:26:00 PM |
| README.TXT | 2 KB | 12/19/2005 | 12:00:00 AM |

**Figure 2.** FreeBSD versions at ftp.freebsd.org

a CURRENT machine. The kernel and all userland utilities are meant to be upgraded simultaneously, and must be kept synchronized. While Linux users are usually forced to acknowledge this good system administration practice when they upgrade major versions of their kernel (e.g., 2.4 to 2.6), they often maintain the same userland across minor kernel versions. FreeBSD strongly encourages users to always keep the userland and kernel in sync using the methods explained in the Handbook and elaborated upon in this document.

When thinking of what it means to be *up-to-date*, one can see that the oldest version of FreeBSD 7.2 is that which was *pressed to CD* – RELENG_7_2_0_RELEASE or FreeBSD 7.2 RELEASE. The newest version of FreeBSD 7.x would be 7-STABLE (also called 7.2-STABLE), a constantly moving target modified and improved on a daily basis. How does an administrator decide what to run on her machines?

I prefer to begin a system's life by installing `RELEASE` software, like FreeBSD 7.2 RELEASE. *As long as the systems performs* as I would expect it to, I then track the `RELENG_7_2` or *security* branch. This allows me to incorporate critical bug and security fixes that could jeopardize the system.

Occasionally I may encounter a system that requires a feature (like supporting a new piece of hardware) not present in the RELEASE or security branches. In cases where that feature is supported by STABLE, I will upgrade to that branch. In the rare cases where not even STABLE has the feature I need, I might install a snapshot of the CURRENT branch. I do not recommend running CURRENT in production environments as it is not supported like the RELEASE or STABLE versions are.

## Learning About Security Issues

FreeBSD security advisories are published at the FreeBSD security page and at the freebsd-security-notifications mailing list. (*http://www.freebsd.org/security/advisories.html* and *http://lists.freebsd.org/pipermail/freebsd-security-notifications/*) I recommend all FreeBSD users subscribe to the moderated, very low volume notification mailing list. The advisories provide

**Listing 4.** Installing GPG

```
freebsd7# gpg --import /usr/share/doc/en_US.ISO8859-1/books/handbook/
pgpkeys.html
gpg: directory '/root/.gnupg' created
gpg: new configuration file '/root/.gnupg/gpg.conf' created
gpg: WARNING: options in '/root/.gnupg/gpg.conf' are not yet active during
this run
gpg: keyring '/root/.gnupg/secring.gpg' created
gpg: keyring '/root/.gnupg/pubring.gpg' created
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key CA6CDFB2: public key "FreeBSD Security Officer <security-
officer@FreeBSD.org>" imported
gpg: key FF8AE305: public key "core-secretary@FreeBSD.org" imported
gpg: key 7414629C: public key "FreeBSD portmgr secretary <portmgr-
secretary@FreeBSD.org>" imported
gpg: Total number processed: 3
gpg:               imported: 3  (RSA: 1)
gpg: no ultimately trusted keys found
```

**Listing 5.** Contents of /usr/src

```
freebsd7# ls /usr/src
COPYRIGHT                 contrib                   rescue
LOCKS                     crypto                    sbin
MAINTAINERS               etc                       secure
Makefile          games                     share
Makefile.inc1             gnu                       sys
ObsoleteFiles.inc         include                   tools
README                    kerberos5                 usr.bin
UPDATING                  lib                       usr.sbin
bin                       libexec
cddl                      release
```

**Listing 6.** Checking out source code using CVS

```
freebsd7# cd /usr

freebsd7# cvs -d anoncvs@anoncvs1.freebsd.org:/home/ncvs co -r RELENG_7_1_
0 src
cvs checkout: Updating src
cvs checkout: Updating src/bin
cvs checkout: Updating src/bin/cat
...truncated...
```

**Listing 7.** Installing CVSup

```
freebsd7# pkg_add -vr cvsup-without-gui
...edited...
x bin/cvpasswd
x bin/cvsup
x sbin/cvsupd
...edited...
Package cvsup-without-gui-16.1h_4 registered in /var/db/pkg/cvsup-without-
gui-16.1h_4
```

**Listing 8.** Retrieving patch for ktimer

```
freebsd7# fetch http://security.FreeBSD.org/patches/SA-09:06/ktimer.patch
ktimer.patch                                      100% of  476  B   61 kBps

freebsd7# fetch http://security.FreeBSD.org/patches/SA-09:06/
ktimer.patch.asc
ktimer.patch.asc                                  100% of  195  B   24 kBps
```

**Listing 9.** Verifying ktimer patch

```
freebsd7# gpg --verify ktimer.patch.asc ktimer.patch
gpg: Signature made Sun Mar 22 19:59:58 2009 EDT using DSA key ID CA6CDFB2
gpg: Good signature from "FreeBSD Security Officer <security-
officer@FreeBSD.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:       There is no indication that the signature belongs to the owner.
Primary key fingerprint: C374 0FC5 69A6 FBB1 4AED  B131 15D6 8804 CA6C DFB2
```

**Listing 10.** Patching ktimer

```
freebsd7# patch < /root/ktimer.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|Index: sys/kern/kern_time.c
|===================================================================
|--- sys/kern/kern_time.c  (revision 190192)
|+++ sys/kern/kern_time.c  (working copy)
--------------------------
Patching file sys/kern/kern_time.c using Plan A...
Hunk #1 succeeded at 1079 (offset -6 lines).
done
```

**Listing 11.** Rebuilding the kernel

```
freebsd7# cd /usr/src/sys/i386/conf
freebsd7# cp GENERIC FREEBSD7
freebsd7# cd /usr/src
freebsd7# make buildkernel KERNCONF=FREEBSD7
--------------------------------------------------------------
>>> Kernel build for FREEBSD7 started on Thu Aug 20 11:01:55 EDT 2009
--------------------------------------------------------------
===> FREEBSD7
mkdir -p /usr/obj/usr/src/sys
--------------------------------------------------------------
>>> stage 1: configuring the kernel
--------------------------------------------------------------
cd /usr/src/sys/i386/conf;  PATH=/usr/obj/usr/src/tmp/legacy/usr/sbin:
/usr/obj/usr/src/tmp/legacy/usr/bin:/usr/obj/usr/src/tmp/legacy/usr/games:
/usr/obj/usr/src/tmp/usr/sbin:/usr/obj/usr/src/tmp/usr/bin:/usr/obj/usr/
src/tmp/usr/games:/sbin:/bin:/usr/sbin:/usr/bin  config  -d /usr/obj/usr/
src/sys/FREEBSD7  /usr/src/sys/i386/conf/FREEBSD7
Kernel build directory is /usr/obj/usr/src/sys/FREEBSD7
...edited...
--------------------------------------------------------------
>>> Kernel build for FREEBSD7 completed on Thu Aug 20 11:54:29 EDT 2009
--------------------------------------------------------------
```

background, a problem description, an impact statement, workaround advice, a solution to fix the problem, and correction details. We'll take a closer look at an actual security advisory when we learn how to apply patches manually to the operating system.

## Starting with the Installation

Let's start with the most common deployment scenario, using FreeBSD 7.1 RELEASE as our starting point. For this version, the CVS tag is `RELENG_7_1_0_RELEASE` for the version shipped on CD and `RELENG_7_1` for the security branch.

The administrator installs FreeBSD 7.1 RELEASE from CD on a new server. She installs the User distribution set (*Average User* – binaries and doc only) and installs the ports tree. When installation is done, a check of uname output shows what the system looks like prior to any changes: see Listing 2.

She does not need to modify the kernel and is running the GENERIC version shipped with the OS.

At this point the system is running, but it requires security updates.

## Installing Gnupg and Importing Keys

Whenever an administrator wants to manually apply a security patch, it is important to validate those patches using Gnu Privacy Guard (Gnupg, *http://www.freshports.org/security/gnupg*). In this section we will install Gnupg and import FreeBSD developer keys (see Listing 3).

Notice in the output above that the version of Gnupg shipped with FreeBSD 7.1 (in packages-7.1-release) is the version installed automatically here.

Next we import required PGP keys (see Listing 4).

With Gnupg installed, you will be able to check signatures on patches applied later.

## Installing Source Code

When the administrator installed FreeBSD 7.1, she did not install the source code for the system. We'll do that next.

FreeBSD source code can either be checked out from CVS online, or installed from other media. Since this system was just installed from CD, and we have the CD handy, we'll install the source code from CD.

The easiest way to install source code from CD is to use the sysinstall program.

First, note that the source code is not available yet on the system.

```
freebsd7# ls /usr/src
freebsd7#
```

Launch *sysinstall*.

- Select *Configure – Do post-install configuration of FreeBSD*
- Select *Distributions – Install additional distribution sets*
- Select *src – Sources for everything* by highlighting and hitting the space bar
- Select *All – Select all of the below* by highlighting and hitting return. Tab to OK and hit return.
- Tab to OK on the *Select the distributions you wish to install* page and hit return.
- Select *CD/DVD – Install from a FreeBSD CD/DVD* and hit return.
- Wait until the source code is installed, then exit sysinstall.

Now, listing `/usr/src` shows the source code is installed (Listing 5).

An alternative to installing the source code from CD involves using cvs to check it out. In this example we access an anonymous FreeBSD CVS server (*http://www.freebsd.org/doc/en/books/handbook/anoncvs.html*). For example: see Listing 6.

With the source code on the system, you will be able to manually apply patches and recompile the whole system or kernel as necessary.

## Installing CVSup

The final addition to our FreeBSD 7.1 RELEASE system is the cvsup-without-gui package (see Listing 7).

It turns out that CVSup isn't really needed on modern FreeBSD systems, but I include it here because it is the single most recognizable update tool for FreeBSD.

At this point we have the infrastructure in place to try applying patches as required.

## Applying Kernel Patches Manually

In the following sections we will examine a variety of ways to keep FreeBSD up-to-date. In this section we will look at applying kernel patches manually. We've already seen how FreeBSD can make updating the GENERIC kernel very easy. However, the situation becomes more complicated when administrators run custom kernels or make other local modifications.

To demonstrate how to manually patch the FreeBSD kernel on our FreeBSD 7.1 RELEASE system, we will use the FreeBSD-SA-09:06.ktimer advisory as an example (*http://security.freebsd.org/advisories/FreeBSD-SA-09:06.ktimer.asc*).

To implement this advisory, we follow the instructions in part 2 (see Listing 8). Next we validate the patch (see Listing 9).

GPG warns us that we have not taken any steps to trust the signature of the FreeBSD Security Officer. One of the ways to make this warning disappear would be to sign the key of the FreeBSD Security Officer ourselves. We might do that after confirming in person or on the telephone that the primary key fingerprint of the FreeBSD Security Officer's key is as stated in the output above. (Beyond this example, I will not show verifying future patches.)

**Listing 12.** Installing kernel

```
freebsd7# make installkernel KERNCONF=FREEBSD7
--------------------------------------------------------------
>>> Installing kernel
--------------------------------------------------------------
cd /usr/obj/usr/src/sys/FREEBSD7;  MAKEOBJDIRPREFIX=/usr/obj  MACHINE_
ARCH=i386  MACHINE=i386  CPUTYPE=  GROFF_BIN_PATH=/usr/obj/usr/src/tmp/
legacy/usr/bin  GROFF_FONT_PATH=/usr/obj/usr/src/tmp/legacy/usr/share/
groff_font  GROFF_TMAC_PATH=/usr/obj/usr/src/tmp/legacy/usr/share/tmac
PATH=/usr/obj/usr/src/tmp/legacy/usr/sbin:/usr/obj/usr/src/tmp/legacy/usr/
bin:/usr/obj/usr/src/tmp/legacy/usr/games:/usr/obj/usr/src/tmp/usr/sbin:
/usr/obj/usr/src/tmp/usr/bin:/usr/obj/usr/src/tmp/usr/games:/sbin:/bin:
/usr/sbin:/usr/bin  make KERNEL=kernel install
thiskernel='sysctl -n kern.bootfile' ;  if [ ! "'dirname "$thiskernel"'"
-ef /boot/kernel ] ; then  chflags -R noschg /boot/kernel ;  rm -rf /
boot/kernel ;  else  if [ -d /boot/kernel.old ] ; then  chflags -R noschg
/boot/kernel.old ;  rm -rf /boot/kernel.old ;  fi ;  mv /boot/kernel
/boot/kernel.old ;  sysctl kern.bootfile=/boot/kernel.old/"'basename
"$thiskernel"'" ;  fi
kern.bootfile: /boot/kernel/kernel -> /boot/kernel.old/kernel
mkdir -p /boot/kernel
install -p -m 555 -o root -g wheel kernel /boot/kernel
...edited...
install -o root -g wheel -m 555   if_zyd.ko.symbols /boot/kernel
kldxref /boot/kernel
```

**Listing 13.** Uname output before recompiling kernel

```
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.1-RELEASE FreeBSD 7.1-RELEASE #0: Thu Jan
1 14:37:25 UTC 2009     root@logan.cse.buffalo.edu:/usr/obj/usr/src/sys/
GENERIC  i386
freebsd7# reboot
```

**Listing 14.** Uname output after recompiling kernel

```
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.1-RELEASE FreeBSD 7.1-RELEASE #0: Thu Aug
20 11:24:04 EDT 2009     root@freebsd7.localdomain:/usr/obj/usr/src/sys/
FREEBSD7  i386
```

**Listing 15.** Patching telnetd

```
freebsd7# fetch http://security.FreeBSD.org/patches/SA-09:05/telnetd.patch
telnetd.patch
                                       100% of 1010  B  280 kBps
freebsd7# fetch http://security.FreeBSD.org/patches/SA-09:05/
telnetd.patch.asc
telnetd.patch.asc                                      100% of  195  B   53 kBps

freebsd7# gpg --verify telnetd.patch.asc telnetd.patch
gpg: Signature made Mon Feb 16 16:30:19 2009 EST using DSA key ID CA6CDFB2
gpg: Good signature from "FreeBSD Security Officer <security-
officer@FreeBSD.org>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:        There is no indication that the signature belongs to the owner.
Primary key fingerprint: C374 0FC5 69A6 FBB1 4AED  B131 15D6 8804 CA6C DFB2

freebsd7# cd /usr/src
freebsd7# patch < /root/telnetd.patch
Hmm...  Looks like a unified diff to me...
The text leading up to this was:
--------------------------
|Index: contrib/telnet/telnetd/sys_term.c
|===================================================================
|--- contrib/telnet/telnetd/sys_term.c      (revision 188667)
|+++ contrib/telnet/telnetd/sys_term.c      (working copy)
--------------------------
Patching file contrib/telnet/telnetd/sys_term.c using Plan A...
Hunk #1 succeeded at 1285 (offset 14 lines).
Hunk #2 succeeded at 1310 (offset 14 lines).
done

freebsd7# cd /usr/src/lib/libtelnet
freebsd7# make obj && make depend && make

/usr/obj/usr/src/lib/libtelnet created for /usr/src/lib/libtelnet
rm -f .depend
mkdep -f .depend -a    -I/usr/src/lib/libtelnet/../../contrib/telnet -
DENCRYPTION -DAUTHENTICATION -DSRA -DKRB5 -I/lib/krb5 -I -I -DFORWARD
-Dnet_write=telnet_net_write /usr/src/lib/libtelnet/../../contrib/telnet/
libtelnet/genget.c
...edited...
building static telnet library
ranlib libtelnet.a

freebsd7# cd /usr/src/libexec/telnetd

freebsd7# make obj && make depend && make && make install
/usr/obj/usr/src/libexec/telnetd created for /usr/src/libexec/telnetd
rm -f .depend
mkdep -f .depend -a    -DLINEMODE -DUSE_TERMIO -DDIAGNOSTICS -DOLD_ENVIRON
-DENV_HACK -DINET6 -I/usr/src/libexec/telnetd/../../contrib/telnet -
DAUTHENTICATION -DENCRYPTION -DKRB5 -DFORWARD -Dnet_write=telnet_net_write
/usr/src/libexec/telnetd/../../contrib/telnet/telnetd/global.c
...edited...
install -s -o root -g wheel -m 555   telnetd /usr/libexec
install -o root -g wheel -m 444 telnetd.8.gz  /usr/share/man/man8
```

For now we assume that the patch has not been tampered with and move on to applying it per theadvisory's instructions. Now we apply the patch (see Listings 10).

Finally we compile a new kernel for our system. Note that we decide to make a copy of the configuration file called FREEBSD7. We do not leave the kernel as GENERIC because we have patched it (see Listing 11).

After waiting several minutes we install the new kernel (see Listing 12).

After a final check of the installed kernel (which is still running), we reboot (see Listing 13).

After reboot, notice that the new kernel is installed (see Listing 14).

The compilation date also matches the date the new kernel was compiled.

## Applying Userland Patches Manually

In the previous section we saw how to apply a patch to the kernel, then recompile and install the patched kernel. Here we will look at applying a patch to a userland application that ships with the FreeBSD OS. For this example we will use the FreeBSD-SA-09:05.telnetd advisory (*http://security.freebsd.org/advisories/FreeBSD-SA-09:05.telnetd.asc*).

To implement this advisory, we follow the instructions in part 2 (see Listing 15).

Since telnetd runs from inetd, we can be sure the next time telnetd starts it will be patched.

In the previous edition of this document (published in 2005), we provided an example of manually patching the userland for FreeBSD-SA-04:05.openssl. That advisory required recompiling the entire userland. The same is true for `FreeBSD-SA-06:23.openssl`. However, there does not seem to be an advisory since 2006 that required recompiling the whole userland. Even FreeBSD-SA-09:08.openssl, another OpenSSL advisory, only required recompiling part of the userland, as was the case with this telnetd example. In the event you wish to apply a userland patch manually, and it requires recompiling the userland, follow the instructions in the advisory as we have done with these last two examples.

## Using CVSup to Apply Patches

So far we have shown how to do quick binary updates using FreeBSD Update, and we manually applied a kernel patch and then a userland patch. In this example we will use the traditional CVSup tool to update the entire system to a specific point in time. For this example we will use the FreeBSD-SA-09:07.libc security advisory (*http://security.freebsd.org/advisories/FreeBSD-SA-09:07.libc.asc*) to guide our actions.

This security advisory requires a patch to libc. We could have user binary updates to fix this, or applied the security patch manually. Instead we are going to update the whole system to a time when the patch was integrated into the FreeBSD source tree. This is `Solution 1` in the advisory. We take the time from the *Corrected* section of the advisory. Because our system is running FreeBSD 7.1, we look for the date involving that version of FreeBSD.

```
2009-04-22 14:07:14 UTC (RELENG_7_1,
7.1-RELEASE-p5)
```

*This means we can update all of the source code on our system to a date after 2009-04-22 14:07:14 UTC to be sure the libc patch is applied.*

In order to do that, we will use CVSup. We need to create a *supfile* that controls how CVSup operates. Examples are on the system already (see Listing 16).

Please replace *INSERTYOURCHOICE.FreeBSD.org* in this and later occurrences with the hostname of a real CVSup server as listed in the FreeBSD Handbook (*http://www.freebsd.org/doc/en/books/handbook/cvsup.html*).

We set the date to be in the minute after the correction time noted earlier.

Now we are ready to use CVSup to update our source tree (see Listing 20).

Notice the last date listed for updates to src/UPDATING is less than the time specified in our supfile. There are no updates beyond 2009-04-22 14:07:14 UTC. This means CVSup is working as expected. In other words, we are getting updates to 7.1 RELEASE, but not newer than our specified correction date.

Note that CVSup does not natively support HTTP proxies. For information on how to use CVSup through a proxy, specifically mentioning FreeBSD, see my blog post Updating FreeBSD Using CVSup through HTTP Proxy (*http://taosecurity.blogspot.com/2009/08/updating-freebsd-using-cvsup-*

**Listing 16.** Example supfiles

```
freebsd7# ls /usr/share/examples/cvsup
README                    ports-supfile             standard-supfile
cvs-supfile               refuse                    www-supfile
doc-supfile               refuse.README
gnats-supfile             stable-supfile
We create our own file with these contents:

freebsd7# cat /usr/local/etc/freebsd7-example.supfile
*default host=INSERTYOURCHOICE.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_7_1
*default delete use-rel-suffix
*default date=2009.04.22.14.08.00
*default compress
src-all
```

**Listing 17.** Running CVSup

```
freebsd7# cvsup -g -L 2 /usr/local/etc/freebsd7-example.supfile
Parsing supfile "/usr/local/etc/freebsd7-example.supfile"
Connecting to cvsup3.FreeBSD.org
Connected to cvsup3.FreeBSD.org
Server software version: SNAP_16_1h
Negotiating file attribute support
Exchanging collection information
Establishing multiplexed-mode data connection
Running
Updating collection src-all/cvs
 Edit src/UPDATING
  Add delta 1.507.2.13.2.4 2009.01.07.20.17.55 simon
  Add delta 1.507.2.13.2.5 2009.01.13.21.19.27 simon
  Add delta 1.507.2.13.2.6 2009.02.16.21.56.17 cperciva
  Add delta 1.507.2.13.2.7 2009.03.23.00.00.50 cperciva
  Add delta 1.507.2.13.2.8 2009.04.22.14.07.14 cperciva
 Edit src/contrib/bind9/lib/dns/openssldsa_link.c
  Add delta 1.1.1.3.2.1.4.1 2009.01.13.21.19.27 simon
 Edit src/contrib/bind9/lib/dns/opensslrsa_link.c
  Add delta 1.1.1.4.6.1 2009.01.13.21.19.27 simon
...edited...
 SetAttrs src/usr.sbin/pkg_install/tkpkg,v
Shutting down connection to server
Finished successfully
```

**Listing 18.** Commands to rebuild and install userland and kernel from source

```
cd /usr/src
make buildworld
make buildkernel KERNCONF=FREEBSD7
make installkernel KERNCONF=FREEBSD7
mergemaster -p
make installworld
mergemaster
reboot
```

**Listing 19a.** Demonstrating rebuilding and installing userland and kernel from source

```
-# $FreeBSD: src/etc/master.passwd,v 1.40.18.1 2008/
11/25 02:59:29 kensmith Exp $
+# $FreeBSD: src/etc/master.passwd,v 1.40 2005/06/06
20:19:56 brooks Exp $
 #
-root:$1$GblWCfv6$O..51HNClSYy5aEgE43Lx/:0:0::0:0:
Charlie &:/root:/bin/csh
+root::0:0::0:0:Charlie &:/root:/bin/csh
 toor:*:0:0::0:0:Bourne-again Superuser:/root:
 daemon:*:1:1::0:0:Owner of many system processes:
/root:/usr/sbin/nologin
 operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
@@ -21,4 +21,3 @@
 pop:*:68:6::0:0:Post Office Owner:/nonexistent:/usr/
sbin/nologin
 www:*:80:80::0:0:World Wide Web Owner:/nonexistent:
/usr/sbin/nologin
 nobody:*:65534:65534::0:0:Unprivileged user:/
nonexistent:/usr/sbin/nologin
-analyst:$1$FNYoY3Rk$lLVv/eHHIuLpz0AEAAYxO/:1001:1001:
:0:0:analyst:/home/analyst:/bin/sh

  Use 'd' to delete the temporary ./etc/master.passwd
  Use 'i' to install the temporary ./etc/master.passwd
  Use 'm' to merge the temporary and installed
versions
  Use 'v' to view the diff results again

  Default is to leave the temporary file to deal with
by hand

How should I deal with this? [Leave it for later] d
An alternative to deleting the temporary file and not
accepting changes is to manually integrate changes to
files.  See the FreeBSD Handbook for information on
that process.

In the following we show sample output from the entire
update process.

freebsd7# cd /usr/src
freebsd7# make buildworld
--------------------------------------------------
>>> World build started on Fri Aug 21 09:15:41 EDT
2009
--------------------------------------------------


--------------------------------------------------
>>> Rebuilding the temporary build tree
--------------------------------------------------
rm -rf /usr/obj/usr/src/tmp
mkdir -p /usr/obj/usr/src/tmp/legacy/usr/bin
mkdir -p /usr/obj/usr/src/tmp/legacy/usr/games
...edited...
===> etc/sendmail (all)
rm -f freebsd.cf
m4 -D_CF_DIR_=/usr/src/etc/sendmail/../../contrib/
sendmail/cf/   /usr/src/etc/sendmail/../../contrib/
sendmail/cf/m4/cf.m4 /usr/src/etc/sendmail/freebsd.mc
> freebsd.cf
chmod 444 freebsd.cf
rm -f freebsd.submit.cf
m4 -D_CF_DIR_=/usr/src/etc/sendmail/../../contrib/
sendmail/cf/   /usr/src/etc/sendmail/../../
contrib/sendmail/cf/m4/cf.m4 /usr/src/etc/sendmail/
freebsd.submit.mc > freebsd.submit.cf
chmod 444 freebsd.submit.cf

--------------------------------------------------
>>> World build completed on Fri Aug 21 12:34:00 EDT
2009
--------------------------------------------------

freebsd7# make buildkernel KERNCONF=FREEBSD7

--------------------------------------------------
>>> Kernel build for FREEBSD7 started on Fri Aug 21
12:34:28 EDT 2009
--------------------------------------------------
===> FREEBSD7
mkdir -p /usr/obj/usr/src/sys

--------------------------------------------------
>>> stage 1: configuring the kernel
--------------------------------------------------
...edited...
ld -Bshareable  -d -warn-common -o if_zyd.ko.debug
if_zyd.kld
objcopy --only-keep-debug if_zyd.ko.debug if_
zyd.ko.symbols
objcopy --strip-debug --add-gnu-debuglink=if_
zyd.ko.symbols if_zyd.ko.debug if_zyd.ko
--------------------------------------------------
---------
>>> Kernel build for FREEBSD7 completed on Fri Aug 21
13:35:05 EDT 2009
--------------------------------------------------

freebsd7# make installkernel KERNCONF=FREEBSD7
--------------------------------------------------
>>> Installing kernel
--------------------------------------------------
cd /usr/obj/usr/src/sys/FREEBSD7;  MAKEOBJDIRPREFIX=/
usr/obj  MACHINE_ARCH=i386
...edited...
install -o root -g wheel -m 555   if_zyd.ko.symbols
/boot/kernel
kldxref /boot/kernel

freebsd7# mergemaster -p
*** Unable to find mtree database. Skipping auto-
upgrade.
```

**Listing 19b.** Demonstrating rebuilding and installing userland and kernel from source, continued

```
*** Creating the temporary root environment in /var/
tmp/temproot
 *** /var/tmp/temproot ready for use
 *** Creating and populating directory structure in
/var/tmp/temproot
*** Beginning comparison
 *** Temp ./etc/master.passwd and installed have the
same CVS Id, deleting
 *** Temp ./etc/group and installed have the same CVS
Id, deleting
*** Comparison complete
Do you wish to delete what is left of /var/tmp/
temproot? [no]
 *** /var/tmp/temproot will remain
grep: /etc/make.conf: No such file or directory
*** Comparing make variables
*** From /etc/make.conf
*** From /usr/src/share/examples/etc/make.conf
freebsd7# make installworld
mkdir -p /tmp/install.fsulHZM5
for prog in [ awk cap_mkdb cat chflags chmod chown
date echo egrep find grep install-info  ln lockf
make mkdir mtree mv pwd_mkdb rm sed sh sysctl  test
true uname wc zic; do  cp 'which $prog' /tmp/
install.fsulHZM5;  done
...edited...
===> etc/sendmail (install)
cd /usr/src/etc/../share/man; make makedb
makewhatis /usr/share/man
makewhatis /usr/share/openssl/man
rm -rf /tmp/install.fsulHZM5
freebsd7# mergemaster
*** Unable to find mtree database. Skipping auto-
upgrade.
*** The directory specified for the temporary root
environment,
    /var/tmp/temproot, exists.  This can be a security
risk if untrusted
    users have access to the system.
  Use 'd' to delete the old /var/tmp/temproot and
continue
  Use 't' to select a new temporary root directory
  Use 'e' to exit mergemaster
  Default is to use /var/tmp/temproot as is
How should I deal with this? [Use the existing /var/
tmp/temproot]
   *** Leaving /var/tmp/temproot intact
*** Creating the temporary root environment in /var/
tmp/temproot
 *** /var/tmp/temproot ready for use
 *** Creating and populating directory structure in
/var/tmp/temproot
mtree -eU  -f /usr/src/etc/mtree/BSD.root.dist -p
/var/tmp/temproot/
./bin missing (created)
./boot missing (created)
./boot/defaults missing (created)
...edited...
 *** Temp ./etc/login.access and installed have the
same CVS Id, deleting
 *** Temp ./etc/login.conf and installed have the same
CVS Id, deleting
 *** Temp ./etc/mac.conf and installed have the same
CVS Id, deleting
==================================================
  *** Displaying differences between ./etc/motd and
installed version:
--- /etc/motd   2009-08-21 08:49:15.000000000 -0400
+++ ./etc/motd  2009-08-21 13:51:40.000000000 -0400
@@ -1,4 +1,4 @@
-FreeBSD 7.1-RELEASE (GENERIC) #0: Thu Jan  1 14:37:
25 UTC 2009
+FreeBSD ?.?.?  (UNKNOWN)

 Welcome to FreeBSD!
  Use 'd' to delete the temporary ./etc/motd
  Use 'i' to install the temporary ./etc/motd
  Use 'm' to merge the temporary and installed
versions
  Use 'v' to view the diff results again

  Default is to leave the temporary file to deal with
by hand

How should I deal with this? [Leave it for later] i

   *** ./etc/motd installed successfully
 *** Temp ./etc/netconfig and installed have the same
CVS Id, deleting
 *** Temp ./etc/network.subr and installed have the
same CVS Id, deleting
...edited...
 *** Temp ./.profile and installed have the same CVS
Id, deleting
 *** Temp ./COPYRIGHT and installed have the same CVS
Id, deleting

*** Comparison complete
*** Saving mtree database for future upgrades

Do you wish to delete what is left of /var/tmp/
temproot? [no]
 *** /var/tmp/temproot will remain

freebsd7# reboot

freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.1-RELEASE-p5 FreeBSD
7.1-RELEASE-p5 #0: Fri Aug 21 12:59:27 EDT 2009     ro
ot@freebsd7.localdomain:/usr/obj/usr/src/sys/FREEBSD7
i386
```

Listing 20a. FreeBSD-SA-09:09.pipe security advisory

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

======================================================
FreeBSD-SA-09:09.pipe
      Security Advisory

    The FreeBSD Project

Topic:          Local information disclosure via
direct pipe writes

Category:       core
Module:         kern
Announced:      2009-06-10
Credits:        Pieter de Boer
Affects:        All supported versions of FreeBSD.
Corrected:      2009-06-10 10:31:11 UTC (RELENG_7,
7.2-STABLE)
                2009-06-10 10:31:11 UTC (RELENG_7_2,
7.2-RELEASE-p1)
                2009-06-10 10:31:11 UTC (RELENG_7_1,
7.1-RELEASE-p6)
                2009-06-10 10:31:11 UTC (RELENG_6,
6.4-STABLE)
                2009-06-10 10:31:11 UTC (RELENG_6_4,
6.4-RELEASE-p5)
                2009-06-10 10:31:11 UTC (RELENG_6_3,
6.3-RELEASE-p11)

For general information regarding FreeBSD Security
Advisories,
including descriptions of the fields above, security
branches, and the
following sections, please visit <URL:http://
security.FreeBSD.org/>.

I.   Background
One of the most commonly used forms of interprocess
communication on FreeBSD and other UNIX-like systems
is the (anonymous) pipe.  In this mechanism, a pair of
file descriptors is created, and data written to one
descriptor can be read from the other.
FreeBSD's pipe implementation contains an optimization
known as "direct writes".  In this optimization,
rather than copying data into kernel memory when the
write(2) system call is invoked and then copying the
data again when the read(2) system call is invoked,
the FreeBSD kernel takes advantage of virtual memory
mapping to allow the data to be copied directly
between processes.

II.  Problem Description
An integer overflow in computing the set of pages
containing data to be copied can result in virtual-to-
physical address lookups not being performed.

III. Impact
An unprivileged process can read pages of memory which
belong to other processes or to the kernel.  These may
contain information which is sensitive in itself; or
may contain passwords or cryptographic keys which can
be indirectly exploited to gain sensitive information
or access.

IV.  Workaround
No workaround is available, but systems without
untrusted local users are not vulnerable.  System
administrators are reminded that even if a system is
not intended to have untrusted local users, it may
be possible for an attacker to exploit some other
vulnerability to obtain local user access to a system.

V.   Solution
Perform one of the following:
1) Upgrade your vulnerable system to 6-STABLE, or
7-STABLE, or to the RELENG_7_2, RELENG_7_1, RELENG_
6_4, or RELENG_6_3 security branch dated after the
correction date.
2) To patch your present system:
The following patches have been verified to apply to
FreeBSD 6.3, 6.4, 7.1, and 7.2 systems.
a) Download the relevant patch from the location
below, and verify the detached PGP signature using
your PGP utility.
# fetch http://security.FreeBSD.org/patches/SA-09:
09/pipe.patch
# fetch http://security.FreeBSD.org/patches/SA-09:
09/pipe.patch.asc
b) Apply the patch.
# cd /usr/src
# patch < /path/to/patch
c) Recompile your kernel as described in
<URL:http://www.FreeBSD.org/handbook/kernelconfig.html>
and reboot the system.

VI.  Correction details
The following list contains the revision numbers of
each file that was corrected in FreeBSD.

CVS:
Branch                                        Revision
  Path
-----------------------------------------------------
RELENG_6
  src/sys/kern/sys_pipe.c                   1.184.2.5
...edited...
RELENG_7
  src/sys/kern/sys_pipe.c                   1.191.2.5
RELENG_7_2
  src/UPDATING                           1.507.2.23.2.4
  src/sys/conf/newvers.sh                 1.72.2.11.2.5
  src/sys/kern/sys_pipe.c                 1.191.2.3.4.2
```

*through.html*). Now we are ready to execute the commands required to rebuild the system using source code. See Listing 18 for instructions.

Note in the following output, that when asked whether to install a change using the `i` input, we usually answer yes. The main exception invovles overwriting files used for authentication, like `/etc/passwd`. In the event a file like that is overwritten, the administrator can log in at the console as root (with no password), and then manually reinstall user accounts and set passwords.

In the following example, we do NOT install the file provided by the upgrade, because doing so would delete our `/etc/master.passwd` file (see Listing 19).

The system is now completely updated to the time specified in the supfile. However, the compilation date for the kernel shows when the kernel was compiled.

## Using Csup to Apply Patches

In the last example we used the traditional CVSup tool to apply patches to a system. Most FreeBSD administrators are very familiar with using that tool. However, since FreeBSD 6.2, a C replacement called Csup by Maxime Henrion has been available. In this example we will use the new Csup tool to update the entire system to a specific point in time. For this example we will use the FreeBSD-SA-09:09.pipe.asc security advisory (*http://security.freebsd.org/advisories/FreeBSD-SA-09:09.pipe.asc*) to guide our actions (see Listing 20a and 20b ).

This security advisory requires a patch to the kernel. We could have user binary updates to fix this, or applied the security patch manually. Instead we are going to update the whole system to a

**Listing 20b.** FreeBSD-SA-09:09.pipe security advisory

```
RELENG_7_1
  src/UPDATING                           1.507.2.13.2.9
  src/sys/conf/newvers.sh                 1.72.2.9.2.10
  src/sys/kern/sys_pipe.c                 1.191.2.3.2.2
-----------------------------------------------------

Subversion:

Branch/path                                  Revision
------------------------------------------------------
stable/6/                                     r193893
releng/6.4/                                   r193893
releng/6.3/                                   r193893
stable/7/                                     r193893
releng/7.2/                                   r193893
releng/7.1/                                   r193893
------------------------------------------------------

VII. References
The latest revision of this advisory is available at
http://security.FreeBSD.org/advisories/FreeBSD-SA-09:
09.pipe.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (FreeBSD)

iEYEARECAAYFAkovjN0ACgkQFdaIBMps37JkXwCgmLcEMOMAEIXRo
J220zwZhMKn
f+gAn1bZyLMhfZU7TI0xxhizwetDwMVI
=J37B
-----END PGP SIGNATURE-----
```

**Listing 21.** Supfile for specific CVS date

```
freebsd7# cat /usr/local/etc/freebsd7-example.supfile
*default host=INSERTYOURCHOICE.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_7_1
*default delete use-rel-suffix
*default date=2009.06.10.10.32.00
*default compress
src-all
```

**Listing 22.** CSup to a specific date

```
freebsd7# csup -g -L 2 /usr/local/etc/freebsd7-
example.supfile
Parsing supfile "/usr/local/etc/freebsd7-
example.supfile"
Connecting to cvsup3.FreeBSD.org
Connected to 128.31.0.28
Server software version: SNAP_16_1h
Negotiating file attribute support
Exchanging collection information
Establishing multiplexed-mode data connection
Running
Updating collection src-all/cvs
 Edit src/UPDATING
  Add delta 1.507.2.13.2.9 2009.06.10.10.31.11
cperciva
 Edit src/contrib/ntp/ntpd/ntp_crypto.c
  Add delta 1.1.1.3.18.1.2.2 2009.06.10.10.31.11
cperciva
 Edit src/sys/conf/newvers.sh
  Add delta 1.72.2.9.2.10 2009.06.10.10.31.11 cperciva
 Edit src/sys/kern/sys_pipe.c
  Add delta 1.191.2.3.2.2 2009.06.10.10.31.11 cperciva
 Edit src/sys/netinet6/in6.c
  Add delta 1.73.2.4.2.2 2009.06.10.10.31.11 cperciva
Shutting down connection to server
Finished successfully
```

**Listing 23.** Uname after updating to specific date

```
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.1-RELEASE-p6 FreeBSD
7.1-RELEASE-p6 #1: Fri Aug 21 19:35:25 EDT 2009      ro
ot@freebsd7.localdomain:/usr/obj/usr/src/sys/FREEBSD7
i386
```

Listing 24. Available binary updates

```
Index of /
         Name            Last Modified      Size        Type
   Parent Directory/                          -      Directory
   5.5-RELEASE/       2009-Jan-06 15:31:40 -      Directory
   6.0-RELEASE/       2009-Jan-06 15:31:40 -      Directory
   6.1-RELEASE/       2009-Jan-06 15:31:40 -      Directory
   6.2-RELEASE/       2009-Jan-06 15:31:40 -      Directory
   6.3-RELEASE/       2009-Jul-29 16:19:01 -      Directory
   6.4-RELEASE/       2009-Jul-29 16:19:09 -      Directory
   7.0-RELEASE/       2009-Apr-22 13:44:47 -      Directory
   7.1-BETA/          2009-Jan-06 15:31:40 -      Directory
   7.1-BETA2/         2009-Jan-06 15:31:40 -      Directory
   7.1-RC1/           2009-Jan-06 15:31:40 -      Directory
   7.1-RC2/           2009-Jan-07 20:16:13 -      Directory
   7.1-RELEASE/       2009-Jul-29 16:19:17 -      Directory
   7.2-BETA1/         2009-Apr-01 17:44:28 -      Directory
   7.2-RC1/           2009-Apr-22 14:00:51 -      Directory
   7.2-RC2/           2009-Apr-24 12:13:42 -      Directory
   7.2-RELEASE/       2009-Jul-29 16:19:26 -      Directory
   8.0-BETA1/         2009-Jul-30 06:04:42 -      Directory
   8.0-BETA2/         2009-Jul-30 06:04:51 -      Directory
   8.0-BETA3/         2009-Aug-23 21:25:58 -      Directory
   to-7.1-RELEASE/    2009-Jan-06 15:31:40 -      Directory
   to-7.2-BETA1/      2009-Apr-01 17:38:06 -      Directory
   to-7.2-RC1/        2009-Apr-16 15:20:25 -      Directory
   to-7.2-RC2/        2009-Apr-24 12:04:41 -      Directory
   to-7.2-RELEASE/    2009-May-02 17:45:12 -      Directory
   to-8.0-BETA1/      2009-Jul-08 01:06:26 -      Directory
   to-8.0-BETA2/      2009-Jul-17 19:08:49 -      Directory
   to-8.0-BETA3/      2009-Aug-23 22:04:57 -      Directory
   80BETA2.tar        2009-Jul-17 19:45:16 1.7G   application/x-tar
   80BETA3.tar        2009-Aug-23 22:38:23 1.5G   application/x-tar
   updates.tar        2009-Jul-30 06:32:07 13.9M  application/x-tar
```

Listing 25. Uname output for 7-STABLE and Failed FreeBSD Update for STABLE

```
fbsd71toS# uname -a
FreeBSD fbsd71toS.taosecurity.com 7.2-STABLE FreeBSD 7.2-STABLE #0: Sat
Aug 22 23:02:30 EDT 2009     root@fbsd71toS.taosecurity.com:/usr/obj/usr/
src/sys/FREEBSD7  i386
fbsd71toS# freebsd-update -v debug fetch
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
Fetching public key from update5.FreeBSD.org... fetch: http://
update5.FreeBSD.org/7.2-STABLE/i386/pub.ssl: Not Found
failed.
```

Listing 26. Supfile for FreeBSD 7.2

```
*default host=INSERTYOURCHOICE.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_7_2_0
*default delete use-rel-suffix
*default compress
src-all
```

time when the patch was integrated into the FreeBSD source tree. This is *Solution 1* in the advisory. We take the time from the *Corrected* section of the advisory. Because our system is running FreeBSD 7.1, we look for the date involving that version of FreeBSD.

```
2009-06-10 10:31:11 UTC (RELENG_7_1,
7.1-RELEASE-p6)
```

This means we can update all of the source code on our system to a date after 2009-06-10 10:31:11 UTC to be sure the kernel patch is applied.

In order to do that, we will use Csup. We will modify our earlier supfile that controls how Csup operates.

We set the date to be in the minute after the correction time noted earlier.

Now we are ready to use Csup to update our source tree (see Listing 22).

Now we can follow the same process as seen in the previous example (see Listing 18).

After rebooting, you see the new version of the FreeBSD kernel is installed (along with the userland).

As you can see, Csup is functionally equivalent to CVSup, and Csup is packaged with the FreeBSD OS.

## FreeBSD Update's Available Versions

In the first section of this paper, we saw FreeBSD Update used to keep a FreeBSD 7.2 system up-to-date. If you need to understand what sort of updates or upgrades are available for FreeBSD using freebsd-update, you can manually inspect one of the update sites. At the time of writing, visiting *http://update2.freebsd.org* displayed the following (see Listing 24).

Take the 7.2-RELEASE/ directory as an example. This means that FreeBSD Upgrade knows how to start with FreeBSD 7.2 RELEASE (as we started the article) and update or upgrade to the *to-* directories. FreeBSD Update does not have the capability to update from 4.x, for example, or from any STABLE version (e.g., 7.2-STABLE).

For example, if you tried to use FreeBSD Upgrade to *update* a 7.2-STABLE system, it will fail (see Listing 25).

If you are having trouble using FreeBSD Update, it's helpful to activate the '-v debug' switch to see what is happening.

**Listing 27a.** FreeBSD Update from 7.1 to 7.2

```
freebsd7# freebsd-update upgrade -r 7.2-RELEASE
Looking up update.FreeBSD.org mirrors... 3 mirrors found.
Fetching public key from update4.FreeBSD.org... done.
Fetching metadata signature for 7.1-RELEASE from
update4.FreeBSD.org... done.
Fetching metadata index... done.
Fetching 2 metadata files... done.
Inspecting system... done.
The following components of FreeBSD seem to be
installed:
kernel/generic src/base src/bin src/cddl src/contrib
src/crypto src/etc
src/games src/gnu src/include src/krb5 src/lib src/
libexec src/release
src/rescue src/sbin src/secure src/share src/sys src/
tools src/ubin
src/usbin world/base world/dict world/doc world/games
world/info
world/manpages world/proflibs
The following components of FreeBSD do not seem to be
installed:
world/catpages
Does this look reasonable (y/n)? y
Fetching metadata signature for 7.2-RELEASE from
update4.FreeBSD.org... done.
Fetching metadata index... done.
Fetching 1 metadata patches. done.
Applying metadata patches... done.
Fetching 1 metadata files... done.
Inspecting system... done.
Fetching files from 7.1-RELEASE for merging... done.
Preparing to download files... done.
Fetching 30039 patches.....10....20....30....40....50.
...60....70....80....90...
...edited...
..29390....29400....29410....29420....29430....29440..
..29450....29460....29470... done.
Applying patches... done.
Fetching 2273 files... done.
Attempting to automatically merge changes in files... done.
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 7.2-RELEASE have been merged into /etc/group:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/group,v 1.35.6.1 2008/11/25 02:
59:29 kensmith Exp $
+# $FreeBSD: src/etc/group,v 1.35.8.1 2009/04/15 03:
14:26 kensmith Exp $
 #
 wheel:*:0:root,analyst
 daemon:*:1:
 kmem:*:2:
 sys:*:3:
Does this look reasonable (y/n)? y
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 7.2-RELEASE have been merged into /etc/
master.passwd:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/master.passwd,v 1.40.18.1 2008/
11/25 02:59:29 kensmith Exp $
+# $FreeBSD: src/etc/master.passwd,v 1.40.20.1 2009/
04/15 03:14:26 kensmith Exp $
 #
 root:$1$xvjnCDHU$FEOCCNdR1r99CTDQdtBWo0:0:0::0:0:
Charlie &:/root:/bin/csh
 toor:*:0:0::0:0:Bourne-again Superuser:/root:
 daemon:*:1:1::0:0:Owner of many system processes:
/root:/usr/sbin/nologin
 operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
Does this look reasonable (y/n)? y
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 7.2-RELEASE have been merged into /etc/passwd:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/master.passwd,v 1.40.18.1 2008/
11/25 02:59:29 kensmith Exp $
+# $FreeBSD: src/etc/master.passwd,v 1.40.20.1 2009/
04/15 03:14:26 kensmith Exp $
 #
 root:*:0:0:Charlie &:/root:/bin/csh
 toor:*:0:0:Bourne-again Superuser:/root:
 daemon:*:1:1:Owner of many system processes:/root:
/usr/sbin/nologin
 operator:*:2:5:System &:/:/usr/sbin/nologin
Does this look reasonable (y/n)? y
The following files will be removed as part of updating
to 7.2-RELEASE-p3:
/boot/kernel/ath_hal.ko
...edited...
/usr/src/sys/vm/vm_pageq.c
The following files will be added as part of updating
to 7.2-RELEASE-p3:
/boot/kernel/cpuctl.ko
...edited...
/usr/src/usr.sbin/makefs/walk.c
The following files will be updated as part of updating
to 7.2-RELEASE-p3:
/.cshrc
/.profile
/COPYRIGHT
...edited...
/var/yp/Makefile.dist
freebsd7# freebsd-update install
Installing updates...
Kernel updates have been installed. Please reboot and
```

## FreeBSD Update to Upgrade from One Minor Version to Another

You've seen how CVSup and Csup can be used to update the OS and userland according to the tags in a supfile. You could easily continue this process if you wished to upgrade from FreeBSD 7.1 to FreeBSD 7.2 RELEASE. For example, your supfile would say the following (see Listing 26).

Notice we removed the date tag seen earlier. We also changed the release tag to indicate RELENG_7_2_0, which would be the same FreeBSD 7.2 shipped on CD.

It would make more sense to use RELENG_7_2 so the new system would be tracking the security branch.

It would be convenient if we could use binary upgrades via FreeBSD Update. It turns out that in this situation, we can do so. These are the basic commands:

```
freebsd-update upgrade -r 7.2-RELEASE
freebsd-update install
reboot
freebsd-update install
```

Note that this process requires plenty of free space in the `/var partition`. If you have more free space elsewhere (say in /usr), you can specific an alternative work directory for freebsd-update using the `-d` switch, e.g.,

```
freebsd-update -d /usr/db/freebsd-
update upgrade -r 7.2-RELEASE
```

Ensure the specified directory exists before starting FreeBSD Update.

In the following example, we upgrade our FreeBSD 7.1 system to FreeBSD 7.2 using FreeBSD Update. FreeBSD Update will upgrade the system to the latest point in the security branch.

As you can see, we used FreeBSD Update to bring our FreeBSD 7.1 system to the latest security update for FreeBSD 7.2. Notice we are running a GENERIC kernel again (see Listings 27a and 27b).

## STABLE: The End of the Line for a Single Version

The end of the line in the FreeBSD 7.x tree is 7.2-STABLE. The STABLE tree incorporates not only bug fixes and security patches, but upgrades that are Merged From

**Listing 27b.** FreeBSD Update from 7.1 to 7.2, continued

```
run
"/usr/sbin/freebsd-update install" again to finish
installing updates.
freebsd7# reboot
freebsd7# freebsd-update install
Installing updates...done.
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.2-RELEASE-p2 FreeBSD
7.2-RELEASE-p2 #0: Wed Jun 24 00:57:44 UTC 2009
root@i386-builder.daemonology.net:/usr/obj/usr/src/
sys/GENERIC i386
```

**Listing 28.** Supfile for FreeBSD 7-STABLE

```
*default host=INSERTYOURCHOICE.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=RELENG_7
*default delete use-rel-suffix
*default compress
src-all
```

**Listing 29.** Uname output for FreeBSD 7-STABLE

```
freebsd7# uname -a
FreeBSD freebsd7.localdomain 7.2-STABLE FreeBSD 7.2-
STABLE #2: Sat Aug 22 17:12:42 EDT 2009   root@freebsd
7.localdomain:/usr/obj/usr/src/sys/FREEBSD7 i386
```

**Listing 30.** Uname for system running FreeBSD 7-STABLE with desired kernel

```
fbsd71toS# uname -a
FreeBSD fbsd71toS.taosecurity.com 7.2-STABLE FreeBSD
7.2-STABLE #0: Sat Aug 22 23:02:30 EDT 2009   root@fb
sd71toS.taosecurity.com:/usr/obj/usr/src/sys/FREEBSD7
i386
```

**Listing 31.** Uname for system that needs to update its kernel

```
freebsd7S# uname -a
FreeBSD freebsd7S.taosecurity.com 7.2-STABLE FreeBSD
7.2-STABLE #2: Sat Aug 22 17:12:42 EDT 2009   root@fre
ebsd7.localdomain:/usr/obj/usr/src/sys/FREEBSD7 i386
```

**Listing 32.** Mounting remote /usr/src and /usr/obj using NFS

```
freebsd7S# mount -t nfs 172.16.134.130:/usr/src /usr/
src

freebsd7S# mount -t nfs 172.16.134.130:/usr/obj /usr/
obj

freebsd7S# mount
/dev/ad0s1a on / (ufs, local)
devfs on /dev (devfs, local)
/dev/ad0s1f on /home (ufs, local, soft-updates)
/dev/ad0s1g on /tmp (ufs, local, soft-updates)
/dev/ad0s1d on /usr (ufs, local, soft-updates)
/dev/ad0s1e on /var (ufs, local, soft-updates)
172.16.134.130:/usr/src on /usr/src (nfs)
172.16.134.130:/usr/obj on /usr/obj (nfs)
```

**Listing 33.** Supfile for CURRENT

```
*default host=INSERTYOURCHOICE.FreeBSD.org
*default base=/usr
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix
*default compress
src-all
```

CURRENT (aka *MFC'd*). STABLE is a constantly moving target, marked only by the date and time that an administrator uses CVSup to sync with the STABLE tree. For this reason, security advisories, such as FreeBSD-SA-09:12.bind, will list the date and time at which a STABLE branch incorporates a security fix:

```
Corrected:   2009-07-28 23:59:22 UTC
(RELENG_7, 7.2-STABLE)
```

If your STABLE is older than the date specified, your system is vulnerable. Compare that method of gauging a system's exposure to the *patch level* of running the security branch. From the same advisory:

```
        2009-07-29 00:14:14 UTC
(RELENG_7_2, 7.2-RELEASE-p3)
```

Here we also have a timestamp, but it's easier to see that 7.2-RELEASE-p3 is patched for the bind vulnerability.

For demonstration purposes, we will upgrade our FreeBSD 7.2-RELEASE-p2 system to STABLE by modifying our supfile with these contents (see Listing 28).

Next we follow the commands introduced earlier to upgrade to 7.2-STABLE. Begin with:

```
csup -g -L 2 /usr/local/etc/freebsd7-
example.supfile
```

Then continue by using a new copy of the GENERIC kernel configuration file. There may be changes introduced in STABLE that are not reflected in your own kernel configuration file.

```
cp /usr/src/sys/i386/conf/GENERIC
/usr/src/sys/i386/conf/FREEBSD7
```

Now we follow the commands we've seen earlier (see Listing 18).

When done you will be running FreeBSD 7.2-STABLE. When done our uname output appears as follows (see Listing 29).

Notice the output says 7.2-STABLE, although the CVS tag used was 7_RELENG.

## Building a Userland and Kernel on One System and Installing on Another

In the following example, we will show how to install the userland and kernel built on one system onto a second system. The *server* with the desired userland and kernel is fbsd71toS, or 172.16.134.130 (see Listing 30).

Since we are using NFS, the server has the following in `/etc/rc.conf`.

```
rpcbind_enable="YES"
nfs_server_enable="YES"
```

The server also has the following `/etc/exports` file.

```
fbsd71toS# cat /etc/exports
/usr      -alldirs
```

The *client* that will receive the new userland and kernel is freebsd7S (see Listing 31). The client has the following in `/etc/rc.conf`.

```
nfs_client_enable="YES"
```

First we mount `/usr/src` and `/usr/obj` from the server to the client using NFS (see Listing 32). Make sure we are now in `/usr/src`.

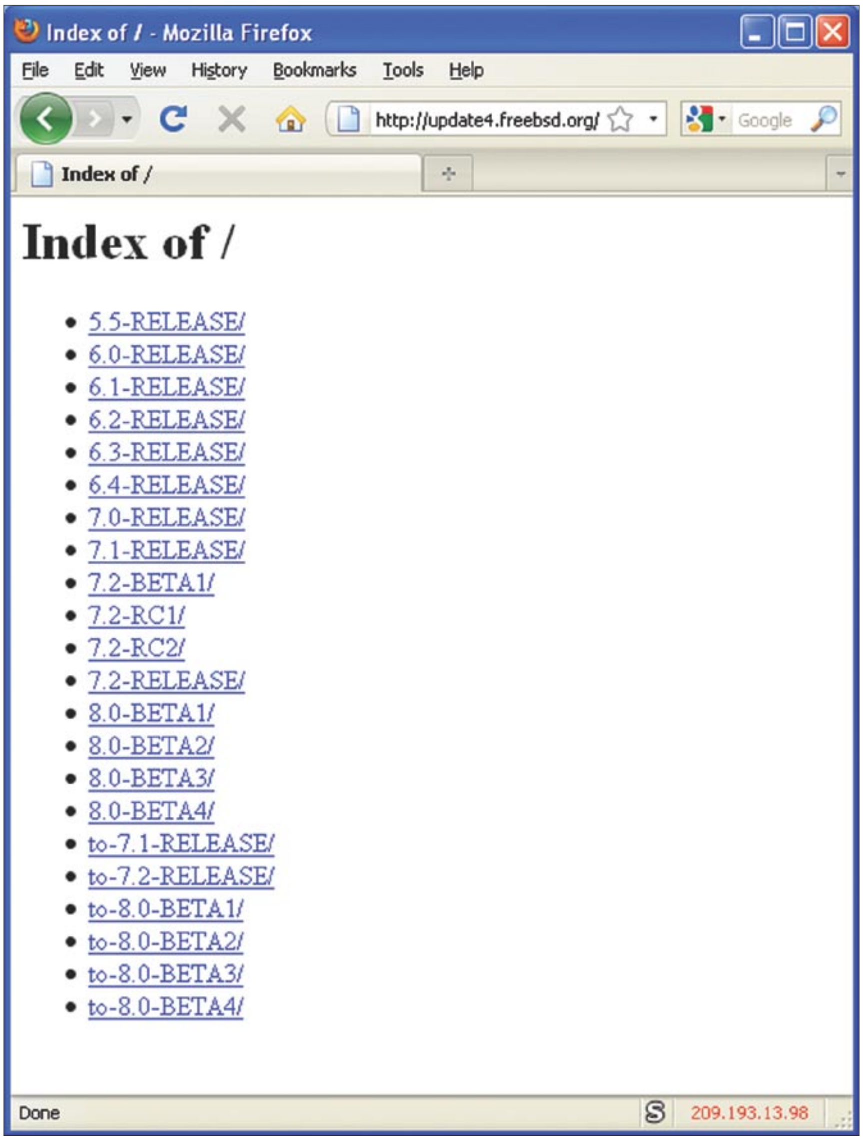
```
freebsd7S# cd /usr/src
```

At this point we can follow the instructions we saw earlier, starting as shown.

```
make installkernel KERNCONF=FREEBSD7
mergemaster -p
make installworld
mergemaster
```

Before reboot I umount the NFS mounts.

```
freebsd7S# pwd
/root
freebsd7S# umount /usr/ports
freebsd7S# umount /usr/src
freebsd7S# umount /usr/obj
reboot
```


Index of / - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://update4.freebsd.org/
Google
Index of /
Done
209.193.13.98

Index of /

- 5.5-RELEASE/
- 6.0-RELEASE/
- 6.1-RELEASE/
- 6.2-RELEASE/
- 6.3-RELEASE/
- 6.4-RELEASE/
- 7.0-RELEASE/
- 7.1-RELEASE/
- 7.2-BETA1/
- 7.2-RC1/
- 7.2-RC2/
- 7.2-RELEASE/
- 8.0-BETA1/
- 8.0-BETA2/
- 8.0-BETA3/
- 8.0-BETA4/
- to-7.1-RELEASE/
- to-7.2-RELEASE/
- to-8.0-BETA1/
- to-8.0-BETA2/
- to-8.0-BETA3/
- to-8.0-BETA4/

**Figure 3.** FreeBSD binary updates

**Listing 34a.** FreeBSD Update from 7.1 to 8.0-BETA3

```
fbsd71-to-8# uname -a
FreeBSD fbsd71-to-8.taosecurity.com 7.1-RELEASE
FreeBSD 7.1-RELEASE #0: Thu Jan  1 14:37:25 UTC 2009
root@logan.cse.buffalo.edu:/usr/obj/usr/src/sys/
GENERIC  i386
fbsd71-to-8# setenv HTTP_PROXY http://172.16.2.1:3128
fbsd71-to-8# freebsd-update upgrade -r 8.0-BETA3
Looking up update.FreeBSD.org mirrors... 3 mirrors
found.
Fetching public key from update5.FreeBSD.org... done.
Fetching metadata signature for 7.1-RELEASE from
update5.FreeBSD.org... done.
Fetching metadata index... done.
Fetching 2 metadata files... done.
Inspecting system... done.
The following components of FreeBSD seem to be
installed:
kernel/generic world/base world/dict world/doc world/
games world/info
world/manpages
The following components of FreeBSD do not seem to be
installed:
src/base src/bin src/cddl src/contrib src/crypto src/
etc src/games
src/gnu src/include src/krb5 src/lib src/libexec src/
release src/rescue
src/sbin src/secure src/share src/sys src/tools src/
ubin src/usbin
world/catpages world/proflibs
Does this look reasonable (y/n)? y
Fetching metadata signature for 8.0-BETA3 from
update5.FreeBSD.org... done.
Fetching metadata index... done.
Fetching 1 metadata patches. done.
Applying metadata patches... done.
Fetching 1 metadata files... done.
Inspecting system...  done.
Fetching files from 7.1-RELEASE for merging... done.
Preparing to download files...
...edited...
9320....9330....9340....9350 done.
Applying patches...  done.
Fetching 750 files... done.
Attempting to automatically merge changes in files...
done.
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 8.0-BETA3 have been merged into /etc/group:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/group,v 1.35.6.1 2008/11/25 02:
59:29 kensmith Exp $
+# $FreeBSD: src/etc/group,v 1.35.10.1 2009/08/03 08:
13:06 kensmith Exp $
 #
 wheel:*:0:root,analyst
 daemon:*:1:
 kmem:*:2:
 sys:*:3:
Does this look reasonable (y/n)? y
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 8.0-BETA3 have been merged into /etc/
master.passwd:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/master.passwd,v 1.40.18.1 2008/
11/25 02:59:29 kensmith Exp $
+# $FreeBSD: src/etc/master.passwd,v 1.40.22.1 2009/
08/03 08:13:06 kensmith Exp $
 #
 root:$1$kF89UpDP$s8QA1LQcpsigLx9tQVgSa1:0:0::0:0:
Charlie &:/root:/bin/csh
 toor:*:0:0::0:0:Bourne-again Superuser:/root:
 daemon:*:1:1::0:0:Owner of many system processes:
/root:/usr/sbin/nologin
 operator:*:2:5::0:0:System &:/:/usr/sbin/nologin
Does this look reasonable (y/n)? y
The following changes, which occurred between FreeBSD
7.1-RELEASE and
FreeBSD 8.0-BETA3 have been merged into /etc/passwd:
--- current version
+++ new version
@@ -1,6 +1,6 @@
-# $FreeBSD: src/etc/master.passwd,v 1.40.18.1 2008/
11/25 02:59:29 kensmith Exp $
+# $FreeBSD: src/etc/master.passwd,v 1.40.22.1 2009/
08/03 08:13:06 kensmith Exp $
 #
 root:*:0:0:Charlie &:/root:/bin/csh
 toor:*:0:0:Bourne-again Superuser:/root:
 daemon:*:1:1:Owner of many system processes:/root:
/usr/sbin/nologin
 operator:*:2:5:System &:/:/usr/sbin/nologin
Does this look reasonable (y/n)? y
The following files will be removed as part of updating
to 8.0-BETA3-p0:
/boot/kernel/ath_hal.ko
/boot/kernel/ath_hal.ko.symbols
/boot/kernel/ath_rate.ko
...edited...
The following files will be added as part of updating
to 8.0-BETA3-p0:
/boot/gptzfsboot
/boot/kernel/accf_dns.ko
...edited...
The following files will be updated as part of updating
to 8.0-BETA3-p0:
/.cshrc
/.profile
```

When done we check the uname output on the client to see that it matches the server from whom it received its kernel and userland.

That kernel matches the one on the server, so we just successfully installed a userland and kernel built on fbsd71toS onto a client, freebsd7.

## What Comes Next?

Beyond STABLE comes CURRENT, or HEAD, or `tag=.` in a supfile. CURRENT represents the next version of FreeBSD. For example, while FreeBSD 7.x is the STABLE version, CURRENT is being prepared as FreeBSD 8.0. At the time of writing, FreeBSD 8.0 is currently in BETA. Although testing the next version of FreeBSD is encouraged in order to support the project and to ensure it works on your platforms, I do not recommened running CURRENT in production.

One could use CVSup or Csup to update to CURRENT using the following supfile (see Listing 33).

However, when I want to try CURRENT, I prefer to start with a snapshot (*http://www.freebsd.org/snapshots/*) and either use the snapshot or CVSup to CURRENT from the snapshot. A snapshot is a version of FreeBSD from various branches. For example, at the time of writing, snapshots for FreeBSD 6.4-STABLE, 7.2-STABLE, and 8.0-CURRENT are posted.

## Upgrading from One Major Version to Another Major Version Using FreeBSD Update

In the final example for this article, I will show how to use binary upgrades via FreeBSD update to upgrade from FreeBSD 7.1 RELEASE to FreeBSD 8.0 BETA3. I follow the instructions posted in the announcement for BETA3 (*http://lists.freebsd.org/pipermail/freebsd-stable/2009-August/051628.html*). By setting a proxy we can have the proxy provide copies of the updates to similar systems that might also need to perform the upgrade, as well as simply use a proxy to reach the Internet.

PLEASE NOTE that you should follow the instructions provided in any release announcement and not just those in this document. For example, the test system used in this article only has cmdwatch and screen installed. This is NOT typical of a production system. It is trivial for me to manually uninstall these applications compiled for 7.x and reinstall the latest versions compiled for 8.x. Therefore, I do not show those steps here.

The official documentation describes ways to handle applications installed as packages or using the ports tree.

This can take a long time, especially at the *Inspecting system...* stages (see Listings 34a and 34b).

That's it – we're running FreeBSD 8.0 BETA3! We would have to reinstall our applications, which is covered in my related article on Keeping FreeBSD Applications Up-To-Date.

For reference, the *install* prior to the first reboot installs the new kernel. The 'install' after the first reboot installs the new userland. The *install* after the second reboot removes any old libraries used by applications that we removed (i.e., cmdwatch and screen).

**Listing 34b.** FreeBSD Update from 7.1 to 8.0-BETA3, continued

```
/COPYRIGHT
...edited...
/var/named/etc/namedb/named.root
/var/yp/Makefile.dist

fbsd71-to-8# freebsd-update install
Installing updates...
Kernel updates have been installed.  Please reboot and run
"/usr/sbin/freebsd-update install" again to finish installing updates.

fbsd71-to-8# reboot

fbsd71-to-8# freebsd-update install
Installing updates...
Completing this upgrade requires removing old shared object files.
Please rebuild all installed 3rd party software (e.g., programs
installed from the ports tree) and then run "/usr/sbin/freebsd-update
install"
again to finish installing updates.

fbsd71-to-8# pkg_info
cmdwatch-0.2.0_1    Watches the output from a command at specified
intervals
screen-4.0.3_6      A multi-screen window manager
fbsd71-to-8# cd /var/db/pkg
fbsd71-to-8# pkg_delete cmdwatch-0.2.0_1/
fbsd71-to-8# pkg_delete screen-4.0.3_6/

fbsd71-to-8# reboot

fbsd71-to-8# uname -a
FreeBSD fbsd71-to-8.taosecurity.com 8.0-BETA3 FreeBSD 8.0-BETA3 #0: Sat
Aug 22 02:36:50 UTC 2009     root@almeida.cse.buffalo.edu:/usr/obj/usr/
src/sys/GENERIC  i386
```

## Conclusion

I hope this article has helped you understand the different ways to keep a FreeBSD system up-to-date with security advisories. It is by no means comprehensive, but by following it you hopefully can judge the different ways to keep your system in sync with the latest security patches and fixes for FreeBSD.

**About the Author**

Richard Bejtlich is Director of Incident Response for General Electric. He has used FreeBSD in production since 2000. Richard operates taosecurity.blogspot.com.

# FreeBSD 8.0 RC1

**The version included on the DVD is the 8.0-RC1 and not 8.0-RELEASE. Please see below for update instructions.**

The FreeBSD Update tool can be used to upgrade from FreeBSD 8.0-RC1 to FreeBSD 8.0-RELEASE once it becomes available. To use this tool, run:

```
# freebsd-update upgrade -r 8.0-RELEASE
```

as root and follow the prompts; you may be asked to confirm that some system configuration files have been correctly updated. This will fetch and verify the files required to upgrade to 8.0-RELEASE. Next, run

```
# freebsd-update install
# shutdown -r now
```

to install updates to the FreeBSD kernel and reboot; after rebooting, run

```
# freebsd-update install
```

a second time, at which point non-kernel updates will be installed, leaving you with a system running FreeBSD 8.0-RELEASE.

## What is FreeBSD?

FreeBSD is an advanced operating system for x86 compatible (including Pentium and Athlon), amd64 compatible (including Opteron, Athlon64, and EM64T), ARM, IA-64, PowerPC, PC-98 and UltraSPARC architectures. It is derived from BSD, the version of UNIX developed at the University of California, Berkeley. It is developed and maintained by a large team of individuals. Additional platforms are in various stages of development.

## Cutting edge features

FreeBSD offers advanced networking, performance, security and compatibility features today which are still missing in other operating systems, even some of the best commercial ones. Visit *http://www.freebsd.org/features.html* to find out more.

## Powerful Internet solutions

FreeBSD makes an ideal Internet or Intranet server. It provides robust network services under the heaviest loads and uses memory efficiently to maintain good response times for thousands of simultaneous user processes.

## Advanced Embedded Platform

From mail and web appliances to routers, time servers, and wireless access points, vendors around the world rely on FreeBSD's integrated build and cross-build environments and advanced features as the foundation for their embedded products. And the Berkeley open source license lets them decide how many of their local changes they want to contribute back.

## Run a huge number of applications

With over 20,000 ported libraries and applications, FreeBSD supports applications for desktop, server, appliance, and embedded environments.

## Easy to install

FreeBSD can be installed from a variety of media including CD-ROM, DVD, or directly over the network using FTP or NFS. Get directions from *www.freebsd.org*.

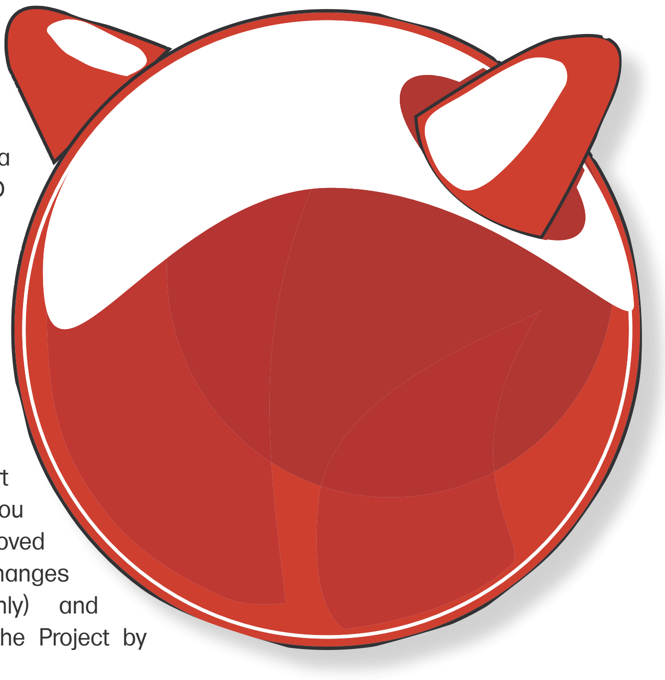
## FreeBSD is free

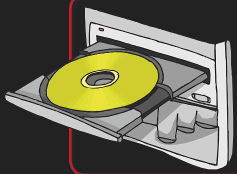
While you might expect an operating system with these features to sell for a high price, FreeBSD is available free of charge and comes with full source code.

## Contributing to FreeBSD

It is easy to contribute to FreeBSD. All you need to do is find a part of FreeBSD which you think could be improved and make those changes (carefully and cleanly) and submit that back to the Project by means of send-pr or a committer, if you know one. This could be anything from documentation to artwork to source code. See the Contributing to FreeBSD article at *http://www.freebsd.org/about.html* for more information.

Even if you are not a programmer, there are other ways to contribute to FreeBSD. The FreeBSD Foundation is a non-profit organization for which direct contributions are fully tax deductible. Please contact *board@FreeBSDFoundation.org* for more information or write to: The FreeBSD Foundation, P.O. Box 20247, Boulder, CO 80308, USA.





**If the DVD content cannot be accessed and the disc is not damaged, try to run it on at least two DVD-ROMs.**

MAGAZINE BSD 1/2010

If you have encountered any problems with the DVD, please write to: cd@software.com.pl

# Using BSD for your Studies

Edd Barrett

About four years ago I was starting my undergraduate computing degree. I knew that UNIX-like operating systems had proven themselves in the server room, but how would they fare in the lecture theatre?

At the time that I started studying, I had already been using OpenBSD for a couple of years, but was curious as to whether it was going to be feasible as an everyday study aid. In this informal and non-technical article I hope to share my findings and highlight some tools I found useful for my degree.

## Introduction

Academic computing boils down to three categories of tasks: coding, documenting and diagramming, each of which I found to require different specialised tools greatly depending upon the nature of the task. Why did I need so many tools? A modern computing degree is likely to comprise of a diverse range of subjects, usually including units for object oriented programming, web-based programming, software engineering techniques, relational databases, networking and possibly computer law/ethics. Let's take a look at what free software has to offer your BSD workstation with regards to these subject areas.

## Coding

In terms of programming languages, a BSD user is spoilt for choice. Take a look at your package collection and you will find a huge selection of programming and scripting languages at your disposal. My University primarily taught using the Java programming language from Sun Microsystems (as many Universities now do). Initially for a BSD user, this was quite awkward as Sun were very particular about how Java could be distributed and no binary packages were available for quite some time. Recently however, the OpenJDK project has been ported to OpenBSD (in ports under devel/jdk), allowing Java to be installed by simply using `'pkg_add'`. I must admit that I tended to choose C or C++ over Java if possible, as the compilers (gcc and g++) are a part of the base install.

Other languages were also a breeze to install. PHP (www/php5) could be used with the in-base apache web server shipping with OpenBSD for the web-based unit. I also found myself using several other scripting languages including Ruby (lang/ruby) and Python (lang/python) for various other tasks.

Next you can start looking into a text editor or IDE to actually write some code with. I think students are pretty much free to do as they please here and I am sure most computing students will already have a favourite text editor. I pretty much exclusively used Vim (editors/vim) for coding and found it to be mostly excellent for this purpose. A lot of my course-mates liked to use eclipse (devel/eclipse) for Java development for it's code completion features and one of my lecturers (who even ran OpenBSD on his laptop) swore by Nedit (editors/nedit). Ultimately text editors are vastly down to personal choice, so I couldn't say any editor is *the best*. My advice here is to go and play until you find one you like.

So now we have everything we need to start hacking out some code, but there's some other tools you might wish to use. I went further by using source control. A bit overkill, some might say (for a single developer project), however I found these tools essential. For those unfamiliar with the term *source control* (or sometimes *version control* or *source code management*), this term relates to tracking changes in a software project. So why would you want to do that? Ever found yourself in a *Where did I get to again*? or a *What have I changed that broke that*? situation?

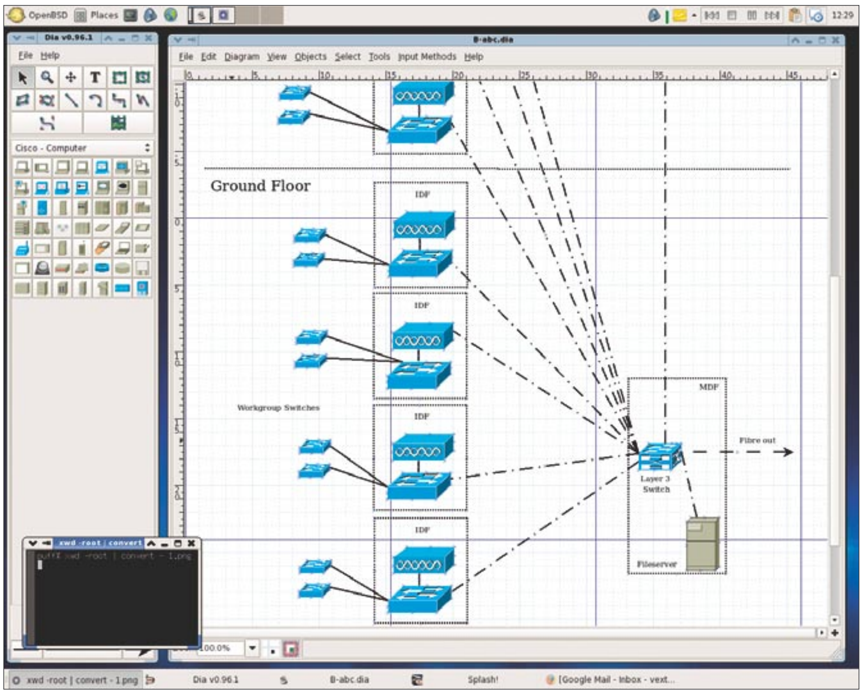
Well tools such as (but not limited to) CVS (in base install) or subversion (devel/subversion) can help answer those questions. Also if your source control server is separate from your workstation, a code checkout acts as a poor man's backup. Once you have lost a large portion of code once, for example, the bit you stayed up all night last night implementing; You will see why backups are useful. To complement source control, I recommend the Trac software package (www/trac), which comes with a source control browser, bug tracker and wiki. Very useful indeed.

## Writing Documentation

The next major task, and probably the most significant in any degree, is the task of producing write-ups. At your disposal here are two classes of software: word processors and typesetters. I am sure you are all familiar with the concept of a word processor, so we probably don't need a detailed discussion. For word processing, the popular choice amongst free software users at my Uni was OpenOffice Writer (editors/openoffice), which is a fully featured word processing package offering an interface not dissimilar to Microsoft Word. Other word processors you may wish to try are AbiWord (editors/abiword) and Kword (x11/kde/office3).

At the other end of the spectrum are typesetters, which take an entirely different approach to document generation. With a typesetter you write your document in a mark-up language which is then compiled into a viewable document. During the first year of my course I discovered the LaTeX typesetter and used it ever since for all of my assignments. Some fellow students also picked up LaTeX and found it practical too, but others disliked writing documents in markup. Think of it like Marmite; you either love it or you hate it, so try for yourself (print/texlive). Personally I like it because:

- The results look elegant and professional.
- It is easily imported into source control because the input format is textual.
- Source code listings can be directly included by file name rather than with copy and paste (and again when the source changes).



```
xwd -root | convert - 1.png
```

Figure 1. Drawing a network diagram in dia.

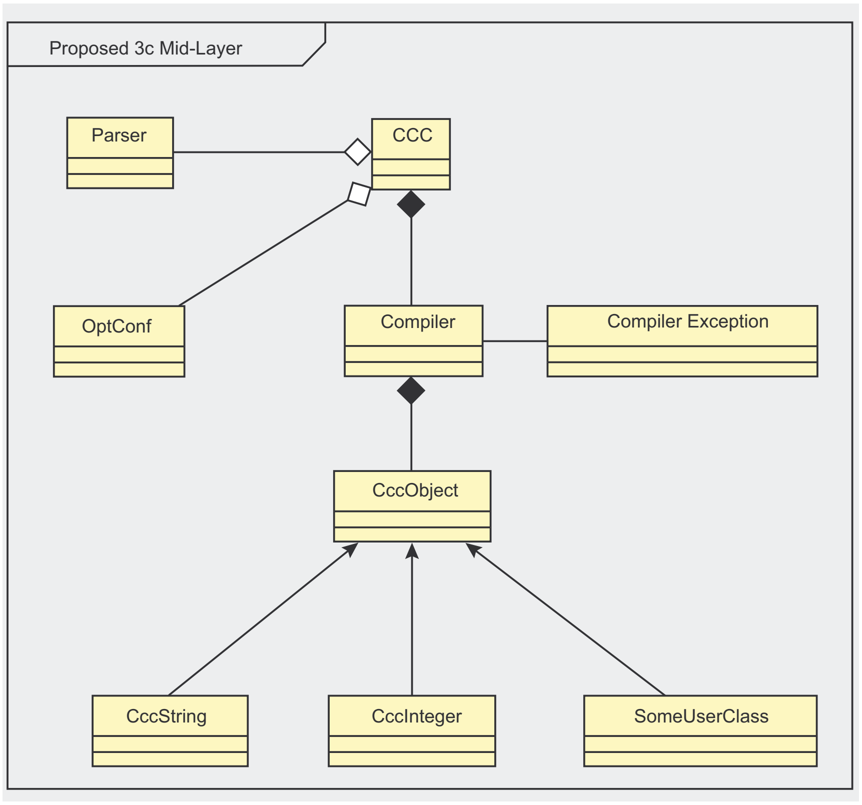


Figure 2. Bouml makes tidy UML diagrams for your write-ups.

## Drawing Diagrams

Academics really love to invent diagram notations. After half a year of University life you will find yourself knee deep in all kinds of types of diagrams. I could never cover them all in this short article, but what I can do is briefly cover the main ones which a computing student is likely to need to know.

The most complete *generic* diagram drawing program that I found was dia (graphics/dia). It is not dissimilar to Microsoft Visio and can be used for entity relationship diagrams, flow charts, logical network diagrams and much more, however you will often find that specialist tools can achieve better results (if they exist).

Probably the most commonly used diagram for academic software development is the *Unified Modelling Language* (UML) class diagram. A bunch of people at the *Object Model Group* (OMG) devised a set diagrams which could be used to visually model aspects of software projects. The UML class diagram is commonly used to model relationships between classes in object oriented languages like Java and C++. My personal choice for such a diagram is Bouml (devel/bouml), which can model many different UML diagrams and also output SVG/PNG graphics for inclusion in your write-ups. The user interface is a little quirky at first, but once you get used to it, you can slap up class diagrams very quickly and neatly.

For flow charts and control flow graphs, I found the best option to be graphviz (math/graphviz). Like LaTeX, graphviz uses a markup language which is compiled. One valuable realisation that I made during my studies, is that one of the benefits of these text based mark-up languages is that they are easily auto-generated by other pieces of software. Using for example the classic computer science *off you go and implement a linked list* task; whilst you are traversing your linked list, you could generate graphviz code. The code when compiled could generate a directed graph, showing the nodes of your linked list, then this could be included as a part of your report. If you don't much like the idea of drawing diagrams in code, then trusty old dia is probably going to be a better choice for this kind of task.

Often you will be required to carry out experiments which require the results of which to be plotted on a graph for the purposes of observing trends. A quick graph is very easily generated using OpenOffice calc, and is usually the way I would choose to draw a small graph. Having said that sometimes it is not convenient to work in this way. I once found myself with a large amount of data, which had been collected automatically by a Java program. Inputting this vast chunk of data would have been rather soul destroying and in such a situation you may wish to look into a solution using gnuplot (math/gnuplot). With a little research I managed to quickly adapt my program to generate gnuplot source code, which I could compile and include in my LaTeX documents.

## Conclusion

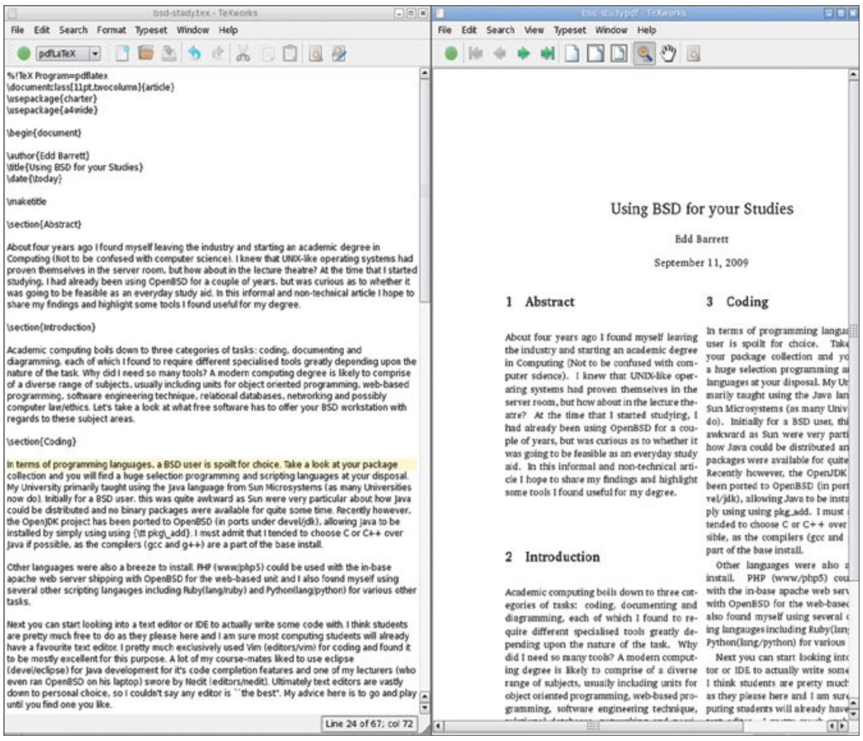
All in all, a BSD desktop is perfectly acceptable as a study workhorse. If you did want to go down this path, I would recommend using a stable release and not a developer snapshot, as things can get awkward at times when really, you just want to finish that darned assignment (I learnt that the hard way).

I think that any form of open-source software offers a large learning opportunity to students. The code for such software is freely available for study and citation. I benefited from this with my dissertation, which was in the field of compiler design. I frequently found myself referring the reader to various open source language parser grammars and virtual machine implementations, which otherwise would not have been available.

Ultimately you probably will get bitten by the Windows world at some point. A large number of commercial specialist tools do not run on BSD and there are no good open source alternatives. In these cases you just have to find a windows box and deal with it. Looking back, I think I only found myself in this situation a couple of times and you can always dual boot BSD and Windows if you wish.

If your University has UNIX labs, you may find some of the aforementioned tools there readily available to you. We were lucky enough to have access to a lab of Sun thin clients, running Solaris 10 and the N1 grid engine. We spent a lot of time in this lab, exploring, tinkering and generally expanding our UNIX knowledge. We even started a Bournemouth Uni UNIX user group which met on a monthly basis to hold talks and demonstrations on the subject of BSD, UNIX, Linux etc. Resources like these are a great way to learn and meet like minded-students (and staff), so you may wish to see if your Uni has one.

That about covers my key findings in my accademic life so far. Of course I could have gone into far more detail with many of these topics, but that would be out of the scope of this article. I hope you enjoyed the read, even if it was quite informal.


bsd-study.tex - TeXworks
File Edit Search Format Typeset Window Help
pdfLaTeX
Line 24 of 67; col 72
bsd-study.pdf - TeXworks
File Edit Search View Typeset Window Help

```
%!TeX Program=pdflatex
\documentclass[11pt,twocolumn]{article}
\usepackage{charter}
\usepackage{a4wide}

\begin{document}

\author{Edd Barrett}
\title{Using BSD for your Studies}
\date{\today}

\maketitle

\section{Abstract}

About four years ago I found myself leaving the industry and starting an academic degree in
Computing (Not to be confused with computer science). I knew that UNIX-like operating systems had
proven themselves in the server room, but how about in the lecture theatre? At the time that I started
studying, I had already been using OpenBSD for a couple of years, but was curious as to whether it
was going to be feasible as an everyday study aid. In this informal and non-technical article I hope to
share my findings and highlight some tools I found useful for my degree.

\section{Introduction}

Academic computing boils down to three categories of tasks: coding, documenting and
diagramming, each of which I found to require different specialised tools greatly depending upon the
nature of the task. Why did I need so many tools? A modern computing degree is likely to comprise
of a diverse range of subjects, usually including units for object oriented programming, web-based
programming, software engineering technique, relational databases, networking and possibly
computer law/ethics. Let's take a look at what free software has to offer your BSD workstation with
regards to these subject areas.

\section{Coding}

In terms of programming languages, a BSD user is spoilt for choice. Take a look at your package
collection and you will find a huge selection programming and scripting languages at your disposal.
My University primarily taught using the Java language from Sun Microsystems (as many Universities
now do). Initially for a BSD user, this was quite awkward as Sun were very particular about how Java
could be distributed and no binary packages were available for quite some time. Recently however,
the OpenJDK project has been ported to OpenBSD (in ports under devel/jdk), allowing Java to be
installed by simply using using {\tt pkg\_add}. I must admit that I tended to choose C or C++ over
Java if possible, as the compilers (gcc and g++) are a part of the base install.

Other languages were also a breeze to install. PHP (www/php5) could be used with the in-base
apache web server shipping with OpenBSD for the web-based unit and I also found myself using
several other scripting langauges including Ruby(lang/ruby) and Python(lang/python) for various other
tasks.

Next you can start looking into a text editor or IDE to actually write some code with. I think students
are pretty much free to do as they please here and I am sure most computing students will already
have a favourite text editor. I pretty much exclusively used Vim (editors/vim) for coding and found it
to be mostly excellent for this purpose. A lot of my course-mates liked to use eclipse
(devel/eclipse) for Java development for it's code completion features and one of my lecturers (who
even ran OpenBSD on his laptop) swore by Nedit (editors/nedit). Ultimately text editors are vastly
down to personal choice, so I couldn't say any editor is ``the best''. My advice here is to go and play
until you find one you like.
```

Using BSD for your Studies

Edd Barrett

September 11, 2009

1 Abstract

About four years ago I found myself leaving the industry and starting an academic degree in Computing (Not to be confused with computer science). I knew that UNIX-like operating systems had proven themselves in the server room, but how about in the lecture theatre? At the time that I started studying, I had already been using OpenBSD for a couple of years, but was curious as to whether it was going to be feasible as an everyday study aid. In this informal and non-technical article I hope to share my findings and highlight some tools I found useful for my degree.

2 Introduction

3 Coding

**Figure 3.** TeXworks (print/texworks).

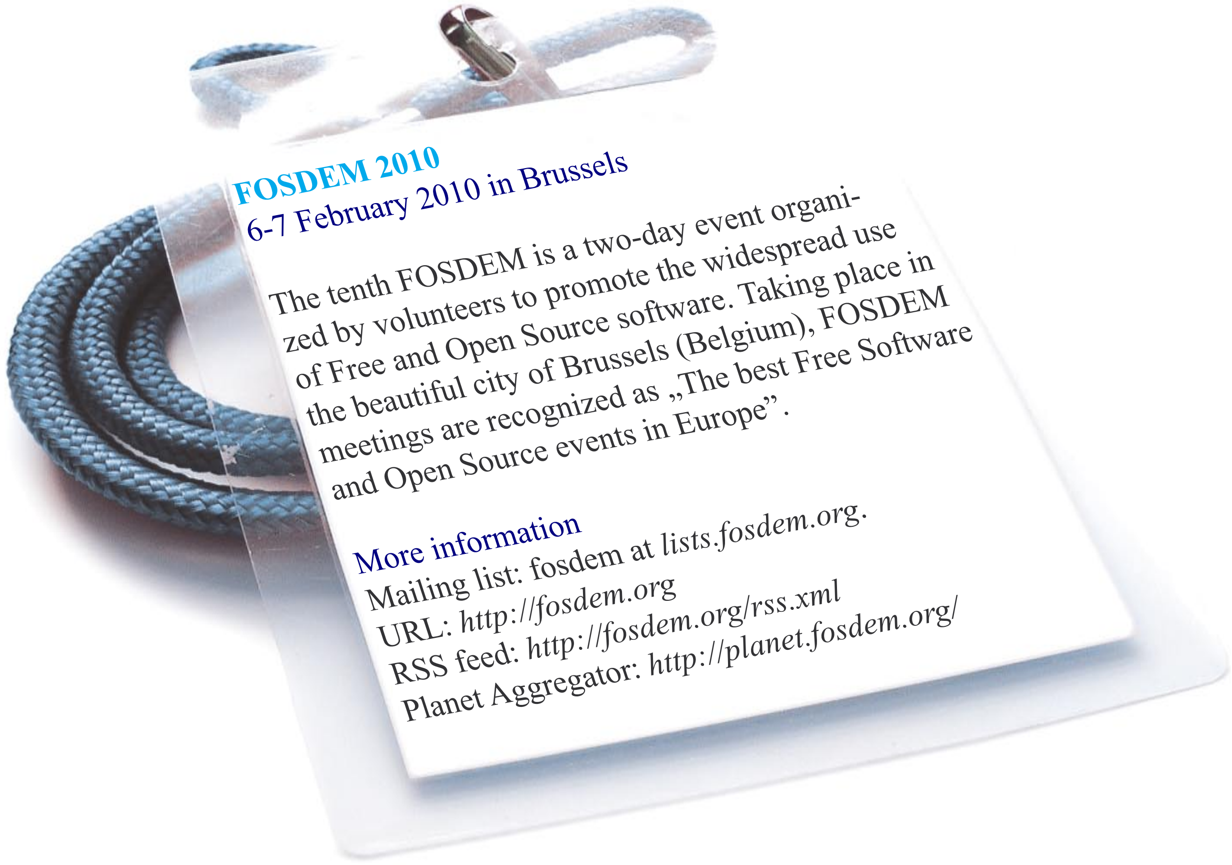
## About the Author

Edd has just graduated from Bournemouth University in the UK and is currently enrolled as a PhD student for the University of Kent, where his research area is reverse engineering. In his spare time Edd likes to read early sci-fi books and has a love for metal (music).

E-mail: *eddbarrett@googlemail.com*

WWW: *http://students.dec.bmth.ac.uk/ebarrett*

# Events!



## FOSDEM 2010

6-7 February 2010 in Brussels

The tenth FOSDEM is a two-day event organized by volunteers to promote the widespread use of Free and Open Source software. Taking place in the beautiful city of Brussels (Belgium), FOSDEM meetings are recognized as „The best Free Software and Open Source events in Europe".

More information

Mailing list: fosdem at *lists.fosdem.org*.

URL: *http://fosdem.org*

RSS feed: *http://fosdem.org/rss.xml*

Planet Aggregator: *http://planet.fosdem.org/*

## *8th USENIX Conference on File and Storage Technologies*

*February 23–26, 2010, San Jose, CA*

*FAST '10 brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. Meet with premier storage system researchers and practitioners for ground-breaking file and storage information!*

*More information:*

*http://www.usenix.org/events/fast10/*

Stay up to date with upcoming BSD events!
Visit www.bsdevents.net regularly!

# The FreeBSD Chatterbox

Eric Vintimilla

Day in and day out, your FreeBSD sits there quietly, processing its workload. It never complains or asks for any favors, but what would it say if it could talk?

The answer to that question is easy. It will say whatever you want it to. With Festival, your FreeBSD box can be more talkative than your last date!

## What is festival?

Festival is a system that gives users text to speech capabilities through the shell level, a Scheme command interpreter, a C++ library, Java, or an Emacs interface. It offers various voices, including different languages and accents, such as English (British and American) and Spanish. Furthermore, more voices have be created by Carnegie Mellon's FestVox project (*http://festvox.org*) and it is relatively straightforward to create your own voice library. Festival has two main modes: command mode, where it can read input from files or interactively, and tts (text-to-speech), where text input is rendered as speech.

## Installing and setting up festival

Installing Festival is quick and easy to do. Find its directory in your ports tree, and install it with portinstall.

```
evvm# cd /usr/ports/audio/festival
evvm# portinstall -P festival
```

A few dependencies will be installed, but the process should not take that long. Now, enter Festival's command line and run a test (see Listing 1).

**Listing 1.** The Festival Command Interpreter

```
evvm# festival
WARNING
No default voice found in ("/usr/local/share/festival/lib/voices/")
either no voices unpacked or voice-path is wrong
Scheme interpreter will work, but there is no voice to speak with.
WARNING
Festival Speech Synthesis System 1.96:beta July 2004
Copyright (C) University of Edinburgh, 1996-2004. All rights reserved.
For details type '(festival_warranty)'
festival>
festival> (SayText "FreeBSD Rocks")
-=-=-=-=-=- EST Error -=-=-=-=-=-
{FND} Feature Token_Method not defined
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
festival>
evvm# ^D
```

It looks like there are no voice databases on our FreeBSD box, so we will have to find some, but where? Luckily, Carnegie Mellon's Festvox voices can also be found in the ports tree (Listing 2).

There should now be a `us1_mbrola` directory under `/usr/local/share/festival/lib/voices/english/`. Now we can test Festival!

You should have heard your FreeBSD box claiming its awesomeness.

## What should your freebsd box say?

Now, you can have your FreeBSD machine say whatever you want, but using Festival's command line interface may not be ideal. Luckily, Festival can receive input from the shell using its `--tts` switch. It can be used to read files or to speak the output of other processes. For example, to read an input file, you would use the following command:

```
evvm#  festival --tts inputfile.txt
```

To make Festival speak process output, you can just use a pipe:

```
evvm#  date | festival --tts
```

You can go one step further, to make the text-to-speech process even easier. We can create a shell script to allow us to pass in any text to Festival.

```
   evvm# touch speak
evvm# chmod u+x speak
   evvm# nano speak
#!/usr/local/bin/bash
fest=$(/usr/bin/which festival)
/bin/echo $1 | $fest -tts
```

Now, you can use this script to make FreeBSD say whatever you want:

```
   evvm# ./speak "Hello!"
evvm# ./speak "$(date)"
```

While having a text-to-speech system may not be as useful as setting up jails or a wireless access point, it can still be used to make your machine more interesting. First, you can set up your computer to greet you every time you log in. If you use bash, just edit your `~/.bash_profile` file, and add the following to the end of it:

```
   sh ~/scripts/speak "Hello
$(whoami)!"
```

Now, every time you log in, your computer will greet you, using your username. You can also have Festival read your mail.

```
   evvm# ~/scripts/ speak "$(cat /var/
mail/$(whoami))"
```

You can also have your FreeBSD speak reminders to you.

```
   evvm# echo "~/scripts/speak 'You
need to buy milk'" | at 15:00
```

Your FreeBSD box can read logs or other documents to you. You can even be mischievous and play tricks on roommates. When you are not home, SSH into your system, and try something like:

```
  evvm# ~/scripts/speak "I'm watching
you."
```

## Summary

Using Festival on your FreeBSD machine can be both useful and fun. You can save yourself reading time and multitask while you listen to text. There are so many things your system can read to you, including logs, emails, alerts, and reminders. The possibilities are practically endless.

**Listing 2.** Voice Packs for Festival

```
evvm# ls /usr/ports/audio/ | grep festvox
festvox-abc
festvox-aec
festvox-czech
festvox-don
festvox-el11
festvox-hvs
festvox-jph
festvox-kal16
festvox-kal8
festvox-ked16
festvox-ked8
festvox-lp
festvox-mwm
festvox-ogirab
festvox-pc
festvox-rab16
festvox-rab8
festvox-tll
festvox-us1-mbrola
festvox-us2-mbrola
festvox-us3-mbrola
evvm# cd /usr/ports/audio/festvox-us1-mbrola/
evvm# portinstall -P
```

**Listing 3.** Testing Festival

```
evvm# festival
Festival Speech Synthesis System 1.96:beta July 2004
Copyright (C) University of Edinburgh, 1996-2004. All rights reserved.
For details type '(festival_warranty)'
festival> (SayText "FreeBSD Rocks!")
#<Utterance 0x28925c40>
festival>
```

# Encrypting the FreeBSD root file system

Jacques Manukyan

Systems are only as secure as you make them. Thankfully, FreeBSD offers an excellent range of tools and mechanisms to insure that all your security needs are met.

No matter how much time you spend securing your operating system, if your workstation or server is physically stolen, you can assure that your sensitive data will be accessed by someone other than you. One way to thwart this type of attack is to encrypt your root and other file systems. One tool you can use in FreeBSD to encrypt your file systems, and specifically, to encrypt your root file system, is to use the cryptographic class called geli.

Keep in mind that security doesn't end simply because you encrypted your file systems. When your server or workstation is up and running, your root and other file systems are mounted and decrypted. This provides the potential for your sensitive data and information to be stolen if your workstation or server is connected to a network since an attacker could still breach your operating system and view or copy your data. It is up to you, the system administrator, to diligently continue to protect your operating system even after encrypting your root and other file systems.

## Geli, a different kind of disk encryption

Starting with FreeBSD 6.0, a new cryptographic GEOM class was made available called GELI. *GELI* differs greatly from *GDBE*, the traditional disk encryption system written for FreeBSD and it was initially introduced in FreeBSD version 5.0.

The most important features of geli are as follows:

- Utilizes the crypto(9) framework, so when there is crypto hardware available, geli will make use of it automatically.
- Supports multiple cryptographic algorithms. As of FreeBSD 7.2, geli supports AES, Blowfish, Camellia, and 3DES.
- Can optionally perform data authentication and integrity verification utilizing one of the following algorithms: HMAC/MD5, HMAC/SHA1, HMAC/RIPEMD160, HMAC/SHA256, HMAC/SHA384 or HMAC/SHA512.
- It is fast – geli performs simple sector-to-sector encryption.

## Considerations and Preparatory Work

Before encryption your root file system, you must first determine the authentication mechanism which decrypts the geli encrypted file system.

Geli currently supports the following mechanisms:

- Passphrase
- Keyfile
- Keyfile plus passphrase

The passphrase mechanism is by far the simplest to utilize and setup. All you need to get a geli encrypted root file system utilizing a passphrase is a DVD drive and a DVD media of the FreeBSD operating system.

### To keyfile or not to keyfile

If you are going to utilize just a keyfile without a passphrase, you must first consider a few things. The keyfile must be kept secure from the world as anyone who can get their hands on the keyfile can decrypt your file systems.

If you decide to save the keyfile locally on the boot drive that you are encrypting, then anytime the server is rebooted, it automatically decrypts the root file system. This basically defeats the purpose of encryption simply because if anyone can decrypt the drive, then it is just as vulnerable as a non encrypted drive.

If you decide to utilize a keyfile, you must move the keyfile to a removable media and keep it secure. For example, you would put the keyfile on a bootable CD or DVD media or a bootable USB device. You would then boot up the system off of the DVD or CD Media, or USB device. The DVD, CD, or USB device will load the kernel, and then it will decrypt and mount the root file system. Without the DVD, CD, or USB device, you would not be able to boot up your root file system.

To reiterate, here are the basic requirements for setting up a geli encrypted root file system utilizing the different decryption mechanisms:

### Passphrase

- DVD drive
- DVD media of the FreeBSD operating system

### Keyfile / Keyfile plus passphrase

- Writable CD or DVD drive with a Blank CDR or DVDR media (method 1)
- USB port with a USB thumb drive (method 2)
- A working computer with FreeBSD installed

## Encrypting a FreeBSD root file system utilizing only a passphrase

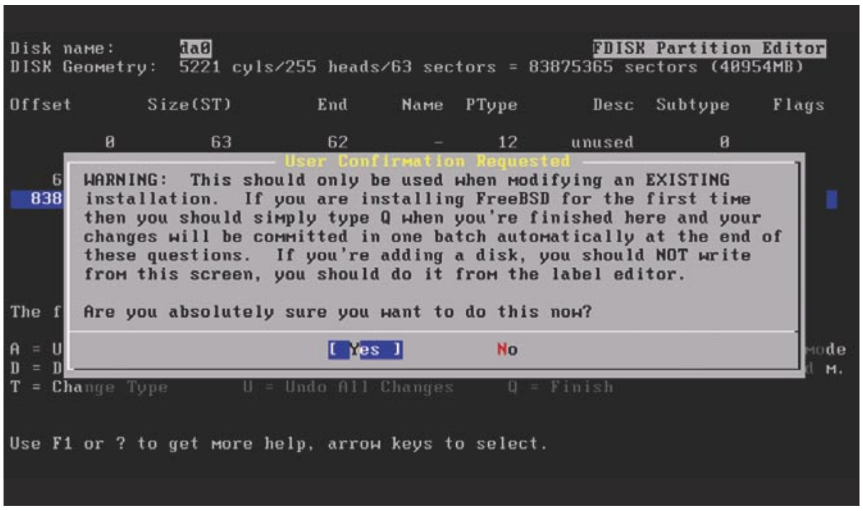
To start the installation process, you must boot up your server with the FreeBSD operating system DVD media. For the below examples, I am utilizing the FreeBSD 7.2 DVD image found on the FreeBSD Project website (*http://www.freebsd.org/*). I'm also utilizing only one 40 GB SCSI hard disk in my workstation.

Once the FreeBSD DVD media starts the installer, select the *Fixit* option from the menu. Then select option number 2 to use the *live* file system from the DVD media. This will drop you into a shell.

We want to create two slices on our primary hard disk. The boot directory, where the programs and configuration files used during the operating system bootsrap are located, cannot be on an encrypted file system.

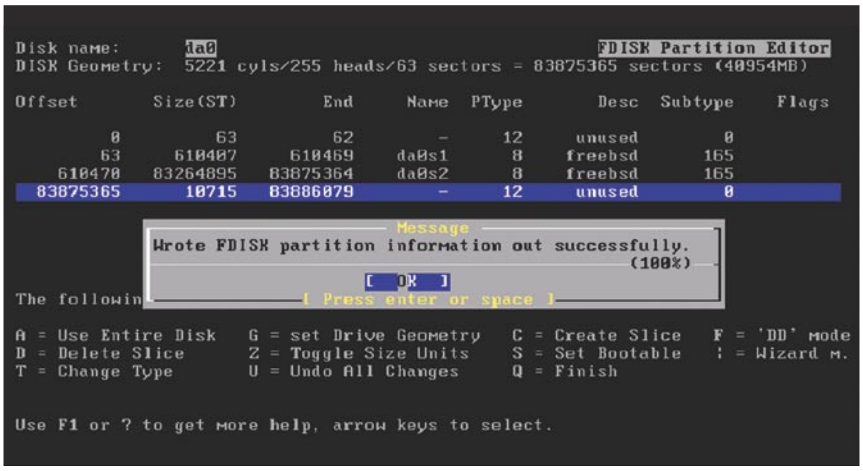
The kernel will load from the first slice and in turn, geli will load as well. Geli will then decrypt the second slice and boot up the root and other file systems located on the partitions on slice two.

Before creating the two slices, we need to determine how much space we want to utilize for the boot slice. I usually use a 300 MB slice size for the boot slice. I then dedicate the rest of the space for the second, operating system,



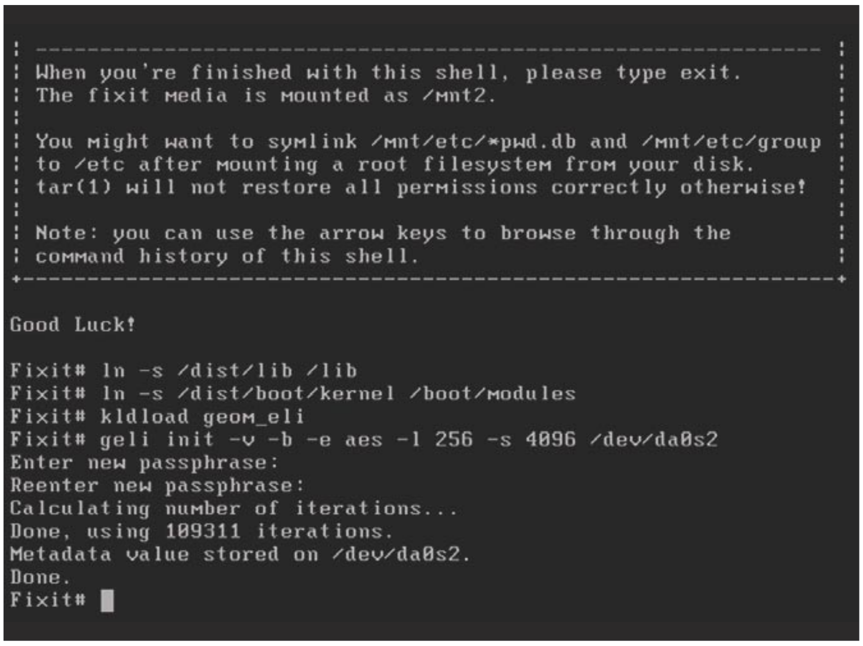
```
Disk name:      da0                                    FDISK Partition Editor
DISK Geometry:  5221 cyls/255 heads/63 sectors = 83875365 sectors (40954MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

         0         63         62        -     12     unused        0
         6 ---------- User Confirmation Requested ----------
       838  WARNING:  This should only be used when modifying an EXISTING
            installation.  If you are installing FreeBSD for the first time
            then you should simply type Q when you're finished here and your
            changes will be committed in one batch automatically at the end of
            these questions.  If you're adding a disk, you should NOT write
            from this screen, you should do it from the label editor.

The f       Are you absolutely sure you want to do this now?

A = U                        [ Yes ]            No                      mode
D = D                                                                   d M.
T = Change Type        U = Undo All Changes      Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

**Figure 1.** fdisk 1



```
Disk name:      da0                                    FDISK Partition Editor
DISK Geometry:  5221 cyls/255 heads/63 sectors = 83875365 sectors (40954MB)

Offset       Size(ST)        End     Name  PType       Desc  Subtype    Flags

         0         63         62        -     12     unused        0
        63     610407     610469    da0s1      8    freebsd      165
    610470   83264895   83875364    da0s2      8    freebsd      165
  83875365      10715   83886079        -     12     unused        0

                  ------------------- Message -------------------
                  Wrote FDISK partition information out successfully.
                                                              (100%)
                  -------------------------------------------------
                                        [  OK  ]
The followin      ------------[ Press enter or space ]------------

A = Use Entire Disk     G = set Drive Geometry   C = Create Slice    F = 'DD' mode
D = Delete Slice        Z = Toggle Size Units    S = Set Bootable    | = Wizard m.
T = Change Type         U = Undo All Changes     Q = Finish

Use F1 or ? to get more help, arrow keys to select.
```

**Figure 2.** fdisk



```
| ----------------------------------------------------------------- |
| When you're finished with this shell, please type exit.           |
| The fixit media is mounted as /mnt2.                              |
|                                                                   |
| You might want to symlink /mnt/etc/*pwd.db and /mnt/etc/group     |
| to /etc after mounting a root filesystem from your disk.          |
| tar(1) will not restore all permissions correctly otherwise!      |
|                                                                   |
| Note: you can use the arrow keys to browse through the            |
| command history of this shell.                                    |
+-------------------------------------------------------------------+

Good Luck!

Fixit# ln -s /dist/lib /lib
Fixit# ln -s /dist/boot/kernel /boot/modules
Fixit# kldload geom_eli
Fixit# geli init -v -b -e aes -l 256 -s 4096 /dev/da0s2
Enter new passphrase:
Reenter new passphrase:
Calculating number of iterations...
Done, using 109311 iterations.
Metadata value stored on /dev/da0s2.
Done.
Fixit# █
```

**Figure 3.** Geliinit

slice. The boot slice of 300 MB is large enough to accommodate my kernel and additional items. At a bare minimum however, I recommend at least a 150 MB slice. Please be aware that if you do not give yourself enough room on your boot slice, you may run into disk constraint problems in the future when upgrading your server.

At the command prompt, you would use the fdisk utility to create your two slices. Please refer to the fdisk(8) man page for more information on utilizing fdisk.

If you are unfamiliar with the fdisk command or utility, you can use the guided sysinstall utility to create your two slices. To do this, type exit at the command prompt to go back to the sysinstall menu. Next, go back to the sysinstall main menu. From there, select configure. Now scroll down and select the Fdisk option. This will bring you to the *FDISK Partition Editor.*

Now let's create our two slices using the *FDISK Partition Editor*. Hit `c` for Create Slice and specify a value of 300M. Set the type to 165, which is the default, to create a native FreeBSD slice. Now hit `c` again to create the second slice. Select the default value given for the size which should already be set to the maximum available space on the drive left. Then set the type to 165 again. Now hit `w` to write the changes to the disk. You will then be asked if you want to install a boot manager. Install the FreeBSD Boot Manager and then hit ok. FDISK should have completed successfully now (Figure 2). Hit `Q` to exit the *FISK Partition Editor.*

Now that we have created the two slices, go back to the sysinstall main menu and go back to the Fixit shell.

Once at the Fixit shell, we want to first setup our editor. To do this, issue the following command:

```
# export EDITOR=/dist/usr/bin/vi
```

Now that our default editor is setup, we want to create two symlinks so that we can load the geli kernel module. Issue the following commands at the command prompt:

```
# ln -s /dist/lib /lib
# ln -s /dist/boot/kernel /boot/
modules
```

Now we can load the geli module. Issue the following command to load the geli module:

```
# kldload geom_eli
```

Now that geli is loaded, let us encrypt our second slice. My second slice is called `da0s2` so I will issue the following command:

```
# geli init -v -b -e aes -l 256 -s
4096 /dev/da0s2
```

The `-b` option specifies that I want geli to ask for the passphrase on boot. The `-e` and `-l` options specify the encryption algorithm and key length. In this example, I want to use AES 256. The `-s` option specifies the sector size. I chose a large sector size to increase performance. Please refer to the geli(8) man page for detailed configuration information.

When issuing the `geli init` command, you will be asked to specify a passphrase twice (Figure 3).

Now that our slice is encrypted, we need to attach our slice so that we can start using it. Issue the following command to attach the slice:

```
# geli attach /dev/da0s2
```

You will be asked for your passphrase before the slice is attached. Once the slice is attached or decrypted, it will be made available to the operating system via a new device. In this case, our new device is called `/dev/da0s2.eli`.

We now create a single partition on our first or boot slice. Issue the following command:

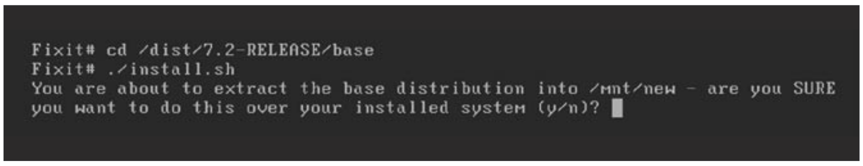
```
# bsdlabel -w /dev/da0s1
```

This will create a device called `/dev/da0s1a`.

We need to now create our partitions on the second slice. Issue the following commands:

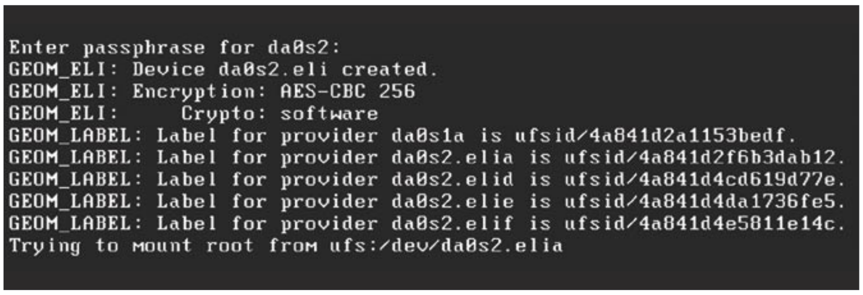
**Listing 1.** Partitioning scheme

```
# /dev/da0s2.eli:
8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:       2G        *    4.2BSD        0     0
  b:       2G        *      swap
  c: 10408111        0    unused        0     0        # "raw" part, don't
edit
  d:       4G        *    4.2BSD        0     0
  e:       2G        *    4.2BSD        0     0
  f:        *        *    4.2BSD        0     0
```



```
Fixit# cd /dist/7.2-RELEASE/base
Fixit# ./install.sh
You are about to extract the base distribution into /mnt/new - are you SURE
you want to do this over your installed system (y/n)?
```

**Figure 4.** Installbase



```
Enter passphrase for da0s2:
GEOM_ELI: Device da0s2.eli created.
GEOM_ELI: Encryption: AES-CBC 256
GEOM_ELI:     Crypto: software
GEOM_LABEL: Label for provider da0s1a is ufsid/4a841d2a1153bedf.
GEOM_LABEL: Label for provider da0s2.elia is ufsid/4a841d2f6b3dab12.
GEOM_LABEL: Label for provider da0s2.elid is ufsid/4a841d4cd619d77e.
GEOM_LABEL: Label for provider da0s2.elie is ufsid/4a841d4da1736fe5.
GEOM_LABEL: Label for provider da0s2.elif is ufsid/4a841d4e5811e14c.
Trying to mount root from ufs:/dev/da0s2.elia
```

**Figure 5.** Bootup

```
# bsdlabel -w /dev/da0s2.eli
# bsdlabel -e /dev/da0s2.eli
```

You must now decide how you want to partition your operating system. For this example, I'm going to use the following scheme (see Listing 1).

In this example, I'm going to use partition `a` as my root `/` file system and I'm allocating 2 GB of space for it. Partition `b` is my swap and I'm allocating 2 GB to it. Partition `d` will be my `/var` directory and I'm allocating 4 GB to it. Partition `e` will be my `/tmp` directory and I'm allocating 2 GB to it. And finally, partition `f` will be my `/usr` directory and I'm allocating all available space to it.

## Note

I'm leaving the offset values at `*` which will make bsdlabel handle the values automatically.

Now save and exit the bsdlabel screen. We can now create our file systems using the newfs command. Issue the following commands (see Listing 2).

As you noticed, I issued the `-O 2` option to newfs. This creates a UFS 2 file system rather than a UFS 1 file system. Please refer to the newfs(8) man page for more options.

Now that we created our new file systems, we want to mount them. To do this, we must first create our mount points. Issue the following commands at the command prompt:

```
# mkdir /mnt/boot
# mkdir /mnt/new
```

We can now mount our file systems by issuing the following commands:

```
# mount /dev/da0s1a /mnt/boot
# mount /dev/da0s2.elia /mnt/new
```

Now that our root file system is mounted to `/mnt/new`, we need to create the directory structure to mount our other file systems. Issue the following commands:

```
# mkdir /mnt/new/var
# mkdir /mnt/new/tmp
# mkdir /mnt/new/usr
```

Now we can go ahead and mount our other file systems that we created:

```
# mount /dev/da0s2.elid /mnt/new/var
# mount /dev/da0s2.elif /mnt/new/usr
```

Note that I am not mounting the `/dev/da0s2.elie` device or our `/tmp` file system. The reason behind this is that during the operating system installation that we are going to be doing, no information is going to be written to that file system.

Now that our file systems are mounted, we can go ahead and install the operating system. Before we proceed with the installation, we need to specify the location we are going to install the operating system files into. To do this, issue the following command:

```
# export DESTDIR=/mnt/new
```

We can no install the operating system. Issue the following commands to start the installation process:

```
# cd /dist/7.2-RELEASE/base
# ./install.sh
```

You will be asked to confirm that you want to install the base into `/mnt/new`. Just type `y` and hit enter (Figure 4). This step will take a few minutes depending on the speed of your DVD drive and hard drive.

If you are using a different version of FreeBSD, replace 7.2-RELEASE with the name corresponding with the version you are installing.

The base operating system will now install onto our encrypted slice and partitions; which are mounted and decrypted.

If you optionally want to install the man pages, you can issue the following commands:

```
# cd /dist/7.2-RELEASE/manpages
# ./install.sh
```

Likewise, if you want to optionally install the FreeBSD docs, you can issue the following commands:

```
# cd /dist/7.2-RELEASE/doc
# ./install.sh
```

Now that our base operating system is installed, we want to install our kernel. Issue the following commands to install the kernel:

```
# cd /dist/7.2-RELEASE/kernels
# ./install.sh generic
```

The kernel will install itself into the `/mnt/new/boot/GENERIC` directory. We now want to move the kernel to its proper location. Issue the following commands at the command prompt:

```
# rmdir /mnt/new/boot/kernel
# mv /mnt/new/boot/GENERIC /mnt/new/
boot/kernel
```

We now want to make sure that geli loads during the boot-up process so that it can decrypt our encrypted root file system. To do this, issue the following command:

**Listing 2.** File system creation

```
# newfs -O 2 /dev/da0s1a
# newfs -O 2 /dev/da0s2.elia
# newfs -O 2 /dev/da0s2.elid
# newfs -O 2 /dev/da0s2.elie
# newfs -O 2 /dev/da0s2.elif
```

**Listing 3.** Sample fstab entries

```
# Device              Mountpoint      FStype  Options         Dump
Pass#


/dev/da0s2.elib       none            swap    sw              0        0
/dev/da0s2.elia       /               ufs     rw              1        1
/dev/da0s2.elid       /var            ufs     rw              2        2
/dev/da0s2.elie       /tmp            ufs     rw              2        2
/dev/da0s2.elif       /usr            ufs     rw              2        2
```

```
# echo geom_eli_load=\"YES\" > /mnt/
new/boot/loader.conf
```

Geli and kbdmux, the keyboard multiplexer driver, seem to have problems working together. For geli to work properly, we need to disable the kbdmux driver. We can do this by issuing the following command at the command prompt:

```
# echo hint.kbdmux.0.disabled=\"1\"
>> /mnt/new/boot/device.hints
```

Now that we have our boot directory setup on the `/mnt/new` file system, we want to copy it over to our boot slice.

We want to make sure that we preserve all file permissions, file modes, user IDs, and group IDs. To do this, issue the `cp` command with the `p` option and the `R` option, for recursion, as follows:

```
# cp -Rp /mnt/new/boot /mnt/boot/
```

We now want to create a proper fstab so that our file system is mounted properly on boot-up. Issue the following command at the command prompt:

```
# vi /mnt/new/etc/fstab
```

Based on the partitions I created, I would create the following entries within the fstab file (see Listing 3).

You can specify additional options or make changes to the above entries based on your needs. Please reference the fstab(5) man page for more information.

Now that we created an fstab, we want to copy it over to the boot slice. To do this, issue the following commands:

```
# mkdir /mnt/boot/etc
# cp /mnt/new/etc/fstab /mnt/boot/
etc/fstab
```

We are now finished with setting up our system. We can unmount our file systems and then reboot our server. Issue the following commands:

```
# umount /mnt/boot
# umount /mnt/new/var
# umount /mnt/new/usr
# umount /mnt/new
```

You can now power cycle your server. Make sure you remove the FreeBSD installation media. When the server reboots, you will be presented with a prompt asking you to specify your passphrase.

**Listing 4.** Partitioning scheme

```
# /dev/da1.eli:

8 partitions:
#        size   offset    fstype   [fsize bsize bps/cpg]
  a:       2G        *    4.2BSD        0     0
  b:       2G        *      swap
  c: 10485759        0    unused        0     0       # "raw" part, don't
edit
  d:       4G        *    4.2BSD        0     0
  e:       2G        *    4.2BSD        0     0
  f:        *        *    4.2BSD        0     0
```

**Listing 5.** Sample fstab entries

```
# Device                Mountpoint      FStype  Options         Dump
Pass#

/dev/da0.elib           none            swap    sw              0       0
/dev/da0.elia           /               ufs     rw              1       1
/dev/da0.elid           /var            ufs     rw              2       2
/dev/da0.elie           /tmp            ufs     rw              2       2
/dev/da0.elif           /usr            ufs     rw              2       2
```

Once you specify the proper passphrase, geli will decrypt and mount the file systems (Figure 5). You can now login as root. Note that there is no password for root yet. Your server or workstation is not configured. You would need to configure your server or workstation and create a proper `rc.conf`.

### Upgrades, Updates, and Patching

One thing to consider when running this kind of setup is that your boot slice will need to be updated in parallel to your root slice. If you decide to compile a new kernel, you want to make sure that you enable geli within your new kernel. To do this, just specify the following options within your kernel configuration file:

```
options GEOM_ELI
device crypto
```

For additional information on compiling a new kernel, reference chapter 8 of the FreeBSD handbook.

Once you compile a new kernel and you install it, you will need to make sure you copy it over to your boot slice. You can always mount your boot slice and copy over the new updates. For example, you could do something like this:

```
# mkdir /mnt/boot
# mount /dev/da0s1a /mnt/boot
# rm -R /mnt/boot/boot
# cp -Rp /boot /mnt/boot/
# umount /mnt/boot
```

If you ever run into an issue where you accidently damaged your boot partition and you can't boot up your operating system, you can always boot off of the DVD FreeBSD operating system media. You can then go to the Fixit shell, load geli, and then mount your encrypted file systems along with your boot partition. You can then repair whatever damage you may have caused. If all else fails, you can always install the default kernel that is found on the FreeBSD operating system DVD media.

## Using a boot CD/DVD or USB drive to load an encrypted FreeBSD root file system

In this section of the article, we're going to create a boot CD or DVD to boot up and decrypt our encrypted root file system. We are going to use a keyfile and password to decrypt our root file system.

If you don't want to use a boot CD or DVD, you can always use a bootable USB thumb drive.

Using a boot CD or DVD, or booting off of a USB drive to decrypt and mount a FreeBSD root file system requires that you already have a FreeBSD server up and running. You will also need a spare hard drive that you will be encrypting and installing the FreeBSD operating system onto.

In this article, I am running a workstation with FreeBSD 7.2 already installed. I have two SCSI hard drives. They are as follows:

- Drive `da0` – contains my FreeBSD 7.2 installation.
- Drive `da1` – empty disk which will contain my new encrypted root file system.

To start, boot up into a shell prompt on your workstation. Make sure you have the geli kernel module compiled into your kernel. If you don't, you will need to recompile your kernel with geli support. Please reference chapter 8 of the FreeBSD handbook for information on how to recompile your kernel.

We need to load the geli module so issue the following command at the command prompt:

```
# kldload geom_eli
```

Now that geli is loaded, we need to first create a directory where we will store our keyfile temporarily. For this example, I'm going to store my keyfile in the `/root/boot/key` directory. I then want to create a keyfile that is 256k in size.

```
# mkdir -p /root/boot/key
# dd if=/dev/random of=/root/boot/
key/master.key bs=256k count=1
```

I created a keyfile with random binary data located in the `/root/boot/key` directory called `master.key`.

Next I want to create my encrypted disk. I want to use a keyfile along with a password. This gives me additional protection in case someone gets a hold of my keyfile.

```
# geli init -v -b -e aes -l 256 -s
4096 -K /root/boot/key/master.key
/dev/da1
```

As you can see, I added the `-K` option to the geli init command. This specifies that I want to use a keyfile.

Now that our disk is encrypted, we need to attach our disk so that we can start using it. Issue the following command to attach the disk:

```
# geli attach -k /root/boot/key/
master.key /dev/da1
```

Note that we specified the `-k` option, in lower case, with the path to our keyfile. You will be asked for your passphrase before the disk is attached. Once the disk is attached or decrypted, it will be made available to the operating system via a new device. In this case, our new device is called /`dev/da1.eli`.

We need to now create our partitions on the attached drive. Issue the following commands:

```
# bsdlabel -w /dev/da1.eli
# bsdlabel -e /dev/da1.eli
```

You must now decide how you want to partition your operating system. For this example, I'm going to use the following scheme (see Listing 4).

In this example, I'm going to use partition `a` as my root `/` file system and I'm allocating 2 GB of space for it. Partition `b` is my swap and I'm allocating 2 GB to it. Partition `d` will be my `/var` directory and I'm allocating 4 GB to it. Partition `e` will be my `/tmp` directory and I'm allocating 2 GB to it. And finally, partition `f` will be my `/usr` directory and I'm allocating all available space to it. Note that I'm leaving the offset values at `*` which will make bsdlabel handle the values automatically.

Now save and exit the bsdlabel screen. We can now create our file systems using the newfs command. I'm going to create UFS 2 file systems so I issue the following commands:

```
# newfs -O 2 /dev/da1.elia
# newfs -O 2 /dev/da1.elid
# newfs -O 2 /dev/da1.elie
# newfs -O 2 /dev/da1.elif
```

Now that we created our new file systems, we want to mount them. To do this, we must first create our mount point. Issue the following command at the command prompt:

```
# mkdir /mnt/new
```

We can now mount our file system by issuing the following command:

```
# mount /dev/da1.elia /mnt/new
```

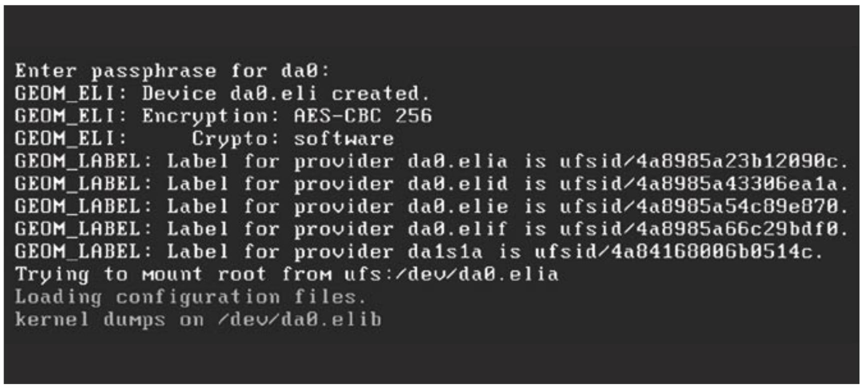
Now that our root file system is mounted to `/mnt/new`, we need to create the directory structure to mount our other file systems. Issue the following commands:

```
# mkdir /mnt/new/var
# mkdir /mnt/new/tmp
# mkdir /mnt/new/usr
```

Now we can go ahead and mount our other file systems that we created:

```
# mount /dev/da1.elid /mnt/new/var
# mount /dev/da1.elif /mnt/new/usr
```

Note that I am not mounting the `/dev/da1.elie` device or our `/tmp` file system.



```
Enter passphrase for da0:
GEOM_ELI: Device da0.eli created.
GEOM_ELI: Encryption: AES-CBC 256
GEOM_ELI:     Crypto: software
GEOM_LABEL: Label for provider da0.elia is ufsid/4a8985a23b12090c.
GEOM_LABEL: Label for provider da0.elid is ufsid/4a8985a43306ea1a.
GEOM_LABEL: Label for provider da0.elie is ufsid/4a8985a54c89e870.
GEOM_LABEL: Label for provider da0.elif is ufsid/4a8985a66c29bdf0.
GEOM_LABEL: Label for provider da1s1a is ufsid/4a84168006b0514c.
Trying to mount root from ufs:/dev/da0.elia
Loading configuration files.
kernel dumps on /dev/da0.elib
```

**Figure 6.** Bootup 2

The reason behind this is that during the operating system installation that we are going to be doing, no information is going to be written to that file system.

Now that our file systems are mounted, we can go ahead and install the operating system. You are going to need to download the full FreeBSD source via csup or cvsup. For information on how to download the FreeBSD source, please refer to chapter 6 of the FreeBSD handbook.

By default, you should have downloaded the FreeBSD source into the `/usr/src` directory. We're going to need to buildworld first so issue the following commands:

```
# cd /usr/src
# make buildworld
```

Please note that this will take quite a while depending upon the speed of your workstation or server. Now that you have compiled the source, we want to install it on the new encrypted file system. Issue the following commands to install the compiled source to our encrypted disk:

```
# make installworld DESTDIR=/mnt/new
# make distribution DESTDIR=/mnt/new
```

Now that we're done installing the FreeBSD operating system on the new drive, we want to compile a new kernel.

If you want to compile a custom kernel, make sure that you have geli enabled and that you disabled kbdmux. To enable geli, add the following to your custom kernel:

```
options GEOM_ELI
device crypto
```

To disabled kbdmux, make sure you omit the following line from your custom kernel:

```
device      kbdmux      # keyboard
multiplexer
```

For additional information on compiling a custom kernel, reference chapter 8 of the FreeBSD handbook.

To compile our kernel, we issue the following command:

```
# make buildkernel KERNCONF=GENERIC
```

In this example, I'm going to use the generic kernel. After your kernel compiles, install it onto your new drive by issuing the following command:

```
# make installkernel KERNCONF=GENERIC
DESTDIR=/mnt/new
```

We now want to make sure that geli loads during the boot-up process so that it can decrypt our encrypted root file system.

To do this, we want to edit the `loader.conf` file. Issue the following command:

```
# vi /mnt/new/boot/loader.conf
```

We now want to create the proper entries in the `loader.conf` so that it loads geli and the keyfile. Do this by putting the following in the loader.conf:

```
geom_eli_load="YES"
geli_da0_keyfile0_load="YES"
geli_da0_keyfile0_type="da0:geli_
keyfile0"
geli_da0_keyfile0_name="/key/
master.key"
```

Keep in mind that while we're setting up the loader.conf file, you have to reference the hard drive by what it will be called when you move it to its final destination. For example, I'm going to put my second SCSI drive called `da1` into a new workstation. On that workstation, it will become the primary drive, or `da0`. Replace `da0` in the above example with whatever the name the drive will become.

Geli and kbdmux, the keyboard multiplexer driver, seem to have problems working together. For geli to work properly, we need to disable the kbdmux driver. We can do this by issuing the following command at the command prompt:

```
# echo hint.kbdmux.0.disabled=\"1\"
>> /mnt/new/boot/device.hints
```

This will not be necessary if you disabled kbdmux inside the kernel.

Now that we have our boot directory setup on the `/mnt/new` file system, we want to copy it over to our temporary boot directory which in this case will be `/root/boot`. We want to make sure that we preserve all file permissions, file modes, user IDs, and group IDs. To do this, issue the `cp` command with the `p` option and the `R` option, for recursion, as follows:

```
# cp -Rp /mnt/new/boot /root/boot/
```

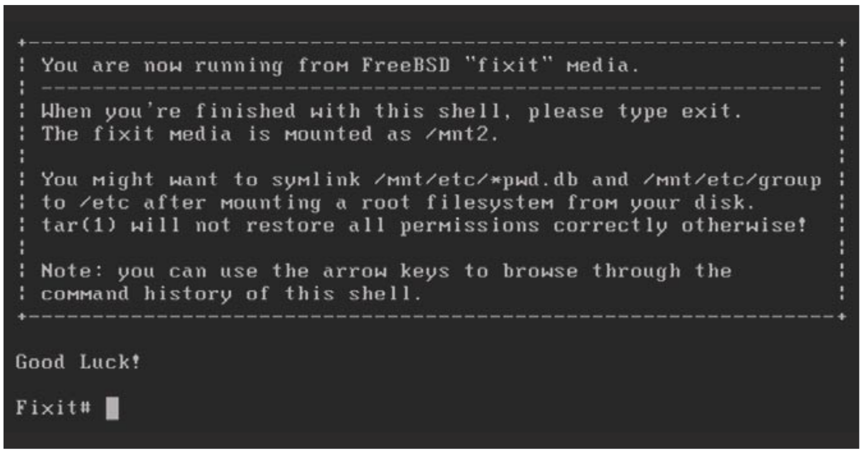
We now want to create a proper fstab so that our file system is mounted properly on boot-up. Issue the following command at the command prompt:

```
# vi /mnt/new/etc/fstab
```

Based on the partitions I created, I would create the following entries within the fstab file (see Listing 5).

Again, keep in mind that I am referencing the final location of the drive. When I move this drive to my other workstation, it will become the primary drive and it will be referenced as `da0` in the operating system.

You can specify additional options or make changes to the above entries



```
+-----------------------------------------------------------------+
| You are now running from FreeBSD "fixit" media.                 |
| --------------------------------------------------------------- |
| When you're finished with this shell, please type exit.         |
| The fixit media is mounted as /mnt2.                            |
|                                                                 |
| You might want to symlink /mnt/etc/*pwd.db and /mnt/etc/group   |
| to /etc after mounting a root filesystem from your disk.        |
| tar(1) will not restore all permissions correctly otherwise!    |
|                                                                 |
| Note: you can use the arrow keys to browse through the          |
| command history of this shell.                                  |
+-----------------------------------------------------------------+

Good Luck!

Fixit# █
```

**Figure 7.** fixit

based on your needs. Please reference the fstab(5) man page for more information.

Now that we created an fstab, we want to copy it over to our temporary boot directory. To do this, issue the following commands:

```
# mkdir /root/boot/etc
# cp /mnt/new/etc/fstab /root/boot/
etc/fstab
```

We are now finished with setting up our system. We can now unmount our file systems. Issue the following commands:

```
# umount /mnt/new/var
# umount /mnt/new/usr
# umount /mnt/new
```

We now want to either create a bootable CD or DVD, or use a bootable USB thumb drive to boot up our new system. To use a bootable CD or DVD, install CDRTools from the ports directory (`sysutils/cdrtools`). This will install mkisofs which we will use to create our bootable CD or DVD.

Issue the following command at the command prompt to create our bootable CD or DVD:

```
# mkisofs -R -no-emul-boot -b boot/
cdboot -o /root/boot.iso /root/boot
```

The above command will take the items inside your `/root/boot` directory and create a proper CD or ISO image. You can now burn the ISO image to a CD or DVD using your favorite CD or DVD burning software.

You can now shutdown your workstation and move the new drive to the new workstation or server. You will need to use your bootable CD or DVD to start up your workstation. During the boot up process, you will be asked for your passphrase. The keyfile will automatically be loaded (Figure 6). You will not be able to boot up your operating system unless you use your bootable CD or DVD media.

Keep in mind that you have a copy of your keyfile within the root directory located at `/root/boot/key/master.key`. I recommend you delete this file and the ISO image you created for security purposes. Your ISO image also contains your master.key file within it. You should also consider creating a second, backup, copy of the DVD or CD in case the first one gets damaged.

If you decide to use a bootable USB thumb drive instead of a bootable DVD or CD, you need to make sure that your computers BIOS supports booting off of USB devices. To setup a bootable USB thumb drive, you would partition and format the USB thumb drive using fdisk and bsdlabel. You would issue the `-B` option to both so that it creates a bootable partition. You would then mount and copy over the `/root/boot` directory.

## Upgrades, Updates, and Patching

One thing to consider when running this kind of setup is that if you lose your bootable CD or DVD, you will lose your whole drive. Therefore, I recommend you create a backup bootable CD or DVD and keep it in a safe, secure place.

Keep in mind that if you need to recompile your kernel, you will need to create a new boot CD or DVD. When you compile a new kernel, copy over the new kernel to your `/root/boot` directory. Then copy your keyfile from the CD or DVD into the `/root/boot/key/master.key` location. After that, you can create a bootable ISO and burn to DVD or CD. Remember to delete your master.key and the bootable ISO off of your hard drive when you are done burning your DVD or CD. If instead of a bootable CD or DVD, you are using a bootable USB boot drive, you will need to mount your USB device and copy over the updated kernel.

## About the Author

Jacques Manukyan is a system and network engineer working in the financial industry in New York City. He has been a BSD user for over a decade and is an avid fan of FreeBSD. He has worked tirelessly to introduce BSD within the organizations he has worked with, including service providers and financial companies located around the Wall Street and World Trade Center areas of downtown NYC.

# Setting up PC-BSD as a server

Jan Stedenhouder

PC-BSD is so easy to install and the KDE-desktop easy enough to use that we might almost forget it's roots as server operating system. Now, and in the future, the majority of desktop users might not consider this piece of information of any value.

But in others, the tinkerers, it might trigger the itch to try their hand on setting up a BSD or Linux based server. FreeNAS (BSD) of ClarkConnect (Linux) are tailor-made for such an experiment.

Personally, I am the proud owner of a BubbaTwo home server, which -in essence- is a Debian-based appliance with an easy-to-understand webinterface to set up the file-, mail-, media- and printservers. My question then became: *Would it be possible to build something similar, but with PC-BSD as a starting point?* The answer, no doubt, is: *Sure, why not*?. In this series of articles we will build up a home server, adding new building blocks step-by-step. The starting point is that of a desktop user, used as he or she is to a nice graphical user interface. The first article deals with installing PC-BSD as server, installing Webmin (a webbased tool to manage this new server) and making changes to the firewall in order to be able to use Webmin.

## Installing PC-BSD as a server

The option to install PC-BSD as server is already part of the graphical installation wizard. Just put the cd or dvd in the drive and boot the computer. Following the wizard we need to set up the system language and the keyboard layout, agree to the license agreement, before we can select *Server edition* as installation option (Figure 1).

This selection is the only one different from a regular installation of the desktop edition. The following steps of the wizard are exactly the same, including the possibility to install various desktop programs by selecting their PBI's (like Amarok and Firefox). The installer proposes a simple setup for the disk partitions, consisting of `/` and swap only. Compare that to the suggestion made by the FreeBSD installer, where the option to automatically create partitions also provides three additional partitions (`/var`, `/tmp` and `/usr`). Experienced users can use the partition editor in PC-BSD to setup the partitions according to their own needs (Figure 2).

Kris Moore, the PC-BSD project leader, acknowledged that the differences between the two install options (desktop and server) are minor at the moment. In fact, all files needed to run a graphical user interface are still there, the desktop is simply disabled. Changing the settings in `/etc/ttys` would be enough to get a fully functional graphical desktop again. Apart from this SSH is enabled and some networking is customized. For this article we stuck to the proposed disk layout and decided not to install additional PBI's.

Once the installation is finished, please reboot the system. The new PC-BSD server is ready for our purposes. The command line awaits us (Figure 3).

## Between the command line and the graphical desktop: Webmin

This would be enough to start adding the various building blocks for our home server. We can install all the software we need and edit the various configuration files to fine tune the various servers. Once our box is up and running, it isn't even necessary to sit behind the actual system. With a program like PuTTY we are able to access the command line from another computer in our network (or, when all elements are in place, from any computer in the world with an internet connection).

Webmin makes it somewhat easier to manage and maintain our new server. It provides a web-based interface and should remove the need to manually edit the configuration files. This doesn't mean that it negates the need to comprehend the various building blocks of our server. But more about that in the upcoming articles.

For now we simply want to install webmin. First, we will install webmin using the ports collection. As an alternative, we use a PBU to install the software.

## Installing webmin via ports

Beginning with the 7.x releases, PC-BSD keeps the FreeBSD ports tree completely separate from the PC-BSD base desktop system. This might not sit well with experienced FreeBSD users, but makes sense when thinking about the goal PC-BSD has in mind and it's target audience. Keeping the two separate allows for us to play with the ports without affecting our PC-BSD desktop negatively. And changes to the PC-BSD desktop won't corrupt the ports tree. Feel free to read up on this in the PC-BSD Handbook (*http://wiki.pcbsd.org/index.php/PC-BSD_Users_Handbook*).

The first step is to become root by entering:

```
$ su
```

and provide your administrator password. Then, we need to switch to -what is called- the FreeBSD LOCALBASE:

```
# runports
```

If all is well, we see the message:

```
Running as 'root'. You may now run
'make' in the FreeBSD Ports tree
#
```

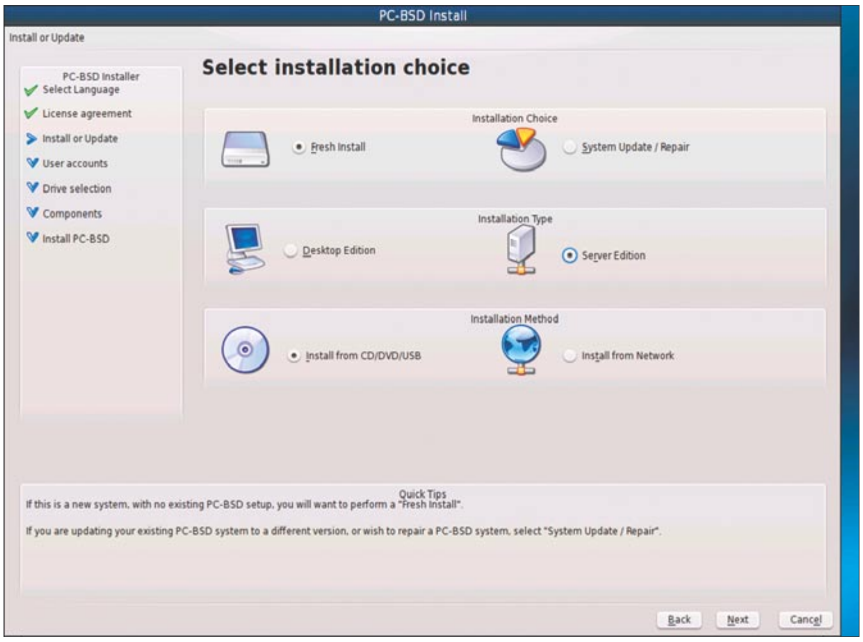
During installation we opted not to install any additional components, which included the ports collection. At this point we need to get a new ports tree. For that, please enter (Figure 4):

```
# portsnap fetch
# portsnap extract
```

Once this is finished we can install webmin by using the following instructions:

```
# cd /usr/ports/sysutils/webmin
# make install clean
```

All dependencies will be taken care of, but we need to stay behind our computer to answer a few questions here and there (for instance while



Select installation choice

Installation Choice: [✓ Fresh Install] System Update / Repair

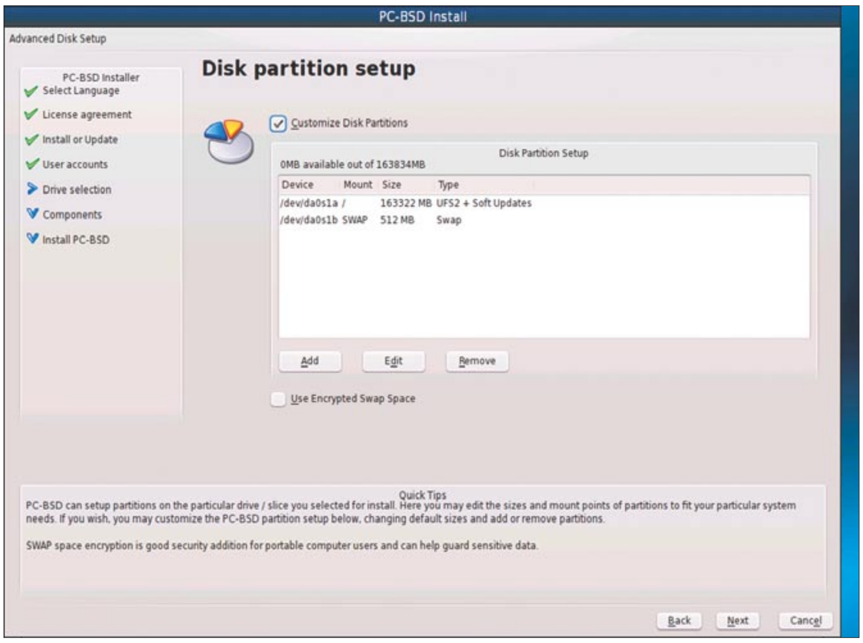
Installation Type: Desktop Edition [✓ Server Edition]

Installation Method: [✓ Install from CD/DVD/USB] Install from Network

Quick Tips
If this is a new system, with no existing PC-BSD setup, you will want to perform a "Fresh Install".

If you are updating your existing PC-BSD system to a different version, or wish to repair a PC-BSD system, select "System Update / Repair".

**Figure 1.** The PC-BSD disks provide an option to install the OS as server



Disk partition setup

[✓ Customize Disk Partitions]

Disk Partition Setup
0MB available out of 163834MB

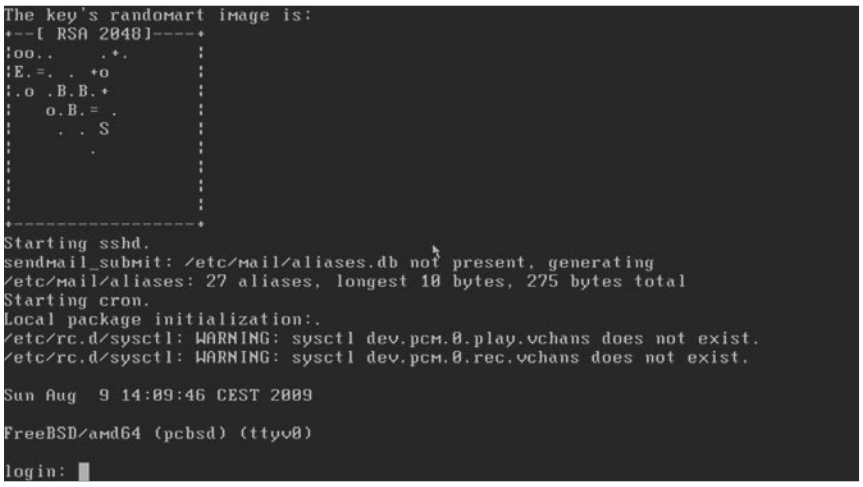
| Device | Mount | Size | Type |
|---|---|---|---|
| /dev/da0s1a | / | 163322 MB | UFS2 + Soft Updates |
| /dev/da0s1b | SWAP | 512 MB | Swap |

Use Encrypted Swap Space

Quick Tips
PC-BSD can setup partitions on the particular drive / slice you selected for install. Here you may edit the sizes and mount points of partitions to fit your particular system needs. If you wish, you may customize the PC-BSD partition setup below, changing default sizes and add or remove partitions.

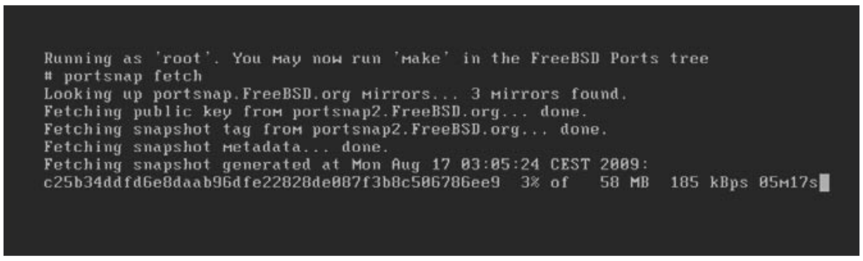
SWAP space encryption is good security addition for portable computer users and can help guard sensitive data.

**Figure 2.** The PC-BSD server install chooses a simple set of partitions



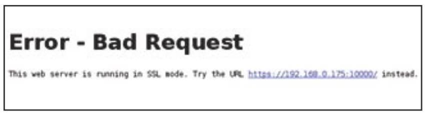
```
The key's randomart image is:
+--[ RSA 2048]----+
|oo..      .+.    |
|E.=.  . +o       |
|.o .B.B.+        |
|    o.B.= .      |
|     . . S       |
|        .        |
|                 |
|                 |
|                 |
+-----------------+
Starting sshd.
sendmail_submit: /etc/mail/aliases.db not present, generating
/etc/mail/aliases: 27 aliases, longest 10 bytes, 275 bytes total
Starting cron.
Local package initialization:.
/etc/rc.d/sysctl: WARNING: sysctl dev.pcm.0.play.vchans does not exist.
/etc/rc.d/sysctl: WARNING: sysctl dev.pcm.0.rec.vchans does not exist.

Sun Aug  9 14:09:46 CEST 2009

FreeBSD/amd64 (pcbsd) (ttyv0)

login: █
```

**Figure 3.** For someone used to a graphical user interface this takes some getting used to



```
Running as 'root'. You may now run 'make' in the FreeBSD Ports tree
# portsnap fetch
Looking up portsnap.FreeBSD.org mirrors... 3 mirrors found.
Fetching public key from portsnap2.FreeBSD.org... done.
Fetching snapshot tag from portsnap2.FreeBSD.org... done.
Fetching snapshot metadata... done.
Fetching snapshot generated at Mon Aug 17 03:05:24 CEST 2009:
c25b34ddfd6e8daab96dfe22828de087f3b8c506786ee9  3% of   58 MB  185 kBps 05m17s
```

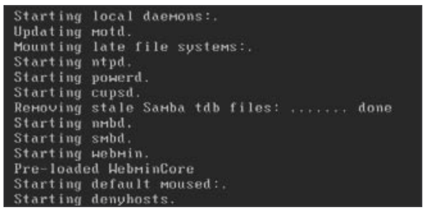
**Figure 4.** We need to get a clean ports tree in order to install webmin



**Error - Bad Request**

This web server is running in SSL mode. Try the URL https://192.168.0.175:10000/ instead.

**Figure 5.** Using a non-secure connection when SSL is enabled, webmin kindly suggests using the proper URL



```
Starting local daemons:.
Updating motd.
Mounting late file systems:.
Starting ntpd.
Starting powerd.
Starting cupsd.
Removing stale Samba tdb files: ....... done
Starting nmbd.
Starting smbd.
Starting webmin.
Pre-loaded WebminCore
Starting default moused:.
Starting denyhosts.
```

**Figure 6.** When we see this message scrolling by, we are certain that webmin is running/

installing perl). The default answers should suffice.

Now we need to setup webmin, for which we use:

```
# /usr/local/lib/webmin/setup.sh
```

This scripts asks us a few questions:

```
Web server port (default 10000):
Login name (default admin):
Login password:
Password: again:
Use SSL (y/n):
```

The default port for webmin is 10000. We can change the login name and should provide a password. It is possible to leave a blank password, but that's not advised. Another layer of security is added by using SSL. Without SSL we would be able to access the web interface by using http://server-ip-address:10000. SSL changes that to https://server-ip-address:10000. (Figure 5)

With this the installation of webmin is finished. To start the program we use:

```
# /usr/local/etc/rc.d/webmin start
```

If this were a regular FreeBSD server, we could edit the rc.conf file so webmin starts at boot. To do that, please use:

```
# ee /etc/rc.conf
```

ee is a nice command line text editor which is somewhat easier to use than vi, not in the least because it offers something resembling a menu with the various instructions. Add the following line:

```
webmin_enable="YES"
```

But this isn't a regular FreeBSD server, this is a PC-BSD server and webmin is not installed as part of the PC-BSD base system. We need to create (as root) a symlink to the `/Programs/rc.d` folder:

```
# ln -s /usr/local/etc/rc.d/webmin
/Programs/rc.d/webmin
```

This way webmin is started at boot (Figure 6).

## Installing webmin using the PBI

Another method is to use the PBI that is offered at *http://www.pbidir.org*. Simply use *webmin* as search phrase and select either the 32-bits or 64-bits version. What we need is the exact download address for the PBI. We use this URL in combination with 'fetch' to get the PBI to our server. In my case it came down to:

```
$ fetch ftp://www.nl.freebsd.org/
pub/pcbsd/PBI/Utilities/Webmin/7/x64/
Webmin1.480_1-PV0.pbi
```

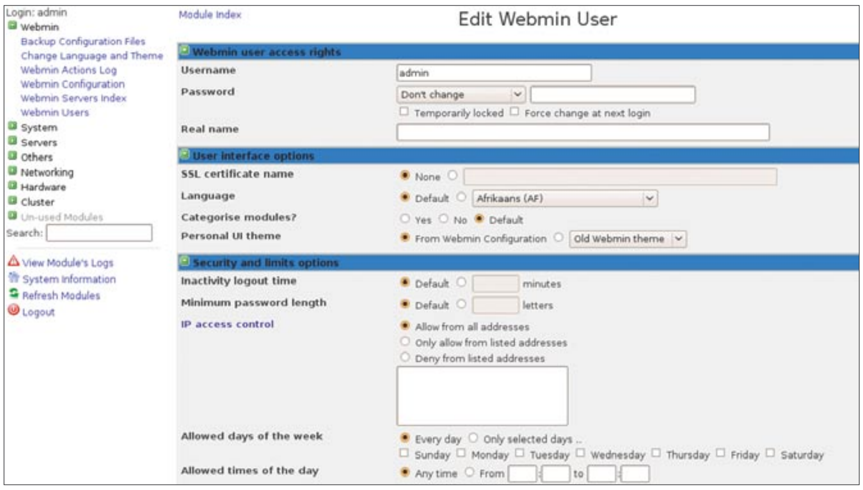
On a graphical desktop it now boils down to double-clicking the PBI and follow the wizard that appears. On the command line we need to change the file permissions first:

```
$ su
# chmod 777 Webmin1.480_1-PV0.pbi
```

Then we can install webmin by entering:

```
# ./Webmin1.480_1-PV0.pbi -text -
accept
```

Using the `-text` option tells the installation we are using a text-based interface, while the – `accept` options deals with possible



Edit Webmin User

Webmin user access rights

Username: admin
Password: Don't change
☐ Temporarily locked ☐ Force change at next login
Real name:

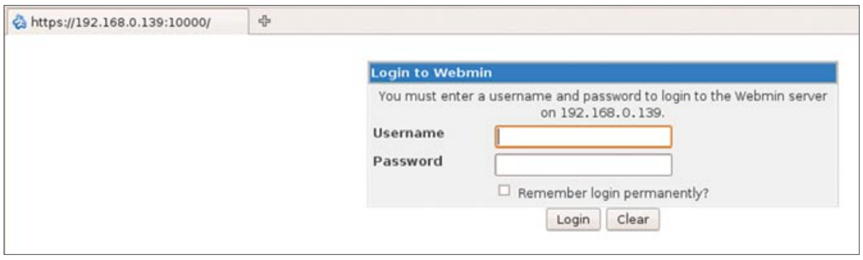
User interface options

SSL certificate name: [✓ None] ○
Language: [✓ Default] ○ Afrikaans (AF)
Categorise modules?: ○ Yes ○ No [✓ Default]
Personal UI theme: [✓ From Webmin Configuration] ○ Old Webmin theme

Security and limits options

Inactivity logout time: [✓ Default] ○ minutes
Minimum password length: [✓ Default] ○ letters
IP access control: [✓ Allow from all addresses] ○ Only allow from listed addresses ○ Deny from listed addresses
Allowed days of the week: [✓ Every day] ○ Only selected days ..
☐ Sunday ☐ Monday ☐ Tuesday ☐ Wednesday ☐ Thursday ☐ Friday ☐ Saturday
Allowed times of the day: [✓ Any time] ○ From : to :

**Figure 7.** When installing webmin via the PBI we need to make it more secure through the webmin options



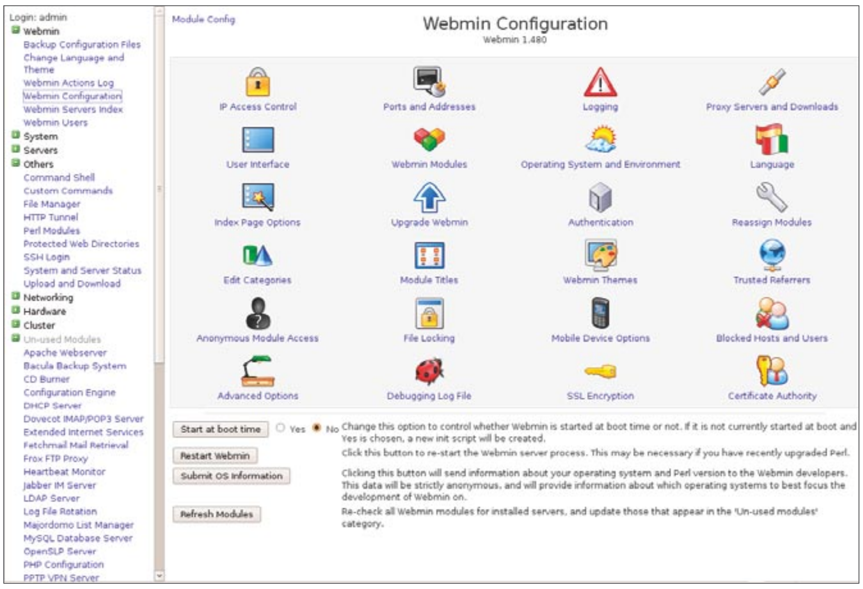
https://192.168.0.139:10000/

Login to Webmin

You must enter a username and password to login to the Webmin server on 192.168.0.139.

Username:
Password:
☐ Remember login permanently?

**Figure 8.** This shows that our installation of webmin was successful.We can login!



Start at boot time: Yes [✓ No] Change this option to control whether Webmin is started at boot time or not. If it is not currently started at boot and Yes is chosen, a new init script will be created.

Restart Webmin: Click this button to re-start the Webmin server process. This may be necessary if you have recently upgraded Perl.

Submit OS Information: Clicking this button will send information about your operating system and Perl version to the Webmin developers. This data will be strictly anonymous, and will provide information about which operating systems to best focus the development of Webmin on.

Refresh Modules: Re-check all Webmin modules for installed servers, and update those that appear in the 'Un-used modules' category.

**Figure 9.** Webmin is a powerful tool to manage and maintain our new PC-BSD server

pop-ups. There are some differences with the ports-based installation. Using the PBI doesn't give us the choice to enable SSL (disabled by default), changing the login name (admin by default) or providing an admin password (blank by default). This can be changed when running webmin (Figure 7).

## Setting up the firewall

Either way, we now have a working version of webmin on our server. We are, however, still not able to access it from another computer in our network. As a default most ports to the outside world are closed by the firewall, including port 10000. Installing webmin doesn't automagically open the gate. We need to edit the pc.conf file by hand:

```
#ee /etc/pf.conf
```

and add the line:

```
pass in on em0 proto tcp from any to
(em0) port 10000 keep state
```

and save the file. Port 10000 is now open to accept requests from all computers that have access to our network. Mind you, this isn't the most secure setup, but for now it will suffice. Please open a browser on another computer and use the following url:

```
https://server-ip-address:10000
(or http://server-ip-address:10000,
when SSL is not (yet) enabled)
```

In Firefox you might be asked to add an exception for the specific site as the SSL certificate isn't recognized. When all goes well we will be greeted by the webmin login (Figure 8).

Figure 9 gives an indication of the wide range of functions and options that can be dealt with via webmin. In the upcoming articles we will add new functions to our server, add users and groups and see how far we can tweak it all in order to provide a stable, easy to use, feature rich and secure home server.

P.S. I'd like to add a word of thanks to Kris Moore who answered my questions quick and clear enough for me to write this article.

### About the Author

Jan Stedehouder writes about open source software and open standards, mostly from the perspective of a novice user who wishes to migrate away from Windows. He is the author of three books, contributed to a textbook on open source and open standards and was co-editor of the Dutch Open Source Yearbook 2008-2009. His most recent book Open source en open standaarden. Voor niets gaat de zon op? (translated: Open source and open standards. Is it a free ride?) aims at introducing the general public to this topic. According to him, BSD should be seriously considered for desktop users as well.

# How to Build a Scalable Search Engine Using the BuildaSearch Web Service

Diego Montalvo

BuildaSearch was featured in the 4/2009 issue of BSD Magazine. While other articles do a fantastic job focusing on core BSD technology, I feel that it is also important to cover web services powered by BSD systems.

As the Internet and web applications continue to evolve and become both more useful and complex, the operating systems behind these Internet based services must also evolve into more efficient and powerful systems.

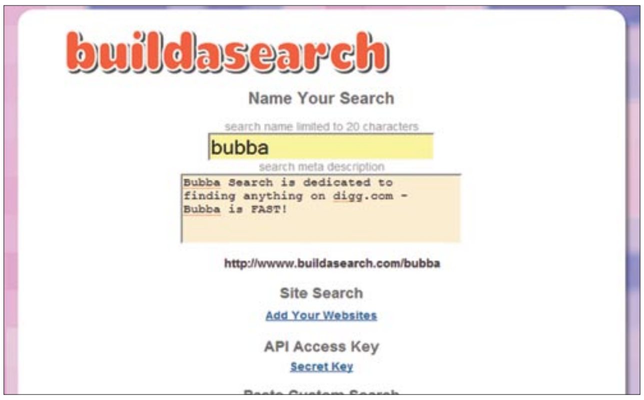
Being the lead developer of BuildaSearch.com, I am very familiar with resource intensive web services and their need for powerful operating systems.

FreeBSD provides the power for one of the most unique and advanced BuildaSearch features: a real-time indexer, which produces extremely fresh search results in minutes.

Since my early days developing WAP search technology via a FreeBSD back-end, I decided BuildaSearch would run on the same operating system. FreeBSD has become a critical part of BuildaSearch and its many search services.

In this tutorial I will be covering web-based indexing technology and simple deployment procedures. After reading the following article you will be able to: crawl one or more websites in real-time, add a scalable search engine to your personal, business or blog website. Things you'll need for this tutorial:

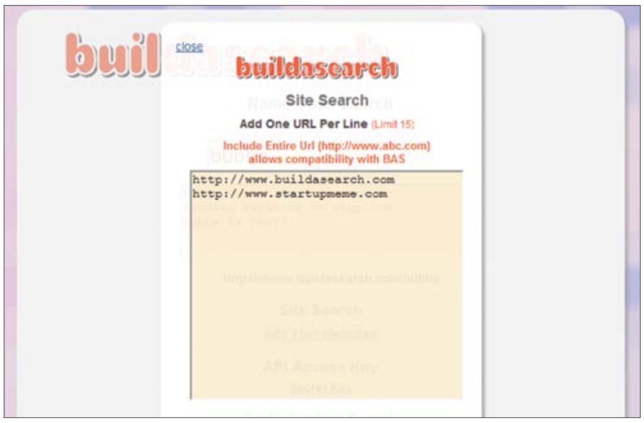
· web server
· PHP5.x



bubba

```
Bubba Search is dedicated to
finding anything on digg.com -
Bubba is FAST!
```

Figure 1. Giving Your Search a Name

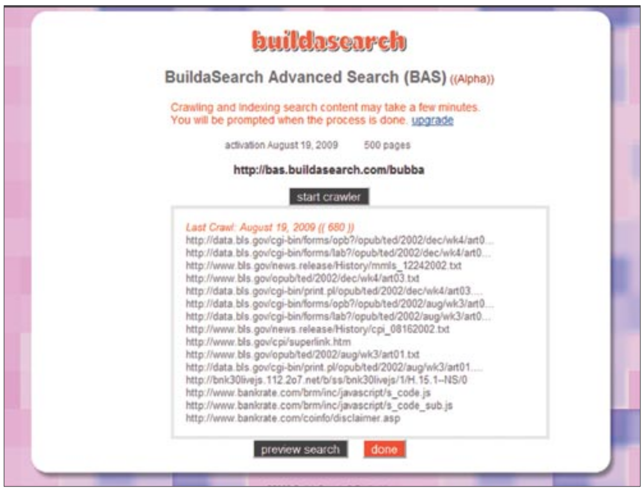


Add One URL Per Line (Limit 15)

Include Entire Url (http://www.abc.com) allows compatibility with BAS

```
http://www.buildasearch.com
http://www.startupmeme.com
```

Figure 2. Adding Websites to be Indexed



Crawling and indexing search content may take a few minutes. You will be prompted when the process is done. upgrade

activation August 19, 2009 500 pages

http://bas.buildasearch.com/bubba

```
Last Crawl: August 19, 2009 (( 680 ))
http://data.bls.gov/cgi-bin/forms/opb?/opub/ted/2002/dec/wk4/art0...
http://data.bls.gov/cgi-bin/forms/lab?/opub/ted/2002/dec/wk4/art0...
http://www.bls.gov/news.release/History/mmls_12242002.txt
http://www.bls.gov/opub/ted/2002/dec/wk4/art03.txt
http://data.bls.gov/cgi-bin/print.pl/opub/ted/2002/dec/wk4/art03....
http://data.bls.gov/cgi-bin/forms/opb?/opub/ted/2002/aug/wk3/art0...
http://data.bls.gov/cgi-bin/forms/lab?/opub/ted/2002/aug/wk3/art0...
http://www.bls.gov/news.release/History/cpi_08162002.txt
http://www.bls.gov/cpi/superlink.htm
http://www.bls.gov/opub/ted/2002/aug/wk3/art01.txt
http://data.bls.gov/cgi-bin/print.pl/opub/ted/2002/aug/wk3/art01....
http://bnk30livejs.112.2o7.net/b/ss/bnk30livejs/1/H.15.1--NS/0
http://www.bankrate.com/brm/inc/javascript/s_code.js
http://www.bankrate.com/brm/inc/javascript/s_code_sub.js
http://www.bankrate.com/coinfo/disclaimer.asp
```

Figure 3. BAS Crawler Display Screen

- SimpleXML (enabled) in PHP
- BuildaSearch XML API

Note: The BuildaSearch API is programming language independent you can use anything from JavaScript to C++ to parse the XML API. This tutorial provides a PHP5 code sample.

## Step 1. Getting Started

Registration at BuildaSearch is pretty straight forward. Register and confirm your email address in your inbox.

Once you login and are in the *Main Menu* click on the *Name Search* link, next you will see the *Name Your Search* interface. Begin by typing the name you would like to give your search. Note: only letters and numbers are supported.

The search in this tutorial is given the name *bubba* at web address: *http://bas.buildasearch.com/bubba* (Figure 1).

Once you are finished typing in your search name, click on the *Add Your Websites* link. In the smaller screen add the websites you would like to crawl. Note: You may add up to 15 different websites.

It is important to add the entire web address including its protocol and host.

```
correct: "http://www.abc123.com"
incorrect: "abc123.com"
```

Once you have added your list of websites, click *close* then click on the *save changes* button. Lastly click on *Continue Editing* then *home* on the top menu (Figure 2).

## Step 2. Crawling the Web

In the *Main Menu* click on the *BuildaSearch Advanced Search* (BAS) link, once in the BAS interface, you will notice that BuildaSearch offers 500 free search pages.

Note: You may upgrade to more pages but for this tutorial 500 pages is suffice.

Click *next* and you will be prompted with the default 500 pages and with a character set drop-down. Note: Most sites use either *European latin1* or *Unicode utf8* character sets.

Next click on *continue activation* once you are ready to begin crawling your websites, click *start crawler.* Note: Crawling and Indexing search content may take a few minutes. As the crawler does its magic you will see the list of links change.

Once all indexing processes are completed, you will be prompted with two buttons: *preview search* and *done.* You may test search results by clicking on the *preview search* and entering your own search query into the provided search box. Once you are satisfied with testing your custom search you may proceed to the next step by clicking on *done* (Figure 3).

## Step 3. Using the BuildaSearch API

The BAS XML API works similar to other APIs, simply use any programming language of your choice to parse and manipulate the feed. Your custom search API can be viewed at the following url: *http://bas.buildasearch. com/xml/your_search_name?e=query_string&bastart=0&bascount=10.*

## Step 4. Embedding and Customizing Your Search Engine

The last step to this tutorial is parsing the BAS API using your choice of programming language. I have provided a simple PHP5 SimpleXML code sample below.

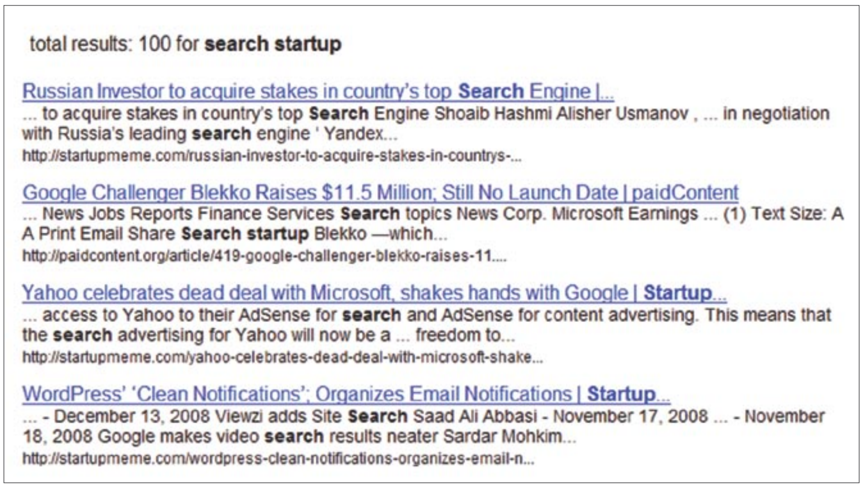
Even though this code sample provides basic functionality it is a great start for developing a more complete search engine which could be enhanced with: pagination, spell checking, CSS and more.

**Listing 1.** Simple XML Response

```
<?xml version="1.0" encoding="UTF-8" ?>
<response>
<results>
<query>market7</query>
<bastotal>171</bastotal>
<bastart>10</bastart>
<bacount>1</bacount>
<basresult>
<title><![CDATA[Shoaib Hashmi (ShoaibHashmi) on Twitter]]></title>
<longurl><![CDATA[http://twitter.com/ShoaibHashmi]]></longurl>
<basummary><![CDATA[... reply to anumvighio RT @ startupmeme :
<b>Market7</b> Goes Big, Provides Google With ]]></basummary>
<showurl><![CDATA[http://twitter.com/ShoaibHashmi]]></showurl>
</basresult>
</results>
</response>
```

**Table 1.**

| Customizable Values | |
|---|---|
| your_search_name | search name given in step one |
| query_string | search query value |
| bastart | starting point of search results (0) default |
| bascount | display X number of results (10) default |


total results: 100 for **search startup**

Russian Investor to acquire stakes in country's top **Search** Engine |...
... to acquire stakes in country's top **Search** Engine Shoaib Hashmi Alisher Usmanov , ... in negotiation with Russia's leading **search** engine ' Yandex...
http://startupmeme.com/russian-investor-to-acquire-stakes-in-countrys-...

Google Challenger Blekko Raises $11.5 Million; Still No Launch Date | paidContent
... News Jobs Reports Finance Services **Search** topics News Corp. Microsoft Earnings ... (1) Text Size: A A Print Email Share **Search startup** Blekko —which...
http://paidcontent.org/article/419-google-challenger-blekko-raises-11....

Yahoo celebrates dead deal with Microsoft, shakes hands with Google | **Startup**...
... access to Yahoo to their AdSense for **search** and AdSense for content advertising. This means that the **search** advertising for Yahoo will now be a ... freedom to...
http://startupmeme.com/yahoo-celebrates-dead-deal-with-microsoft-shake...

WordPress' 'Clean Notifications'; Organizes Email Notifications | **Startup**...
... - December 13, 2008 Viewzi adds Site **Search** Saad Ali Abbasi - November 17, 2008 ... - November 18, 2008 Google makes video **search** results neater Sardar Mohkim...
http://startupmeme.com/wordpress-clean-notifications-organizes-email-n...

**Figure 4.** Screenshot of Parsed Search Results

**Listing 2.** PHP5 SimpleXML Code Sample

```
<?php
header("Content-type: text/html; charset=UTF-8");
########################################
# BUILDASEARCH BSD MAGAZINE EXAMPLE
# USING BAS API
# USING SIMPLEXML V.081909
########################################
$SECRET = 'bubba';//YOUR CUSTOM SEARCH NAME
$QUERY = urlencode($_GET['e']);
$START = 0;
$NUM = 10;
if(empty($QUERY)){
echo '<form method="get" action="">';
echo '<input type="text" name="e">';
echo '<input type="submit" value="search">';
echo '</form>';
  } else {
//BUILDASEARCH API URL
$API = 'http://bas.buildasearch.com/xml/'.$SECRET.'?e='.$QUERY.'&bastart='.$START.'&bascount='.$NUM.'';
//LOAD EXTERNAL BAS API
$xml = simplexml_load_file($API, 'SimpleXMLElement',LIBXML_NOCDATA);
//DISPLAY TOTAL
$result = $xml->xpath('/response/results/bastotal');
while(list( , $node) = each($result)) {
echo 'total: ',$node,"<br/>";
}
//DISPLAY RECORDS BELOW
foreach ($xml->results->basresult as $record) {
echo '<a href="'.$record->longurl.'">'.$record->title.'</a><br/>';
echo $record->basummary.'</br>';
echo $record->showurl.'</br>';
echo '<hr/>';
        }
}
?>
```

## Resources

You can test drive the code sample in this tutorial at: *http://www.buildasearch.com/bsdmag/*
Obtain the code sample at URL below: *http://www.buildasearch.com/bsdmag/source.phps*

**Table 2.**

| BuildaSearch API Return Values | |
|---|---|
| query | search query |
| bastotal | total returned results |
| title | title of search results includes bold highlighted text |
| longurl | non-truncated original URL |
| basummary | text summary includes bold highlighted text |
| showurl | truncated display URL |
| bastart | result set starting point |
| bacount | total results being displayed |

## Conclusion

Adding an advanced search engine to your website can be done in minutes instead of hours. As BuildaSearch technology improves, more and more advanced features will be implemented into our web service. If you have any questions or comments feel free to contact me at: *diego@earthoid.com.*

## About the Author

Diego Montalvo is the founder and the core developer of BuildaSearch.com. When he is not behind the old computer, he enjoys chilling out, biking, painting, reading and a cold pint of Guinness.

great subscription offer

v7.0 Now Available
aintcorporation.com
Integrated Vulnerability Scanning and Penetration Testing
MAGAZ
BSD
WHAT'S NEW
NOW 84 PAGES
INFINITY. FREEDOM. FREEBSD.
INSIDE
HOW TO BUILD A SCALABLE SEARCH ENGINE USING THE BUILDASEARCH WEB SERVICE
EMAIL SERVER IN FREEBSD - CONFIGURING FREEBSD AS A MAIL SERVER WITH POSTFIX AND DOVECOT IN FREEBSD 7X
BSD AS THE PLATFORM FOR CONNECTING STRATEGY TO OPERATIONS THROUGH A DATA CONCOURSE SERVICE
KEEPING FREEBSD UP-TO-DATE: OS ESSENTIALS •
ENCRYPTING THE FREEBSD ROOT FILE SYSTEM •
SETTING UP PC-BSD AS A SERVER •
IS NETBSD READY FOR A DESKTOP? •
FREEBSD ON THE SHEEVAPLUG •
USING BSD FOR YOUR STUDIES •
THE FREEBSD CHATTERBOX •
EXCLUSIVELY
BSD Tips and Tricks by Dru Lavigne and Mikel King

**SAVE $20!**

Get your copy of BSD mag and save $20 of the shop prices

**Why subscribe?**

- save $20
- 4 issues delivered directly to you
- never miss an issue

**3 EASY WAYS to subscribe:**

- *visit: www.bsdmag.org*
- *call: 1 917 338 36 31*
- *fill in the form and post it*

# BSD magazine ORDER FORM

□ **Yes**, I'd like to subscribe to BSD magazine starting with issue:

□ 1/2008(1) FreeBSD Ins & Outs
□ 2/2008(2) OpenBSD in the Limelight
□ 1/2009(3) Explore NetBSD
□ 2/2009(4) PC-BSD Uncovered
□ 3/2009(5) Guide to FreeBSD
□ 4/2009(6) BSD Security/OpenBSD
□ 1/2010(7) Infinity. Freedom. FreeBSD
□ 2/2010(8)
□ 3/2010(9)

Order information
(□ individual user/ □ company)

Title ______________________
Name and surname ______________________
address ______________________
______________________
postcode ______________________
tel no. ______________________
email ______________________
Date ______________________

Company name ______________________
Tax Identification Number ______________________
Office position ______________________
Client's ID* ______________________
Signed** ______________________

Payment details:
□ USA $39.99
□ Europe 29.99€
□ World 39.99€

I understand that I will receive 4 issues over the next 12 months.

Contact Information:

Software Wydawnictwo Sp.z.o.o
Bokserska 1
02-682 Warszawa
Poland

Subscription Service
email: *subscription_support@bsdmag.org*
phone: 1 917 338 36 31

# Is NetBSD ready for a desktop?

Petr Topiarz

In this article I am focusing on the usability of the NetBSD as a desktop. I would like to show what NetBSD can do today and whether it is mature enough to challenge PC-BSD or Linux. If you want to know, keep reading.

This article will not explain how to install NetBSD as that has been covered by many other articles, but how to tweak it and hack it to make it work similarly user-friendly and comfortably like a current standard Linux distro.

## The idea of a NetBSD desktop

I wonder if you ever had a NetBSD desktop. I used to run it, mainly for fun, three years ago as my secondary desktop. It was a lot of tweaking and tuning, but after patching kernel and doing a lot of hacks it was quite nice – even with a splash screen when loading. However, a lot of things I needed were not working reliably. A year ago, or two, the pkgsrc collection (the main source of third party packages) went through major changes and both Gnome and KDE did not compile successfully for months. So I stopped using it. The problems at that time could be clearly attributed to switching X-server from *Xfree86* to *Xorg*, which required changing lots of dependencies and introducing new ones, but they were enough to keep me off the NetBSD system for a long time.

However, few months ago Jared McNeill and Andrew Doran announced they were starting the NetBSD Desktop project. You can read about it at: *http://wiki.netbsd.se/Desktop_Project*. Their aim was a fully featured gnome desktop with everything that Linux distros such as Ubuntu or Fedora run today regularly. Even automountig with hal and all the stuff. What a surprise, I said to myself, I knew the former name very well, as it was the developer who helped me patch the kernel for the loading splash three years ago. I began interested and tried to install Gnome. It compiled well both from the stable sources 2009Q1, and 2009Q2 and even on current. I tried automounting with hal and it worked (after asking Jared Mc Neill for help again). I tried Flash 10 on native Firefox 3 and it worked. Well, it really seems, there is something going on in the NetBSD world! If you like a preview of what we can expect from the NetBSD Desktop's project, here come my notes of how to install NetBSD with all what a desktop needs.

### What a desktop needs

Let's find out what a regular spoiled Linux user expects from a desktop:

- complete Gnome or KDE
- nice splash while loading and cool look
- automounting USB and CDROM devices
- easy printing
- easy CD burning
- easy scanning
- OpenOffice, PDF-reader, streaming player
- Skype, ICQ, Jabber, etc
- DVD playing
- Internet streaming music playing

and a lot of other special things, I believe. Even though the items above are probably standard, I'll take you on a guided tour and we'll see if those things work on NetBSD or not.

## The binary packages

The package management system on NetBSD is very unique. Thanks to backwards compatibility you can theoretically use a package compiled for two years old release on the newest kernel and userland. You can also use the newest packages on a two years old kernel and userland, e.g. Use 2009Q2 branch packages on NetBSD 4.0 if you like. If you use a package compiled for a different release, it does not always work, but if there are not many changed dependencies it is likely to work. In some cases, that can save you, when the current package will not compile. It has its advantages and drawbacks. It was more then once that I had to use a package compiled for 3.0 release with an old packages branch on my 4.0 release, if I remember correctly, it was the rsync package, when it failed to compile. Of course, you never can achieve that stability and perfection like with the OpenBSD packages compiled only for one Xenocara release and kernel version, however, it often allows you to find a solution when the current package is missing, especially if the package does not have too many dependencies.

You can find the packages for intel platform on: *ftp://ftp.netbsd.org/pub/pkgsrc/packages/NetBSD/i386/5.0_2009Q2/All*, the number 5.0 refers to the release of NetBSD the number 2009Q2 refers to the frozen state of PKGSRC after the 2nd quarter of the year 2009 to use that normally as a repository you have to

```
# export PKG_PATH=ftp://
ftp.netbsd.org/pub/pkgsrc/packages/
NetBSD/i386/5.0_2009Q2/All
```

If you want it to be permanent then paste the above line into `/etc/profile` – if it does not exist, you have to create that file using the packages. Then it is very easy:

- adding a gnome desktop package:

```
# pkg_add -v gnome
```

- deleting the rsync package:

```
# pkg_delete rsync
```

- deleting all packages at once:

```
# pkg_delete '*'
```

This can be very useful if you are upgrading from one set of packages to the newer, e.g. From 2009Q1 to 2009Q2.

- finding out which packages are installed: pkg info on searching for a package I truly recommend Swedish web engine: *http://pkgsrc.se.*

It is very helpful in finding any package or its proper name or version. It allows you to search all packages or only those in CURRENT branch or the stable 2009Q2 branch or earlier.

I do not use packages from NetBSD, and I do not recommend it generally. Thanks to the backward compatibility and possibility to compile packages from almost any branch against any release of NetBSD and any kernel, a number of them are often not compiled against the same kernel and userland as you are using, so they do not always work as expected.

My latest bad experience was the GDM binary from the 2009Q1 that installed without hesitations but always collapsed when trying to start. After searching in the error the logs I found out the binary was trying to use the old version of X server and so I had to make

**Listing 1.** rc.conf

```
#       $NetBSD: rc.conf,v 1.96 2000/10/14 17:01:29 wiz Exp $
#
# see rc.conf(5) for more information.
#
# Use program=YES to enable program, NO to disable it. program_flags are
# passed to the program on the command line.
#
# Load the defaults in from /etc/defaults/rc.conf (if it's readable).
# These can be overridden below.
#
if [ -r /etc/defaults/rc.conf ]; then
        . /etc/defaults/rc.conf
fi
# If this is not set to YES, the system will drop into single-user mode.
#
rc_configured=YES
# Add local overrides below
wscons=YES
dhclient=YES
sshd=YES
famd=YES
rpcbind=YES
dbus=YES
hal=YES
avahidaemon=YES
gdm=YES
cupsd=YES
slpd=NO
```

**Listing 2.** Adding more applications

```
#! /bin/sh
cd /usr/pkgsrc/print/cups &&
make install &&
cd /usr/pkgsrc/sysutils/gnome-volume-manager  &&
make install &&
cd /usr/pkgsrc/multimedia/adobe-flash-plugin  &&
make install &&
cd /usr/pkgsrc/www/firefox3  &&
make install &&
cd /usr/pkgsrc/www/nspluginwrapper &&
make install &&
cd /usr/pkgsrc/net/skype &&
make install &&
echo "Skype, firefox3, nspluginwrapper, flash and cups printing system have
installed succesfully on your system!  "
```

link from the new to the old directory to make it work. So what do I use and recommend? I use packages that I compile myself and I use the magic of pkgsrc – the packages source.

## Pkgsrc

PKGSRC is a system similar to FreeBSD's ports and portage in the Linux world. PKGSRC is a hierarchy of folders and files where information how to compile packages, where to get them, how to patch them and pack them for NetBSD is stored. And it works. It works on NetBSD, it works on OpenBSD, Solaris, DragonFly BSD, Linux... probably even on a more recent version of a kitchen toaster.

Once you get into it and understand how it works and how to hack it, nothing will stop you.

NetBSD has a great documentation and you will find there where to grab it and how to install it. Once installed, the PKGSRC resides in `/usr/pkgsrc` on a NetBSD machine. There are not only sources of real packages but also sources of so called meta-packages. So make sure your connection to the Internet works, become root and descend to the folder called meta-packages and go to the gnome folder:

```
$ su
# cd  /usr/pkgsrc/meta-pkgs/gnome
```

Then you can have a look at the Makefile where information of what will compile is stored.

You can comment out some things or add others if you like, however, you should know what you're doing before doing so. Then you tell the system to install the complete desktop:

```
# make install
```

and the circus starts, after a day or two – depending on your connection speed and PC capabilities, your gnome desktop is there and you will see this message appear:

```
$NetBSD: MESSAGE,v 1.6 2009/03/17 14:
46:39 jmcneill Exp $
In order to get the GNOME Desktop
running properly, you need to follow
these manual steps:

1) Enable the File Alteration Moni-
   tor. See pkg_info -D fam for more
   information. If you chose to use
   gamin instead of fam, you do not
   need to take this step.
2) Enable the system dbus daemon.
   In order to do that, copy the
   ${PREFIX}/dbus script to /etc/
   rc.d and add dbus=YES to your /
   etc/rc.conf file.
3) Enable the hal daemon if GNOME
   has been built with the hal option
   (the default). In order to do
   that, copy the ${PREFIX}/share/
   examples/rc.d/hal script to /etc/
   rc.d and set hal=YES in your /etc/
   rc.conf file.
4) Enable the cups daemon if you
   installed it. In order to do that,
   copy the ${PREFIX}/share/examples/
   rc.d/{cupsd,slpd} scripts to /etc/
   rc.d and set cupsd=YES and slpd=NO
   in your /etc/rc.conf file.
5) Set up the gnome-screensaver
   PAM service by creating the /
   etc/pam.d/gnome-screensaver file.
   You can use one of the files in
   ${PREFIX}/share/examples/gnome-
   screensaver/pam.d as templates.
6) Optionally enable the Avahi DNS
   Service Discovery service if you
   installed it. In order to do that,
   copy the ${PREFIX}/share/examples/
   rc.d/avahidaemon script to /etc/
   rc.d and set avahidaemon=YES in
   your /etc/rc.conf file.
7) Optionally enable GDM (highly
   recommended). Just copy the
   ${PREFIX}/share/examples/rc.d/
   gdm script to /etc/rc.d and add
   gdm=YES to your /etc/rc.conf file.
```

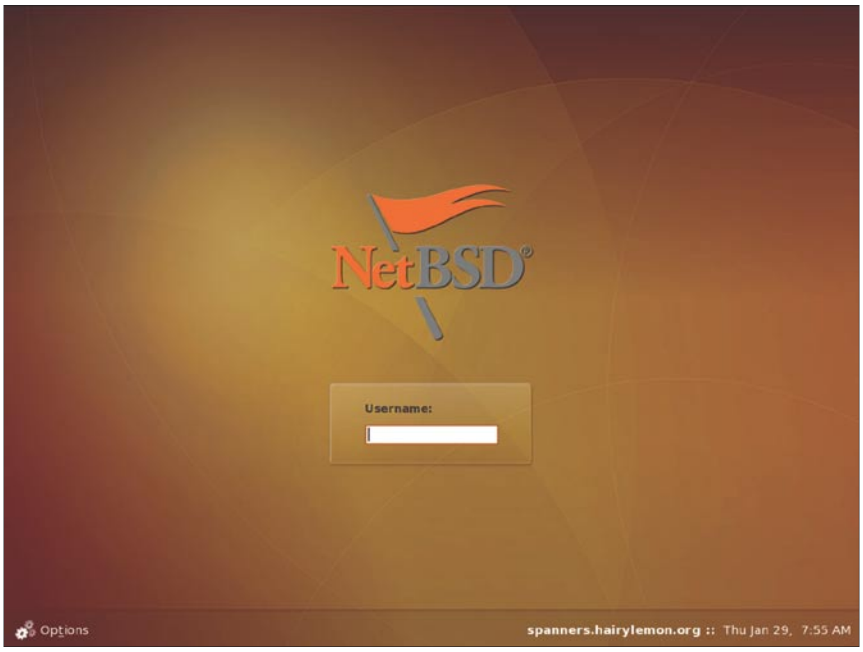
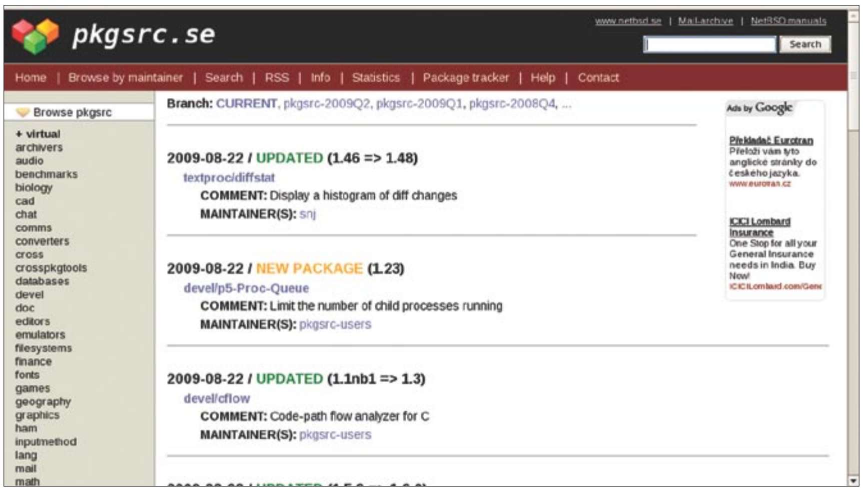


**Figure 1.** Future gdm in netbsd



Branch: CURRENT, pkgsrc-2009Q2, pkgsrc-2009Q1, pkgsrc-2008Q4, ...

2009-08-22 / UPDATED (1.46 => 1.48)
textproc/diffstat
COMMENT: Display a histogram of diff changes
MAINTAINER(S): snj

2009-08-22 / NEW PACKAGE (1.23)
devel/p5-Proc-Queue
COMMENT: Limit the number of child processes running
MAINTAINER(S): pkgsrc-users

2009-08-22 / UPDATED (1.1nb1 => 1.3)
devel/cflow
COMMENT: Code-path flow analyzer for C
MAINTAINER(S): pkgsrc-users

Překladač Eurotran
Přeloží vám tyto anglické stránky do českého jazyka.
www.eurotran.cz

ICICI Lombard Insurance
One Stop for all your General Insurance needs in India. Buy Now!
ICICILombard.com/Gene

**Figure 2.** pkgsrc website

You really should do what they say if you want your computer run gnome properly.

## The /etc/rc.conf

To explain what is going on one has to understand how the system works. Pkgsrc does not make applications and servers start automatically instead – it leaves the choice up to you. So it installs all the starting scripts into: `/usr/pkgsrc/share/examples/rc.d/` They do nothing there, however, if you copy them into `/etc/rc.d/` they are ready to use by the `rc.conf` file containing main system start-up configuration. The `rc.conf` is a the main starting script config. NetBSD uses only this one, in fact. In FreeBSD there are also other files to use for starting scripts and modules loading besides `/etc/rc.conf` and in OpenBSD `/etc/rc.conf` is rather stagnant as most configuration happens in `/etc/rc.local` and `/etc/rc.conf.local`.

Under NetBSD you have a very simple control over everything that the system launches just in one file. Services are switched on with simple YES and off with simple NO and you are ready to go! This choice makes NetBSD very powerful. The creators of a rather recent linux distribution, the Archlinux, who understood the advantage and adopted this concept too.

So if you want to have an openssh-server running on your desktop just add `sshd=yes` into your `/etc/rc.conf`. A sample `rc.conf`, that is enough for running a desktop looks like this (see Listing 1). But let us move back to pkgsrc.

## The /etc/mk.conf

The file called `mk.conf` contains information on compiling. You can put various options here, but for the moment we will stick to allowing pkgsrc to compile things we need and accept licenses and add these lines to `/etc/mk.conf`:

```
ACCEPTABLE_LICENSES+= flash-license
ACCEPTABLE_LICENSES+= skype-license
```

If you are interested what you agree with, go to the pkgsrc file containing sources of that program and issue the following command:

```
$ cd /usr/pkgsrc/net/skype
$ make show-license
```

**Listing 3.** Editing /usr/pkg/etc/PolicyKit/PolicyKit.conf\

```
<?xml version="1.0" encoding="UTF-8"?> <!-- -*- XML -*- -->

<!DOCTYPE pkconfig PUBLIC "-//freedesktop//DTD PolicyKit Configuration 1.0//EN"
"http://hal.freedesktop.org/releases/PolicyKit/1.0/config.dtd">

<!-- See the manual page PolicyKit.conf(5) for file format -->

<config version="0.1">
        <match user="root">
                <return result="yes"/>
        </match>
<match user="peter">
                <return result="yes"/>
        </match>

        <define_admin_auth group="wheel"/>
</config>
```

**Listing 4.** Scanning

```
# sane-find-scanner

  # sane-find-scanner will now attempt to detect your scanner. If the
  # result is different from what you expected, first make sure your
  # scanner is powered up and properly connected to your computer.
  # No SCSI scanners found. If you expected something different, make sure that
  # you have loaded a kernel SCSI driver for your SCSI adapter.

found USB scanner (vendor=0x03f0 [hewlett packard], product=0x4105 [hp scanjet]) at libusb:/dev/usb0:/dev/ugen0
```

## Adding more applications

We now have a nice NetBSD desktop with GNOME environment, but we are still nowhere, no Firefox, no Flash, no automounting no nothing a spoiled Linux user considers appropriate for a desktop. Here is a sample script that will do the choice for you, feel free to change it to your needs (see Listing 2).

## pkgsrc on a laptop

You have probably noticed that a Gnome desktop compiles a day or longer and another extra hours are needed to compile the rest of applications you want. Who can stop using his/her laptop for three days or longer? Here packages come handy. As I showed above, packages from the Internet repository can be rather a risky solution. So let's make them on our own, there is nothing easier. For example using a script (I bet there is hundred people who can make it more simple and clean, but this one works):

```
#! /bin/sh
for i in /usr/pkgsrc/*/*
do
if test -d $i
then cd $i
if test -d work
then
make package
fi
fi
done
```

This script searches your PKGSRC collection to find which packages have been compiled (they are the folders which have stale work in them) and proceeds with making packages. The compiled packages appear in your `/usr/pkgsrc/packages/All` folder. Now you can copy them to your NetBSD laptop and then go to the folder with packages:

```
# pkg_add ./gnome ./skype ./firefox3
(and similarly on, it solves
dependencies well)
```

or simply:

```
# pkg_add *
```

Then quick installation from packages breaks out and your laptop is ready in a few minutes. Of course, you have to do the configuration job with `rc.conf` and copy starting scripts again. This can be done on slow computers or emulated machines as well.

## Cleaning pkgsrc

After the installation process, you will find out that your `/usr` partition is running out of space. That is because after cleaning a lot of pkgsrc files still contains stale work.

The official web-site tells you that the only safe way to clean ports is to go to the main directory and do the following:

```
# cd /usr/pkgsrc/
# make clean
```

After two or three days the pkgsrc is perfectly clean. The average person does not have that much time, so if you want to save some time, here is a script for your comfort:

```
#! /bin/sh
for i in /usr/pkgsrc/*/*
do
if test -d $i
then cd $i
if test -d work
then
make clean
fi
fi
done
```

## Configuring and tweaking

Well, now we think the basic work is done. Yes, we have the applications, but most of them will not do what we want yet.

## Make flash work

As root issue a command:

```
# nspluginwrapper -l
  Original plugin: /usr/pkg/lib/
netscape/plugins/libflashplayer.so
    Wrapper version string: 1.2.2
```

This shows us where the plugin lies. Therefore we know what to install:

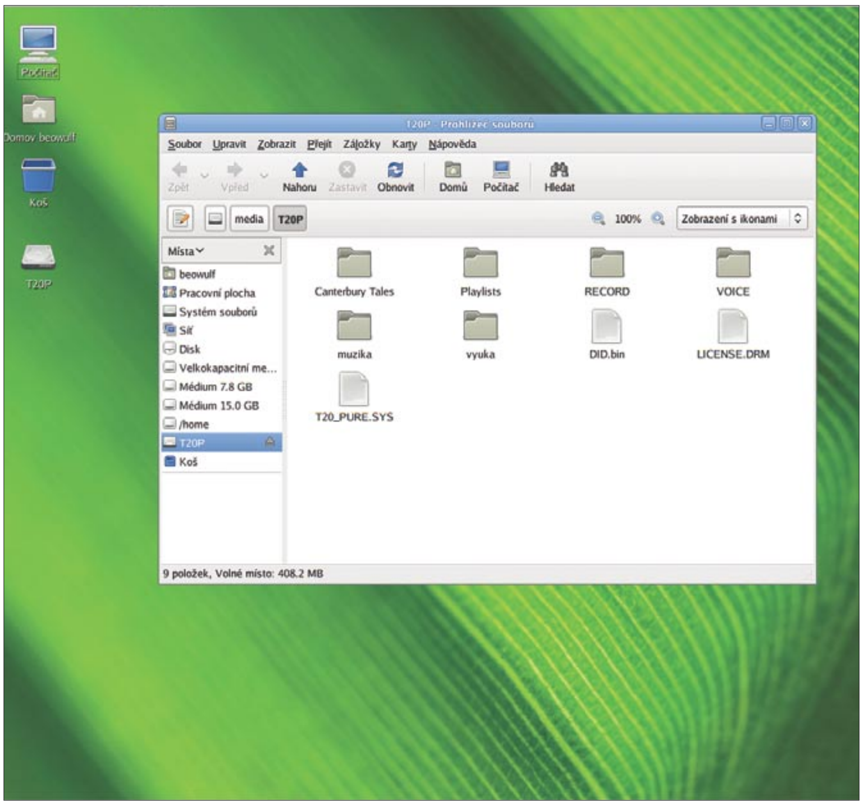


**Figure 3.** Automounting with Hal

```
# nspluginwrapper -i /usr/pkg/lib/
netscape/plugins/libflashplayer.so
```

Now fire up your Firefox 3 and you are in a flash heaven on Youtube. See the screenshot from *youtube.com* (Figure 5).

## Make hal automount the USB flash key and others

Make sure you have installed gnome-volume-manager. Then edit `/usr/pkg/etc/PolicyKit/PolicyKit.conf\` and make sure it looks like this (see Listing 3). You need to change this line `<match user="peter">` to whatever your *username* is. This gives you the power to mount volumes with hal and more, e.g. suspend and resume. Originally, this file only allows root to mount volumes. It can be edited much more cleverly using groups and such. See study hal pages about Policykit if you like: *http://hal.freedesktop.org/docs/PolicyKit/*.

Now, the usb-key mounting or cd-rom mounting should work. In case you still come across any difficulties, check if cdrom or usb is quoted in your `/etc/fstab`. If so, comment out those lines. Hal does not use `/etc/fstab` for mounting usb-keys or CDs. See the screenshot showing automounted USB Iriver player T20 (Figure 3).

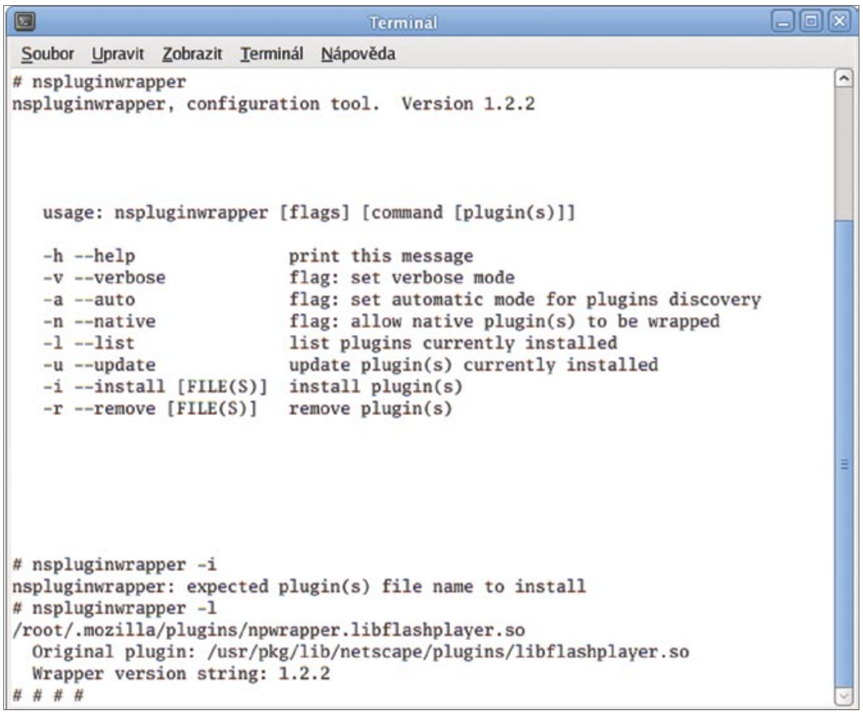
## Make Skype talk

Skype works under Linux emulation on NetBSD. It takes some time to start and I haven't found a solution how to make sound on Skype work with Pulse audio, that comes compiled with Gnome 2.26. now. If you kill pulse audio server, things can work. It is in the process now, clearly, as in 2009Q2 skype 1.4 only chatted. I had to download Skype 1.3 from the NetBSD distfiles and after unpacking and running it I could call, it was stable and I could speak and hear everyone normally. In current pkgsrc, Skype 1.3 did not speak to me at all while the nice Skype 1.4 did play sound and before it got completely stuck it even started to record from my microphone. It is very probable that even version 1.4 will soon work.

To make Skype talk you have to do several adjustments to the system settings.

In any case, to use skype as a user you have to ajust your permissions of

`/dev/sound*` `/dev/audio*` and `/emul/linux/dev/dsp*` `/emul/linux/dev/sound*`.



```
# nspluginwrapper
nspluginwrapper, configuration tool.  Version 1.2.2


   usage: nspluginwrapper [flags] [command [plugin(s)]]

   -h --help                print this message
   -v --verbose             flag: set verbose mode
   -a --auto                flag: set automatic mode for plugins discovery
   -n --native              flag: allow native plugin(s) to be wrapped
   -l --list                list plugins currently installed
   -u --update              update plugin(s) currently installed
   -i --install [FILE(S)]   install plugin(s)
   -r --remove [FILE(S)]    remove plugin(s)



# nspluginwrapper -i
nspluginwrapper: expected plugin(s) file name to install
# nspluginwrapper -l
/root/.mozilla/plugins/npwrapper.libflashplayer.so
  Original plugin: /usr/pkg/lib/netscape/plugins/libflashplayer.so
  Wrapper version string: 1.2.2
# # # #
```

Figure 4. Using nspluginwrapper

Then enable your system to recieve and send audio at the same time:

```
# audioctl -w fullduplex=1
```

This will switch your microphone on:

```
# mixerctl -w inputs.mic.mute=off
```

And adjust microphone recording level:

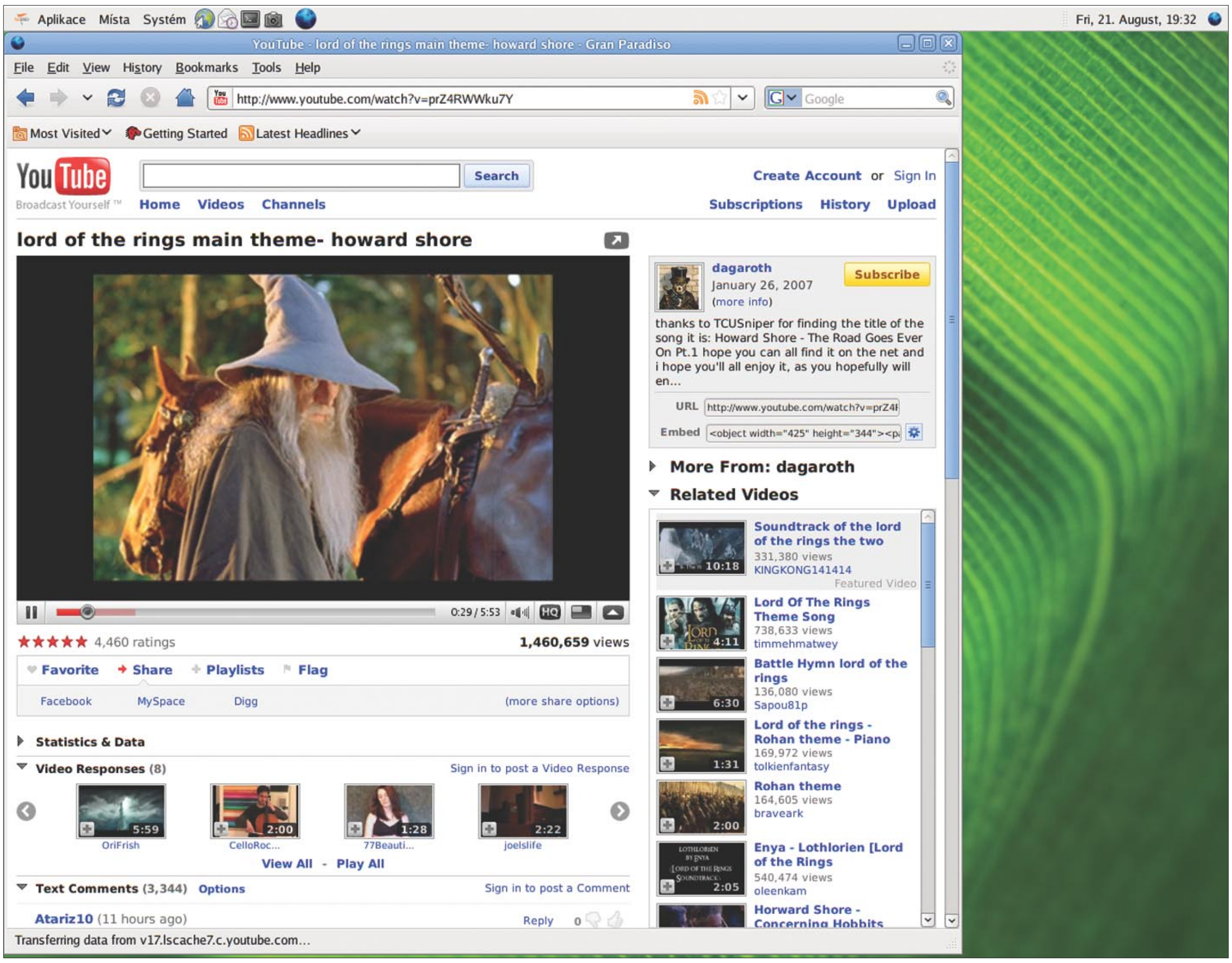
```
# mixerctl -w inputs.mic=240
```

## Make CUPS print from all applications

By default the path is setup so, that when printing applications are set-up to use `/usr/bin/lp` instead of `/usr/pkg/bin/lp`. We need to be able to use cups, so we have to do the following (saving the original files):

```
# cp /usr/bin/lp /usr/bin/lp.old
# cp /usr/bin/lpr /usr/bin/lpr.old
# cp /usr/bin/lpq /usr/bin/lpq.old
# cp /usr/bin/lprm /usr/bin/lprm.old
```

Linking cups's lp files to the original expected path:


Aplikace Místa Systém
Fri, 21. August, 19:32
YouTube - lord of the rings main theme- howard shore - Gran Paradiso
File Edit View History Bookmarks Tools Help
http://www.youtube.com/watch?v=prZ4RWWku7Y
Google
Most Visited Getting Started Latest Headlines
YouTube Broadcast Yourself
Search
Create Account or Sign In
Home Videos Channels
Subscriptions History Upload
lord of the rings main theme- howard shore
dagaroth
Subscribe
January 26, 2007
(more info)
URL http://www.youtube.com/watch?v=prZ4
Embed <object width="425" height="344"><p
More From: dagaroth
Related Videos
0:29 / 5:53
4,460 ratings
1,460,659 views
Favorite Share Playlists Flag
Facebook MySpace Digg (more share options)
Statistics & Data
Video Responses (8)
Sign in to post a Video Response
OriFrish CelloRoc... 77Beauti... joelslife
View All - Play All
Text Comments (3,344) Options
Sign in to post a Comment
Atariz10 (11 hours ago)
Reply
Soundtrack of the lord of the rings the two 331,380 views KINGKONG141414 Featured Video
Lord Of The Rings Theme Song 738,633 views timmehmatwey
Battle Hymn lord of the rings 136,080 views Sapou81p
Lord of the rings - Rohan theme - Piano 169,972 views tolkienfantasy
Rohan theme 164,605 views braveark
Enya - Lothlorien [Lord of the Rings 540,474 views oleenkam
Horward Shore - Concerning Hobbits
Transferring data from v17.lscache7.c.youtube.com...

thanks to TCUSniper for finding the title of the song it is: Howard Shore - The Road Goes Ever On Pt.1 hope you can all find it on the net and i hope you'll all enjoy it, as you hopefully will en...

Figure 5. Flash in Firefox 3

## Notes

If you run PKGSRC package system on a different system, it is likely that your pkgsrc file is in a different location. In that case you have to change /usr/pkg prefix in the above examples to whatever your prefix is. All tests and installations were done on a Pentium 4 2GHz with 512MB RAM, system description: NetBSD 5.99.15 NetBSD 5.99.15 (GENERIC) #0: Fri Aug 14 23:02:59 PDT 2009 *builds@wb33:/home/builds/ab/HEAD/i386/200908140002Z-obj/home/builds/ab/HEAD/src/sys/arch/i386/compile/GENERIC i386* following websites were used and are related to the above text:

- *http://wiki.netbsd.se/*
- *http://pkgsrc.se/*
- *http://www.netbsd.org/*
- *http://www.openunix.eu/*

```
# ln -s /usr/pkg/bin/lp /usr/bin/lp
# ln -s /usr/pkg/bin/lpr /usr/bin/lpr
# ln -s /usr/pkg/bin/lpq /usr/bin/lpq
# ln -s /usr/pkg/bin/lprm /usr/bin/
lprm
```

Then open any internet browser window and go to: *http://localhost:631* or *http://127.0.0.1:631* if the former option does not work. Administering cups is a rose garden walk with that interface. Of course you might want to install additional drivers from hpijs and hplip. You can do that using the pkgsrc.

### Scanning

If you want to scan as a user with a usb-connected scanner, install XSane and adjust devices to be readable and writable by your user. If you add yourself to the wheel and operator groups then it is enough to change the permissions as follows.

In my case, after investigating where the scanner is connected, it is done as root like this (see Listing 4) I knew I had to adjust permission of these files:

```
# chmod 660 /dev/usb* /dev/ugen*
```

Then I ran the testing command as a user and I got a positive answer:

```
$ scanimage -L
device `hp3900:libusb:/dev/usb0:/dev/
ugen0' is a Hewlett-Packard Scanjet
4370 flatbed scanner
```

Now I knew Xsane would find the flatbed scanner even for a user.

### Burning CD's

If you want to burn cd's you have to adjust `/dev/cd0* /dev/rcd*` to be readable and writable by user. If you add yourself and operator groups to the wheel then it is enough to:

```
# chmod 660 /dev/cd0* /dev/rcd*
```

Then you should check that your favourite cd-burner knows that the cd device is `/dev/rcd0` or similar.

### Playing DVD's with and without encryption

First do the same permission adjust-ments as in the case of Burning CD's. Then, if you install vlc or use totem (part of Gnome) or xine-ui, playing DVD is no problem.

There is an issue with encrypted DVD's. Remembering playing encoding DVD's is licensed and restricted in many countries. If you are in the country where it is allowed, you can install libdvdcss, but have to tell the system where to get the source and that we agree to the license. Add these two lines:

```
LIBDVDCSS_MASTER_SITES=http://
download.videolan.org/pub/libdvdcss/
ACCEPTABLE_LICENSES= libdvdcss-license
```

into `/etc/mk.conf` and then go to terminal:

```
# cd /pkgsrc/multimedia/libdvdcss/
# make install
```

### Nice look and NetBSD branding

When the GDM starts you see a very modest login window with grey background and a small NetBSD logo. NetBSD logo also appears at the menu of Gnome. That's in fact all. There are various nice backgrounds in the Internet, you can download and use, but not many. If you download kernel sources open the kernel configuration file and uncomment the following files, a kernel compiles with a nice splash screen with progress:

```
# enable VGA raster mode capable
of displaying multilingual text on
console
# enable splash screen support;
requires hw driver support
options        SPLASHSCREEN
options        SPLASHSCREEN_PROGRESS
```

It is not too modern, however it is quite stylish and nice. Orange and silver logo with blinking dots on the right bottom. If you read what Jared McNeill and Andrew Doran proposed for the NetBSD desktop we can expect visible improvement. By the way, if I am not mistaken, Jared McNeill was the person who made the current splash screen for NetBSD kernel.

If you want to see what the future of NetBSD on the desktop is, you can have a look at the new proposed GDM screenshot for NetBSD Desktop (see Figure 1).

## Conclusion

Though it does not have a 3-click-installer and it really takes some time and knowledge to set-up and tweak, almost everything that a desktop needs works on NetBSD. You can play Flash in a browser, you can listen to music streams, play DVDs, burn CDs, chat and (with some effort) talk over the Internet, automount usb-keys, mount Linux and Windows volumes, write documents, print and scan whatever you like. That means immense progress has been done by NetBSD developers (with the support of donators in Fund-raising Campaign last year and this year again) since last year compared to what NetBSD was a year ago. And the most promising point here is – if Andrew Doran and Jared McNeill accomplish their aim and bring up the NetBSD Desktop project to life, it will definitely be a challenge for PC-BSD or modern Linux desktops.

## Abot the Author

Petr Topiarz is a co-owner and manager of a small language school in Prague, involved also in an EU-financed online teaching project. He started with Linux and BSD back in 2004 and since 2005 he has done the upkeep of three BSD-served networks and has becoome a great BSD fan. He runs a modest portal called openunix.eu dealing with BSD and similar issues.

Looking for help, tip or advice?
Want to share your knowledge with others?
Visit BSD magazine forum
BSD MAGAZINE
Give us your opinion about the magazine's content
and help us to create the most useful source for you!
www.bsdmag.org

# FreeBSD on the SheevaPlug

Donald T. Hayford

Though NetBSD is better known for supporting a wide variety of processors and systems, FreeBSD has an active embedded component, as well. In this article, we'll take a look at the ARM-based SheevaPlug and show you how to boot your Plug using FreeBSD.

The *SheevaPlug* (SP) is the archetype of the plug computer, a new style of small computer whose distinguishing feature is that instead of plugging into a wall wart for power, it is the wall wart (see Figure 1). Built around the Marvell (*www.marvell.com*) 88F6281 Kirkwood line of SOC (system on chip) processors, the SheevaPlug comes with Gigabit Ethernet (but not wireless), USB, and 512 MB of both system RAM and NAND Flash memory. A disassembled SP is shown in Figure 2. The Kirkwood processor supports a large number of peripherals as shown in Table 1, only a few of which are used by the SP. The SP is touted as a development system by Marvel and comes with an internal interface board that adds a SDIO memory slot and a mini-USB port that provides access to a serial port console and JTAG port as two different com ports. The SP is available from Globalscale Technologies (*www.globalscaletechnologies.com*) for around $99US.

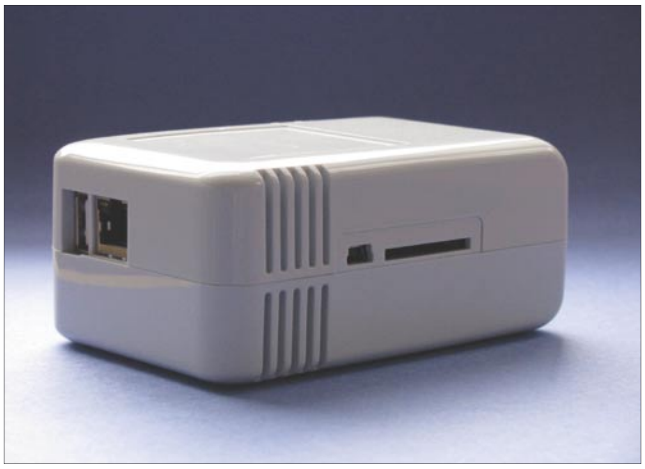


**Figure 1.** The ShevaPlug Computer. The USB and GigE ports are on the left, and the debug devices (mini-USB and SDIO memory card) are on the right

**Listing 1.** CSUP Control File (sheeva.supfile) Used To Download FreeBSD Source Code

```
# IMPORTANT: Change the next line to use one of the CVSup mirror sites
# listed at http://www.freebsd.org/doc/handbook/mirrors.html
*default host=cvsup8.FreeBSD.org
*default base=/var/db
*default prefix=/home/hayford/sp
*default release=cvs tag=. date=2009.08.09.00.00.00
*default delete use-rel-suffix
*default compress
src-all
```

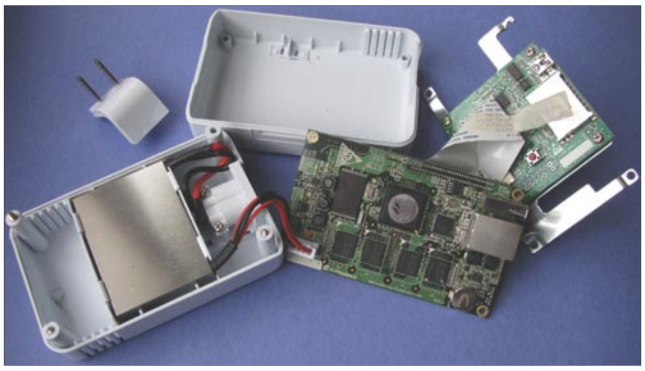


**Figure 2.** The SheevaPlug Disassembled. The circuit board on the far right contains the development hardware (serial console, JTAG interface, USB, and SDIO interfaces). The circuit board in the center holds the Marvell processor, Ethernet port, USB port, and system memory. The power supply is under the metal plate in the case on the left side

According to the PlugComputer website (*www.plugcomputer.org*, operated by Marvell), *A plug computer is a tiny, low power server, intended to provide network-based services within the home...It is an always-on system, and can serve data and applications to computing devices within the home. It can also be a bridge between home computing devices and Internet-based services.* Low power, in this case, means about 15 W, definitely less

**Listing 2.** Steps Required To Build FreeBSD For The SheevaPlug

```
$ mkdir /home/hayford/sp
$ csup sheeva.supfile
$ cd /home/hayford/sp
$ wget http://people.freebsd.org/~raj/patches/arm/sheevaplug.diff
$ cd sp/src
$ patch <../sheevaplug.diff
$ export MAKEOBJDIRPREFIX="/home/hayford/obj"
$ make -j 8 buildworld TARGET_ARCH=arm
$ make buildkernel TARGET_ARCH=arm KERNCONF=SHEEVAPLUG
$ su
# setenv DESTDIR /usr/home/hayford/sroot
# mkdir -p $DESTDIR
# make installworld TARGET_ARCH=arm
# make distrib-dirs TARGET_ARCH=arm
# make distribution TARGET_ARCH=arm
# make buildkernel TARGET_ARCH=arm KERNCONF=SHEEVAPLUG
# exit
$ mkdir /tftpboot/sp
$ cp /home/hayford/obj/arm/usr/home/hayford/sp/src/sys/SHEEVAPLUG/
kernel.bin \
> /tftpboot/sp
```

**Table 1.** Comparison Of The Hardware Capabilities Of The Kirkwood Processor And The SheevaPlug

| Characteristic | Available on Marvell 88F6281 | SheevaPlug Capability |
|---|---|---|
| Clock Speed | 1.0 – 1.2 GHz | 1.2 GHz |
| L1 Cache | 16K data, 16K program | Same |
| L2 Cache | 256 kB | Same |
| Memory Interface | 16-bit, DDR2, up to 800 MHz | 512 MB |
| Ethernet | 2 – GigE Interfaces | 1 – GigE Interface |
| PCI-Express | 1 Port | None available |
| USB | 1 USB-2.0 Port with integrated PHY | Same |
| SATA | 2 SATA 2.0 Ports with integrated PHYs | None available |
| TDM Channels | 2 | None available |
| SD/SDIO/MMC | 1 | 1 SDIO slot (can also be used as General Purpose IO) |
| NAND Flash | 8-bit NAND flash interface with boot support | 512 MB |
| SPI | 1, up to 50 MHz clock | None available |
| TWSI (Two Wire Serial Interface) | 1 General purpose master/slave port | None available |
| UART | 2 Available | 1, Serial Console Interface and Debug Interface |
| Audio | I2S/SPDIF | None available |
| Video | MPEG Transport Stream | None available |

than a standard desktop or laptop. One variety of plug computer, based on the same base design as the SP, is the Pogoplug (*www.pogoplug.com*), a device that allows you to put a USB storage device on the Internet so you can access your data from any computer. The Pogoplug doesn't have the same development capabilities as the SP, so if you're interested in system level programming, stick with the SP. More recently, a slightly more expensive device has become available that takes better advantage of the Kirkwood's capabilities; see *www.open-rd.org* for information on the SP's big brother, the OpenRD platform.

Not surprisingly, the SP comes with Linux preloaded in the flash memory, along with U-Boot, a powerful boot loader that has become very popular on ARM-based devices. Both FreeBSD and NetBSD offer support for a number of varieties of ARM devices, and Rafal Jaworowski (*http://www.semihalf.com*) recently announced support for a number of Marvell devices on the FreeBSD-ARM mail list. Consequently, it is now possible to build a version of FreeBSD that will boot on the SP. While building and running FreeBSD on an embedded processor is interesting in it's own right, we will also use this opportunity to compare the NetBSD build and installation process with FreeBSD's.

**Listing 3.** DHCP Configuration File On The DHCP Server

```
ddns-update-style ad-hoc;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option domain-name-servers xxx.xxx.xxx.xxx; # Use your nameserver address
default-lease-time 2592000;
allow bootp;
allow booting;
option routers 192.168.1.1;

subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.110 192.168.1.189;
        }

group   {
      host sheevaplug {
                hardware ethernet 00:50:43:XX:XX:XX;  # Use your MAC address
                fixed-address 192.168.1.109;
                next-server 192.168.1.171;   # Use your TFTP server address here
               option root-path "192.168.1.171:/usr/home/hayford/sroot";  # Put your NFS server here
                }
        }
```

**Listing 4.** Fragment Of The File etc/inetd.conf On The TFTP Server

```
# run comsat as root to be able to print partial mailbox contents w/ biff,
# or use the safer tty:tty to just print that new mail has been received.
#comsat dgram   udp     wait    tty:tty /usr/libexec/comsat     comsat
#
# ntalk is required for the 'talk' utility to work correctly
#ntalk  dgram   udp     wait    tty:tty /usr/libexec/ntalkd     ntalkd
tftp    dgram   udp     wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
tftp    dgram   udp6    wait    root    /usr/libexec/tftpd      tftpd -l -s /tftpboot
```

**Listing 5.** Add These Lines To /etc/rc.conf To Enable The NFS Server (FreeBSD)

```
# Add these lines to rc.conf
nfs_server_enable="YES"
rpcbind_enable="YES"
mountd_flags="-r"
rpc_lockd_enable="YES"
rpc_statd_enable="YES"
```

## What you'll need:

- A SheevaPlug (*http://www.globalscaletechnologies.com/p-22-sheevaplug-dev-kit-us.aspx*).
- A Windows computer that can run the driver software from Marvell so you can talk to the SheevaPlug. There is information available on the web (*http://dev.gentoo.org/~armin76/arm/sheevaplug/install.xml* or *http://mail-index.netbsd.org/current-users/2009/05/25/msg009557.html*) that shows how to use Linux or NetBSD to talk to the serial port on the SheevaPlug, if you'd like to try that. Be forewarned that some of these methods require you to reflash the SheevaPlug. Besides, you needed to something for that Windows box to do.
- A computer (X86-compatible) that runs a recent version of FreeBSD. For reasons we'll discuss below, the build/boot process will be easier if this computer is also the NFS server that your SP can use as a a root drive.

**Listing 6.** The /etc/exports File That Allows The SheevaPlug To Attach To The Root Directory

```
#The following examples export /usr to 3 machines named after ducks,
#/usr/src and /usr/ports read-only to machines named after trouble makers
#/home and all directories under it to machines named after dead rock stars
#and, /a to a network of privileged machines allowed to write on it as root.
#/usr                   huey louie dewie
#/usr/src /usr/obj -ro  calvin hobbes
#/home   -alldirs       janice jimmy frank
#/a      -maproot=0   -network 10.0.1.0 -mask 255.255.248.0
#
# You should replace these lines with your actual exported filesystems.
# Note that BSD's export syntax is 'host-centric' vs. Sun's 'FS-centric' one.
/usr/home/hayford/sroot -maproot=root -network 192.168.1 -mask 255.255.255.0
```

**Listing 7.** Loading The FreeBSD Kernel Onto The SheevaPlug With TFTP

```
   __  __                      _ _
  |  \/  | __ _ _ ____   _____| | |
  | |\/| |/ _' | '__\ \ / / _ \ | |
  | |  | | (_| | |   \ V /  __/ | |
  |_|  |_|\__,_|_|    \_/ \___|_|_|
 _   _     ____              _
| | | |   | __ )  ___   ___ | |_
| | | |___|  _ \ / _ \ / _ \| __|
| |_| |___| |_) | (_) | (_) | |_
 \___/    |____/ \___/ \___/ \__|
** MARVELL BOARD: SHEEVA PLUG LE

U-Boot 1.1.4 (Mar 19 2009 - 16:06:59) Marvell version: 3.4.16

U-Boot code: 00600000 -> 0067FFF0  BSS: -> 006CEE80

Soc: 88F6281 A0 (DDR2)
CPU running @ 1200Mhz L2 running @ 400Mhz
SysClock = 400Mhz , TClock = 200Mhz

DRAM CAS Latency = 5 tRP = 5 tRAS = 18 tRCD=6
DRAM CS[0] base 0x00000000   size 256MB
DRAM CS[1] base 0x10000000   size 256MB
DRAM Total size 512MB  16bit width
Flash:  0 kB
Addresses 8M - 0M are saved for the U-Boot usage.
Mem malloc Initialization (8M - 7M): Done
NAND:512 MB

CPU : Marvell Feroceon (Rev 1)

Streaming disabled
Write allocate disabled

USB 0: host mode
PEX 0: interface detected no Link.
Net:   egiga0 [PRIME], egiga1
Hit any key to stop autoboot:  0
Marvell>> dhcp
BOOTP broadcast 1
DHCP client bound to address 192.168.1.109
Marvell>> tftpboot 900000 sp/kernel.bin
Using egiga0 device
TFTP from server 192.168.1.171; our IP address is 192.168.1.109
Filename 'sp/kernel.bin'.
Load address: 0x900000
Loading: #############################################
###################
         #############################################
###################
         #############################################
###################
         #############################################
###################
         #############################################
###################
         #############################################
###################
         #############################################
###################
         #############################################
###################
         ##############
done
Bytes transferred = 2729908 (29a7b4 hex)
Marvell>> go 900000
## Starting application at 0x00900000 ...
```

· A computer (can be the same as in 3 above) that will act as a DHCP server and TFTP server.

## Getting and Building FreeBSD

If you've followed along with some of the articles from this magazine that deal with building NetBSD for an embedded system, you'll notice some immediate differences when you go to build FreeBSD. For example, NetBSD uses a build script that will run on virtually any Unix-like system. Partly this is because the script builds most of the ancillary software the NetBSD needs; as a result, it can take a lot longer to build NetBSD for an embedded system. With FreeBSD, on the other hand, you pretty much have to build it using a FreeBSD system. NetBSD's build scripts allow slightly better control over where the object and executables end up. I also had trouble getting FreeBSD to install the world files to a NFS-shared directory on another (Linux) machine and ended up using a FreeBSD system as the NFS server for the root system. All-in-all, the differences between the two systems are relatively small. My prediction – you'll forget about all of these once you see how much faster FreeBSD builds.

Like NetBSD, FreeBSD uses CVS to manage the souce code repository. Instead of using CVS directly, however, FreeBSD supplies an application *csup* that uses an external file to control how and where source code is downloaded. The control file for this project is shown in Listing 1. You need to change line 3 to reflect the CVS host you will be using for the download; see the instructions in lines 1 and 2. You'll need to change the directory in line 5 to show where you want the source code placed, and Line 9 says to download all of the source files, which you'll want. I've added a date tag to line 6 so that the downloaded files are from a date that we know will build and run correctly on the SP (but see the *Note*).

**Note**

When this article was written, support for SheevaPlug was still experimental. By the time you read this, SheevaPlug should be fully supported in FreeBSD 8; however, the build instructions were written to show what works, not what may (or may not) be available when you read this. The instructions, as written, will still work, however, you most likely won't need Lines 4 and 6, and the supfile you use in Line 2 should now be the standard supfile for FreeBSD 8. See the FreeBSD-arm mailing list (in the Resources section) for more details on current support features for SheevaPlug.

Listing 2 shows the commands that are necessary to retrieve the source files and to build both the kernel and the world (all of the files that FreeBSD needs to run that aren't actually part of the kernel). On a reasonably fast machine, this process will take a few hours, including the download. Line 2 is used to download the source code and Line 3-6 will get and apply the patch needed to build the SP kernel. Line 7 is used to control where the files are put during the build, while Line 11 controls where the root directory will be put during the build process. Note the difference between lines 7 and 11; my normal user shell is *bash*, while the root shell is *csh*.

The fact that part of the build steps requires you to change to the superuser is another difference between NetBSD and FreeBSD. Part of the reason for this is that FreeBSD sets the immutable bit on system files so they can't be inadvertently changed, which only the superuser can do. This also makes it more difficult (or impossible – I couldn't find a work-around) to install FreeBSD to a remotely-hosted NFS file system.

Once you have finished building FreeBSD, you will need to setup your local network to provide a DHCP server, a NFS server, and a TFTP server. Since I already had a DHCP server on a Linux machine, I didn't set one up for this project, but doing so is straight-forward. See the Resources section for information on how to do this on FreeBSD. Regardless of the system type, you'll need to make sure your `/etc/dhcpd.conf` file has the lines shown in Listing 3. Make sure the root directory you

**Listing 8.** Partial Listing Of The U-Boot Command Line Interface For The SheevaPlug

```
Marvell>> help
?      - alias for 'help'
base            - print or set address offset
boot            - boot default, i.e., run 'bootcmd'

cpumap          - Display CPU memory mapping settings.

dhcp  - invoke DHCP client to obtain IP/boot params
ext2load- load binary file from a Ext2 filesystem
ext2ls          - list files in a directory (default /)
fatinfo         - print information about filesystem
fatload         - load binary file from a dos filesystem
fatls           - list files in a directory (default /)
go   - start application at address 'addr'
help            - print online help
ls   - list files in a directory (default /)
map  - Diasplay address decode windows
md   - memory display
ping            - send ICMP ECHO_REQUEST to network host
printenv- print environment variables
reset           - Perform RESET of the CPU
resetenv          - Return all environment variable to default.
run   - run commands in an environment variable
saveenv         - save environment variables to persistent storage
setenv          - set environment variables
tftpboot- boot image via network using TFTP protocol
usb  - USB sub-system
usbboot         - boot from USB device
version         - print monitor version
```

put in the DHCP configuration file is the same location that you used on Line 11 of Listing 2. A common mistake is to have multiple DHCP servers on the network. If you have a router, you probably already have a DHCP server. Unfortunately, most routers can hand out addresses but can't handle the additional options you'll need for the SP, so you'll need to disable your router's DHCP capabilities and create your own DHCP server.

For the TFTP server, edit `/etc/inetd.conf` as shown in Listing 4 and uncomment Lines 30-31. Note that your line numbers may be slightly different. To enable the NFS server on FreeBSD, add the lines shown in Listing 5 to your `/etc/rc.conf` and create (or add to) the file `/etc/exports` as shown in Listing 6. Here, make sure you export the same directory as listed in your DHCP configuration file and that you used in Listing 2 to build the root directory on. Because of the immutable bit difficulty described above, I ended up building the root directory locally and then sharing that using a NFS server on FreeBSD.

## Booting the SheevaPlug

Now we're ready to boot up the SheevaPlug. If you haven't done so, you'll need to install the drivers for the SP's USB serial port, and you'll need a program

**Listing 9.** Partial Console Output From Booting FreeBSD On The SheevaPlug

```
Copyright (c) 1992-2009 The FreeBSD Project.
Copyright (c) 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994
        The Regents of the University of California. All rights reserved.
FreeBSD is a registered trademark of The FreeBSD Foundation.
FreeBSD 8.0-BETA2 #0: Sat Aug 22 21:01:49 EDT 2009
    hayford@freeqemu:/usr/home/hayford/obj/arm/usr/home/hayford/sp/src/sys/SHEEVAPLUG
Preloaded elf kernel "elf kernel" at 0xc0bb41c4.
CPU: Feroceon 88FR131 rev 1 (write-through core)
real memory  = 536870912 (512 MB)
avail memory = 520634368 (496 MB)
SOC: (0x6281:0x02) Marvell 88F6281 rev A0, TClock 200MHz
  Instruction cache prefetch enabled, data cache prefetch enabled
  256KB 4-way set-associative write-through unified L2 cache
mbus0: <Marvell Internal Bus (Mbus)> on motherboard
ic0: <Marvell Integrated Interrupt Controller> at mem 0xf1020200-0xf102023b on mbus0
timer0: <Marvell CPU Timer> at mem 0xf1020300-0xf102032f irq 1 on mbus0
rtc0: <Marvell Integrated RTC> at mem 0xf1010300-0xf1010307 on mbus0
gpio0: <Marvell Integrated GPIO Controller> at mem 0xf1010100-0xf101011f irq 35,36,37,38,39,40,41 on mbus0
uart0: <16550 or compatible> at mem 0xf1012000-0xf101201f irq 33 on mbus0
uart0: console (115740,n,8,1)
uart1: <16550 or compatible> at mem 0xf1012100-0xf101211f irq 34 on mbus0
uart1: fast interrupt
ehci0: <Marvell Integrated USB 2.0 controller> at mem 0xf1050000-0xf1050fff irq 48,19 on mbus0
usbus0: <Marvell Integrated USB 2.0 controller> on ehci0
mge0: <Marvell Gigabit Ethernet controller> at mem 0xf1072000-0xf1073fff irq 12,13,14,11,46 on mbus0
miibus0: <MII bus> on mge0
e1000phy0: <Marvell 88E1116R Gigabit PHY> PHY 0 on miibus0
usbus0: 480Mbps High Speed USB v2.0
bootpc_init: wired to interface 'mge0'
Sending DHCP Discover packet from interface mge0 (00:50:43:01:c4:7b)
mge0: link state changed to UP
Sending DHCP Request packet from interface mge0 (00:50:43:01:c4:7b)
Received DHCP Ack packet on mge0 from 192.168.1.171 (accepted) (got root path)
mge0 at 192.168.1.109 server 192.168.1.171 boot file /tftpboot/sp/kernel.bin
subnet mask 255.255.255.0 router 192.168.1.1 rootfs 192.168.1.171:/usr/home/hayford/sroot
Trying to mount root from nfs:
NFS ROOT: 192.168.1.171:/usr/home/hayford/sroot
start_init: trying /sbin/init
Enter full pathname of shell or RETURN for /bin/sh:
# mount
192.168.1.171:/usr/home/hayford/sroot on / (nfs, read-only)
devfs on /dev (devfs, local)
#
```

that runs on Windows that allows you to use the serial port. Regardless of the version of Windows, my favorite is PuTTY, available from *http://www.chiark.greenend.org.uk/˜sgtatham/putty/*. Plug the mini-USB connector into the SP and your Windows computer, start PuTTY, and use the serial port to login to the SP. Chances are that the SP will boot into Debian Linux long before you get PuTTY up and running, so you'll need to login to Debian (user: root, password: nosoup4u) and reboot the SP. At this point, you should see output similar to that shown in the top part of Listing 7. Hit any key when you see the command to stop the SP from booting back into Linux, and type a `?`, as shown at the top of Listing 8. You'll get a long list of available commands that U-Boot understands; generally, entering `? <command-name>` will get you more information on that particular command. At this point, we will only need to use three, `dhcp`, `tftpboot`, and `go`, as shown in the bottom half of Listing 7, but I left in some of the more interesting commands if you'd like to explore U-Boot a little. Once the kernel has been loaded (at address 0x900000 – another common error is to use the wrong load address), type in the `go 900000` command and you should see the FreeBSD bootup text scroll by as shown in Listing 9. When it's all finished you should be logged into the SP as root. From that point, I recommend that you setup a user and follow the normal steps to generate a useful FreeBSD system.

The second most common problem at this point is if the SP is unable to mount the NFS file system as it's root directory, usually because the NFS server isn't setup correctly or the DHCP server didn't tell the SP either the right network address or folder name. Towards the bottom of the bootup output (Listing 9), you'll see the message:

```
Received DHCP Ack packet on mge0 from
192.168.1.171 (accepted) (got root
path)
mge0 at 192.168.1.109 server
192.168.1.171 boot file /tftpboot/sp/
kernel.bin
subnet mask 255.255.255.0 router
192.168.1.1 rootfs 192.168.1.171:
/usr/home/hayford/sroot
```

If these lines don't look right or if you don't see the `got root path` message, check your DHCP configuration file and verify that the root directory is specified correctly. With NetBSD, you can set the kernel to ask for the location of the root device; convenient when you are starting out. FreeBSD isn't quite so flexible and the DHCP server must supply the correct root location for a NFS-mounted root system.

If you're a little adventurous, you can play around with the environment saved in the SP flash that is used by U-Boot to make the SP boot up into FreeBSD (using TFTP and NFS) automatically. But I'll leave that as an exercise for the reader.

### Getting Help

Like NetBSD, FreeBSD has a wiki that contains a lot of useful information on running and installing software, including FreeBSD itself. You can go to the wiki at *wiki.freebsd.org* for the latest information on ARM support. One difference, however, that will quickly become apparent, is that the FreeBSD wiki is for the benefit of developers, not users, and contributions are from a more limited population. This is not a criticism, just an observation. It's unusual to find people that like both developing and writing about software, so the wiki is often out-of-date or incomplete. The mailing list, like most, is extremely helpful and friendly, so go there with questions (after you've tried to work it out for yourself, first, of course). See the Resources section for more details.

## Conclusion

While FreeBSD doesn't yet support as many hardware configurations as NetBSD, the developer's are working hard to increase FreeBSD's credentials in the embedded world. In both the server and desktop world, FreeBSD has a significantly larger installation base than NetBSD, so that could be a real advantage to developers looking to add particular hardware or software items to their embedded system. The NetBSD build and development environment is a little more advanced and can be used on a variety of operating systems (BSD's, Linux, and Windows) while FreeBSD's requires a FreeBSD system. Still, this is not much of a limitation and FreeBSD has a mature and easy-to-use build environment. All in all, FreeBSD is an excellent choice for an embedded operating system and I, for one, look forward to using it on more systems in the future.

The SheevaPlug is an excellent ARM development system that you'll have a lot of fun with, particularly if you've not experienced FreeBSD on an embedded system. At present, there isn't support for the SheevaPlug flash memory in FreeBSd, but I predict that by the time you read this, you'll be able to boot your SheevaPlug as easily with FreeBSD as you can now with Linux.

### Resources

- Information on SheevaPlug hardware: *http://www.plugcomputer.org/data/docs/tech/SheevaPlug%20DevKit%20Reference%20Design-Rev1.1.pdf*
- The FreeBSD web site: *http://www.freebsd.org*.
- The FreeBSD developer's wiki: *http://wiki.freebsd.org/FreeBSDMarvell*.
- The FreeBSD-ARM mail list: *http://lists.freebsd.org/mailman/listinfo/freebsd-arm*.
- *http://www.bsdcan.org/2008/schedule/attachments/49_2008_uboot_freebsd.pdf*
- The description of the command line interface for U-Boot can be found here: *http://www.denx.de/wiki/view/DULG/UBootCommandLineInterface*. Note that this manual includes features not found in the SheevaPlug version of U-Boot.
- To setup a DHCP server on FreeBSD, see *http://www.freebsd.org/doc/en/books/handbook/network-dhcp.html*
- A combined serial port/SSH program for Windows is PuTTY: *http://www.chiark.greenend.org.uk/~sgtatham/putty/*

### Acknowledgments

Many thanks to Rafal Jaworowski of Semihalf for porting FreeBSD to the Marvell processors and for his support with this article. Rafal is a well-known developer for both FreeBSD and NetBSD though, of course, his talents are much broader than that. Learn more about Rafal and SemiHalf at *www.semihalf.com*.

FreeBSD Mall

## Your FreeBSD & PC-BSD Resource

www.FreeBSDMall.com

### FreeBSD 7.2 Jewel Case CD/DVD

**Set contains:**

- **Disc 1**: Installation & Live File System (for system recovery)
- **Disc 2**: Packages and Documentation
- **Disc 3**: Additional Packages
- **Disc 4**: More Packages

| | |
|---|---|
| FreeBSD 7.2 CD | $39.95 |
| FreeBSD 7.2 DVD | $39.95 |
| FreeBSD 6.4 CDROM | $39.95 |
| FreeBSD 6.4 DVD | $39.95 |

### FreeBSD Subscriptions

**Save time and $$$ by subscribing to regular updates of FreeBSD!**

| | |
|---|---|
| FreeBSD Subscription , start with CD 7.2 | $29.95 |
| FreeBSD Subscription, start with DVD 7.2 | $29.95 |
| FreeBSD Subscription, CD 6.4 | $29.95 |
| FreeBSD Subscription, DVD 6.4 | $29.95 |

### PC-BSD 7.1 DVD (Galileo Edition)

| | |
|---|---|
| PC-BSD 7.1 DVD | $29.95 |
| PC-BSD Subscription | $19.95 |

### BSD Magazine

| | |
|---|---|
| BSD Magazine | $15.00 |
| BSD Magazine Subscription | $11.99 |

### The FreeBSD Handbook

| | |
|---|---|
| The FreeBSD Handbook, Volume 1 (User Guide) | $39.95 |
| The FreeBSD Handbook, Volume 2 (Admin Guide) | $39.95 |
| Special: The FreeBSD Handbook, Volume 2 (Both Volumes) | $59.95 |
| Special: The FreeBSD Handbook, Both Volumes, & FreeBSD 7.2 | $79.95 |

### The FreeBSD Bundle

**Inside the Bundle, you'll find:**

- FreeBSD Handbook, 3rd Edition, Users Guide
- FreeBSD Handbook, 3rd Edition, Admin Guide
- FreeBSD 7.2 4-disc set
- FreeBSD Toolkit DVD

| | |
|---|---|
| Special: The FreeBSD CD Bundle | $89.95 |
| Special: The FreeBSD DVD Bundle | $89.95 |

### The FreeBSD Toolkit DVD .......... $39.95

### FreeBSD Mousepad .......... $10.00

### FreeBSD Caps .......... $20.00

### PC-BSD Caps .......... $20.00

For *MORE* FreeBSD & PC-BSD items, visit our website at FreeBSDMall.com!

CALL 925.240.6652 Ask about our software bundles!

t-shirts $18-$21.99

# Email server in FreeBSD

## Configuring FreeBSD as a mail server with Postfix and Dovecot in FreeBSD 7.X

Francisco Reyes

This tutorial is a step by step guide on how to setup your own mail server using Postfix as the Mail Transfer Agent(MTA) and Dovecot as the IMAP server and as the authenticating agent for Postfix. These instructions were tested with FreeBSD 7.2

In addition of Postfix and Dovecot we will also go over how to install Postgresql to store information that both Postfix and Dovecot will need to have simultaneous access to user information. Unless otherwise instructed all operations in this tutorial need to be performed as root.

We will need to use the port system. If you are new to it check chapter 4 of the handbook.

### Process overview

All programs in this artcile will be installed through the port system. The ports used for this article were Postgresql 8.4.0, Dovecot 1.2.4, Dovecot Sieve 1.2+0.1.12 and Postfix 2.6.3

This tutorial installs all ports in batch mode, however if you like you can remove the `BATCH=yes` and do the installs in interactive mode. Be aware that if you do the port install in interactive mode, any ports installed as dependencies will also be in interactive mode. The setup as described in this article uses virtual users so no users need to be created in the operating system to accept mail.

### Postgresql Port

Because it is a dependency for both Dovecot and Postfix let's install Postgresql first. If you already have Postgresql installed you can skip the port installation and only do the sql statements (see Listing 1). We need to create a database and the table. Create a file, mail.sql, with the following content (see Listing 2).

We will then load the file into postgres.

```
#psql -U pgsql -f mail.sql postgres
```

### Dovecot Port

To install this port we do as follows: see Listing 3.

Edit `/usr/local/etc/dovecot.conf` as follows: see Listing 4. Edit `/usr/local/etc/dovecot-sql.conf` as follows: see Listing 5. The directory `/usr/local/share/examples/dovecot/` contains examples of both files with considerable amount of useful information including explanations for all the parameters on the lines in this tutorial.

Create the log file for dovecot:

```
#touch /var/log/dovecot-deliver.log
#chown mailnull:mail /var/log/dovecot-deliver.log
```

### Postfix port

We install postfix last because it depends on both postgresql and dovecot.

```
#cd /usr/ports/mail/postfix
#make WITH_DOVECOT=yes WITH_TLS=yes WITH_PGSQL=yes WITH_
TEST=yes BATCH=yes install clean
#rehash
```

Disable sendmail,the default MTA, and enable postfix at startup by adding to your `/etc/rc.conf`

```
sendmail_enable="NO"
sendmail_submit_enable="NO"
sendmail_outbound_enable="NO"
sendmail_msp_queue_enable="NO"
postfix_enable="YES"
```

Disable some sendmail specific daily maintenance by editing /etc/periodic.conf and placing the following

```
daily_clean_hoststat_enable="NO"
daily_status_mail_rejects_enable="NO"
daily_status_include_submit_mailq="NO"
daily_submit_queuerun="NO"
```

Configure Postfix as the system mailer. Edit `/etc/mail/mailer.conf` as follows: see Listing 6. The Postfix configuration file is `/usr/local/etc/postfix/main.cf`. Change the entire main.cf as follows (see Listing 7). You also need to edit the `/usr/local/etc/postfix/master.cf` and add ad the bottom: see Listing 8.

Create the directory where we will hold the mail and give it the proper rights.

```
mkdir /usr/Mail/
chown mailnull:mail /usr/Mail/
```

As I stated on the `main.cf` note `/usr/Mail` can be replaced for whatever directory you want to use to hold your mail.

Edit `/usr/local/etc/postfix/virtual_maps` with your prefered editor and add users in the format

```
postmaster@mydomain.com
 mydomain.com/postmaster/Maildir
```

Convert the file into a database for faster lookup

```
postmap /usr/local/etc/postfix/virtual_
maps
```

For a fully documented main.cf see `/usr/local/libexec/postfix/main.cf`.

Before we can test our setup we need to populate the table passwd we created.

**Listing 1.** Postgresql installation from ports

```
#cd /usr/ports/databases/postgresql84-server
#make BATCH=yes install clean
#echo 'postgresql_enable="YES"' >> /etc/rc.conf
#/usr/local/etc/rc.d/postgresql initdb
#/usr/local/etc/rc.d/postgresql start
```

**Listing 2.** SQL statements to load into Postgres

```
CREATE USER mail password 'Pass2009';
CREATE DATABASE mail owner mail;
\c mail
CREATE TABLE passwd (
    id character varying(128) DEFAULT ''::character varying NOT NULL,
    crypt character varying(128) DEFAULT ''::character varying NOT NULL,
    clear character varying(128) DEFAULT ''::character varying NOT NULL,
    name character varying(128) DEFAULT ''::character varying NOT NULL,
    uid integer DEFAULT 26 NOT NULL,
    gid integer DEFAULT 6 NOT NULL,
    home character varying(255) DEFAULT ''::character varying NOT NULL,
    maildir character varying(255) DEFAULT ''::character varying NOT NULL,
    defaultdelivery character varying(255) DEFAULT ''::character varying
NOT NULL,
    quota character varying(255) DEFAULT ''::character varying NOT NULL
);

ALTER TABLE public.passwd OWNER TO mail;

ALTER TABLE ONLY passwd
    ADD CONSTRAINT id PRIMARY KEY (id);
```

**Listing 3.** Dovecot installation from ports

```
#cd /usr/ports/mail/dovecot
#make install WITH_PGSQL=yes WITHOUT_IPV6=yes BATCH=yes
#echo 'dovecot_enable="YES"' >> /etc/rc.conf
#cd /usr/ports/mail/dovecot-sieve
#make install BATCH=yes
#make clean
```

### The value of the encrypted password is Pass2009

As you need to create more users, if you don't have an easy way to generate a crypt value you can use *http://Stringsutils.com* Create a newalias file, kill sendmail, start postfix and dovecot. Could not start dovecot before because it will be using the postfix user which was not yet created when we finished creating the Dovecot port.

```
#newaliases
#/etc/rc.d/sendmail stop
#rehash
#/usr/local/etc/rc.d/dovecot start
#postfix start
```

Although the setup described in this article will not use the newalias file, Postfix looks for it. To test postfix is running:

```
telnet localhost 25
You should see a prompt
Trying 127.0.0.1...
Connected to < your host name >
Escape character is '^]'.
```

Use *CTRL+]*, the type quit

## Testing your setup

If you already have an IMAP client you can connect to your newly configured dovecot server using the test user. Remember that when login in you need use the fully qualified user name such as *postmaster@mydomain.com.* Also remember that you will need to use SMTP authentication to send mail through your server.

If you don't have an IMAP client there is a text based mail client, called cone, in the ports system which you can use for your test. You can install cone in the same server where you have intalled dovecot or on a different machine as long as it can connect to the dovecot IMAP server. Cone has several depencencies (gnupg, curl) and it takes quite a bit to compile so if you have a mail client installed it is best to use whatever you already have installed.

### Installing Cone

```
#cd /usr/ports/mail/cone
#make install clean
```

For the rest of the test you do not need to be the root superuser.

**Listing 4.** Dovecot configuration file

```
## Dovecot configuration file
# If you're in a hurry, see http://wiki.dovecot.org/
QuickConfiguration

protocols = imap pop3
disable_plaintext_auth = no
shutdown_clients = yes
ssl = no
login_process_size = 64
mail_location = maildir:/usr/Mail/%d/%n/Maildir
mail_privileged_group = mail

# Rely on O_EXCL to work when creating dotlock files.
NFS supports O_EXCL
# since version 3, so this should be safe to use
nowadays by default.

dotlock_use_excl = yes

verbose_proctitle = yes
first_valid_uid = 26
last_valid_uid = 26

first_valid_gid = 6
last_valid_gid = 6
maildir_copy_with_hardlinks = yes

## IMAP specific settings

protocol imap {
  imap_client_workarounds = delay-newmail netscape-eoh
tb-extra-mailbox-sep
}

## POP3 specific settings
protocol pop3 {
  pop3_uidl_format = %08Xu%08Xv
  pop3_client_workarounds = outlook-no-nuls oe-ns-eoh
}

## LDA specific settings

protocol lda {
  # Address to use when sending rejection mails.

  postmaster_address = postmaster@example.com
  mail_plugins = sieve
  sendmail_path = /usr/sbin/sendmail
  log_path = /var/log/dovecot-deliver.log
  info_log_path = /var/log/dovecot-deliver.log
}

# Log unsuccessful authentication attempts and the
reasons why they failed.

auth_verbose = no

# Even more verbose logging for debugging purposes.
Shows for example SQL
# queries.

auth_debug = no

# In case of password mismatches, log the passwords and
used scheme so the
# problem can be debugged. Enabling this also enables
auth_debug.

auth_debug_passwords = no

auth default {
  mechanisms = plain

  passdb sql {
    args = /usr/local/etc/dovecot-sql.conf
  }

  userdb passwd {
    args = blocking=yes
  }

  userdb sql {
    args = /usr/local/etc/dovecot-sql.conf
  }

  user = root

  socket listen {
    master {
      path = /var/run/dovecot/auth-master
      mode = 0660
      user = mailnull
      group = mail
    }
    client {
      path = /var/run/dovecot/auth-client
      mode = 0660
      user = postfix
      group = mail
    }
  }
}
dict {

  #quota = mysql:/usr/local/etc/dovecot-dict-quota.conf
  #expire = db:/var/db/dovecot/expire.db

}

## Plugin settings
plugin {
}
```

- Start the cone program by typing 'cone' from the command prompt.
- Hit 'M' for the main menu.
- Select 'N' for new account.
- Select 'I' for IMAP
- Type a descriptive name for the account name.
- Type IP or DNS name of the machine in the Server field. Add /novalidate-cert at end. Needed since cone looks for SSL by defaultEnter username including domain in the Login field
- Enter password in Password field.
- Select Inbox folder.

Once you are able to connect to the IMAP server with your own client or with

## On the 'Net

- Dovecot site: *http://dovecot.org/*
- Postfix web site: *http://www.postfix.org/*
- Postgresql site: *http://www.postgresql.org/*
- Cone web site: *http://www.courier-mta.org/cone/*

**Listing 5.** Dovecot SQL configuration file

```
driver = pgsql
connect = host=localhost
dbname=mail user=mail
password=Pass2009
default_pass_scheme = CRYPT
password_query = SELECT crypt as
password FROM passwd WHERE id =
'%u'
user_query = SELECT home, uid, gid
FROM passwd WHERE id = '%u'
```

**Listing 6.** Mailer.conf for Postfix

```
#
# Execute the Postfix sendmail
program, named /usr/local/sbin/
sendmail
#
sendmail        /usr/local/sbin/
sendmail
send-mail       /usr/local/sbin/
sendmail
mailq           /usr/local/sbin/
sendmail
newaliases      /usr/local/sbin/
sendmail
```

**Listing 7.** Postfix configuration file

```
queue_directory = /var/spool/postfix
command_directory = /usr/local/sbin
daemon_directory = /usr/local/libexec/postfix
html_directory = /usr/local/share/doc/postfix
manpage_directory = /usr/local/man
sample_directory = /usr/local/etc/postfix
readme_directory = /usr/local/share/doc/postfix
sendmail_path = /usr/local/sbin/sendmail
newaliases_path = /usr/local/bin/newaliases
mailq_path = /usr/local/bin/mailq
data_directory = /var/db/postfix
debug_peer_level = 2
debugger_command =
      PATH=/bin:/usr/bin:/usr/local/bin:/usr/X11R6/bin
         ddd $daemon_directory/$process_name $process_
id & sleep 5
setgid_group = maildrop
mail_owner = postfix
myorigin = $myhostname
relay_domains = $mydestination
local_recipient_maps = $virtual_mailbox_maps
unknown_local_recipient_reject_code = 550
mynetworks_style = host
mydestination = localhost, localhost.$mydomain

#/usr/Mail arbitrarily chosen. Pick a directory of
your choice.
virtual_mailbox_base = /usr/Mail/
dovecot_destination_recipient_limit = 1
virtual_transport = dovecot

#Multiple domains for virtual_mailbox_domains can be
placed
# comma separatedor use a file like I show you below
for virtual_mailbox_maps
virtual_mailbox_domains = mydomain.com
virtual_mailbox_maps = hash:/usr/local/etc/postfix/
virtual_maps
virtual_uid_maps = static:26
virtual_gid_maps = static:6
message_size_limit=20480000
bounce_queue_lifetime = 2h

smtpd_delay_reject = yes
smtpd_helo_required = yes

smtpd_sender_restrictions =
                permit_mynetworks
    permit_sasl_authenticated
                reject_non_fqdn_sender
                reject_unknown_sender_domain
                reject
smtpd_recipient_restrictions =
                permit_mynetworks
                permit_sasl_authenticated
                reject_unauth_pipelining
                reject_invalid_hostname
                reject_non_fqdn_recipient
                reject_unknown_reverse_client_hostname
                reject_unknown_recipient_domain
                reject_unauth_destination

smtpd_client_restrictions =
                permit_mynetworks
                permit_sasl_authenticated
                reject_unauth_pipelining

######   SASL Authentication    #####
broken_sasl_auth_clients = yes
smtpd_sasl_auth_enable = yes
smtpd_sasl_type = dovecot
smtpd_sasl_path = /var/run/dovecot/auth-client
smtpd_sasl_security_options = noanonymous
```

**Listing 8.** Postfix changes to master.cf

```
# Dovecot LDA
dovecot unix    -     n     n     -     -     pipe
  flags=DRhu user=mailnull:mail argv=/usr/
local/libexec/dovecot/deliver -f ${sender} -d
${user}@${nexthop} -n -m ${extension}
```

**Listing 9.** Adding a user to the database for mail

```
#psql -U pgsql mail
insert into passwd (id,crypt,home)
values
('postmaster@mydomain.com','$1$ivtiEVV9$kpnG/
pBBWm6wNJ.Pe7qgr1','/usr/Mail/mydomain.com/
postmaster');
```

cone you then need to try sending an email to it. If the machine is the machine responsible for handling email for the given domain (i.e. DNS MX records point to it) you can send an email to your test user from any machine. *If the IMAP server doesn't yet has MX records pointing to it, you can still test your setup by setting the machine you configured as your Cyrus server as the SMTP server. If using cone from the same machine you don't need to do anything extra. Just write an email and send it to your test user. By default cone will use the current machine as the delivery SMTP server.*

## Additional notes

The way we configured postgresql for this tutorial allows any user to connect to the server. You need to edit your pg_hba.conf to make it secure. See *http://www.postgresql.org/docs/8.4/interactive/auth-pg-hba-conf.html.*

The sieve plugin we installed allows you to setup filtering at the imap level. See *http://wiki.dovecot.org/LDA/Sieve* for instructions on how to use sive.

EXOnetric

# Reliable FreeBSD Jails and hosting at the heart of the UK Internet

**Find out what we can do for you today...**

**Exonetric Consulting Ltd.**
**Tel.** +44 (0) 870 787 9394
**Fax.** +44 (0) 870 787 9395

www.exonetric.com
info@exonetric.com

# Monitoring OpenBSD with Symon

Matthias Pfeifer

Once you have your OpenBSD Server running, you might want to monitor your machine. There are several ways to do this and there is a large amount of tools you could use for it.

One of these tools I will show you in this *how-to* article is symon. Symon is very easy to install and, once the setup is done, it will provide useful status information of your system. One of the greatest benefit is the graphical presentation and its very simple configuration. Because of the design of symon, you can use symon in a large network environment also. And of course, symon will work on all other BSD's too. Our symon setup comes in two parts: symon and syweb. The symon package contains symon and symux, which are used to collect and prepare the collected information. Syweb is used to display the collected data.

Note
Syweb needs PHP. So make sure that you have PHP installed.

## Installing symon

On OpenBSD, we are in luck, because there are symon and syweb packages available (make sure that your PKG_PATH variable is set (for e.g. export *PKG_PATH=ftp://ftp.openbsd.org/pub/OpenBSD/4.5/packages/i386/*). Then you can just type `pkg_add symon` in your console. For the further configuration it is better to have syweb already installed: (see Listing 2).

## Configuring symon

I will show you a small symon setup here, so we do not change any paths for syweb here. The following configuraion is very easy to extend. See the manpages of symon and symux (man symon, man symux). We start our configuration with the following files:

- `/etc/symon.conf`
- `/etc/symux.conf`

For the beginning, we want monitor just the first CPU (cpu(0)) and the RAM (mem) (Look at the *Data formats* section in the symon manpage for more monitoring targets). Ok, let's start and add this to `/etc/symon.conf`

```
monitor
{
     cpu(0),
     mem
} stream to 127.0.0.1 2100
```

As you can see, we will stream the collected data to localhost. However, you can stream these data to another monitoring station. Just enter the machines IP here.

Now, add the following lines to `/etc/symux.conf` (see Listing 3).

The source section is set for every host which should be monitored. The source section in `symux.conf` is similar to the monitor section in `symon.conf`

**Listing 1.** Installing symon

```
# pkg_add symon
libart-2.3.20p0: complete
rrdtool-1.2.30: complete
symon-2.78: complete
--- symon-2.78 -------------------
Example configurations for both symon and symux have
been installed
in /usr/local/share/examples/symon.
RRD files can be obtained by running
/usr/local/share/symon/c_smrrds.sh
Read the LEGACY section of symux(8) for information
about
migrating RRDs from a previous symux version.
```

Next, we need the datadir directory for our data:

```
mkdir /var/www/symon/rrds/localhost
```

**Listing 2.** Installing syweb

```
# pkg_add syweb
syweb-0.55p1: complete
--- syweb-0.55p1 -------------------
syweb's default install assumes that:
  - apache is chrooted at /var/www
  - rrdtool is installed in the chroot
  - symux rrd files are kept in /var/www/symon/rrds/HOST/*.rrd

rrdtool can be installed in the chroot using
  /var/www/symon/install_rrdtool.sh

Customise /var/www/htdocs/syweb/setup.inc if these assumptions are
incorrect.
```

**Listing 3.** Configuring /etc/symux.conf

```
mux 127.0.0.1 2100
source 127.0.0.1
{
   accept
   {
      cpu(0),
      mem
   }
   datadir "/var/www/symon/rrds/localhost"
}
```

**Listing 4.** Creating rrd files

```
/usr/local/share/symon/c_smrrds.sh /var/www/symon/rrds/localhost/cpu0.rrd
/usr/local/share/symon/c_smrrds.sh /var/www/symon/rrds/localhost/mem.rrd
```

**Listing 5.** Checking processes

```
# ps -waux | grep sym
root     25000  0.0  0.0   520  1020 ??  Is     6:13PM    0:00.02 /usr/
local/libexec/symux
_symon    5334  0.0  0.0   308   844 ??  Ss     6:19PM    0:00.00 /usr/
local/libexec/symon
```

**Listing 6.** Staring services at boot time

```
if [ -x /usr/local/libexec/symux ]; then
  echo -n ' symux';
  /usr/local/libexec/symux
fi

if [ -x /usr/local/libexec/symon ]; then
  echo -n ' symon';
  /usr/local/libexec/symon
fi
```

**Note**
Each source section need its own datadir!
Symon ships with some useful shell scripts which makes the configuration much easier. You can found the scripts in `/usr/local/share/symon/`.

One of the shipped scripts is used to generate the needed rrd files. We need to create two rrd files for our monitoring objects (cpu(0) and mem) (Listing 4).

Now, it's time to start and check the services:

```
# /usr/local/libexec/symux
# /usr/local/libexec/symon
```

When you check the services, you should get an output similar to the following: see Listing 5.

**Note**
Be sure that you start symux at the first. Otherwise symon will not provide any data to symux.

In most cases, we want to start services at boot time. So we add the following lines to `/etc/rc.local` (see Listing 6).

Now we have symon and symux running. That is fine but not really useful for us because we like a graphical presentation of our system statistics.

If you are running apache chrooted in a default setup, all you need is to point your browser to *http://localhost/syweb/*. If you have some other individual configurations, you should adjust your configuration (for e.g. symlink the syweb directory into a appropriate location and configure a virtual host).

## Caveat

There a one well known issue, when symon starts. It could happen that you receive the following message in your logs:

```
symux: could not get a semaphore
```

We need to do a little sysctl tuning (The values are just a advise. Feel free and figure out the best setup for your environment).

```
sysctl -w kern.seminfo.semmni=256
sysctl -w kern.seminfo.semmns=1024
```

To setup these sysctl values, add the following lines to `/etc/sysctl.conf`

```
kern.seminfo.semmni=256
kern.seminfo.semmns=1024
```

Visit *http://www.xs4all.nl/~wpd/symon/index.html* vor additional information.

# BSD as the Platform for Operationalizing Organizational Flexability via a Data Concourse

Richard C. Batka

A major change is about to take place in large organizations worldwide and BSD is positioned perfectly to play a starring role.

Long known for its rock solid stability and reliability in the airline and banking industries, BSD will be used by organizations to build information interchanges. Building on a reliable BSD infrastructure will be the key to operationalzing flexibility and the future standard platform for *Operationally Aware Vector Adjusting Application* (OAVAA) environments.

To meet increased market demand for products and services, large organizations today rely on a complex web of interconnected business relationships with partners, vendors, and suppliers. When unexpected delays occur it creates chaos.

These problems are addressed by creating custom process (called one off exception processing) and as all of you know from your administration/infrastructure experience, anytime you implement sudden/drastic changes throughout the operational ecosystem you consume valuable resource cycles which inevitably prevent operational flexibility.

We are constantly seeking better returns on our technology expenditures (investment) and we are always looking for ways to optimize current service/delivery capability based on new or improved business process (by attempting to integrate information from multiple functional areas within the organization) and for the most part, this is a process that takes time, requires multiple approvals and numerous man hours to complete and speed to completion is always an issue. Typically these services are built on service platforms that span the organization.

## Fact

Over 70 percent of organizations that have invested in enterprise systems have not received the promised benefit on schedule (or) have invested more money than originally anticipated.

## BSD in the Enterprise

Imagine if you will a BSD based data concourse service that enables organizations to achieve a high level of flexibility and the ability to quickly integrate change at an enterprise wide level. This can be achieved by effective communication through an information interchange specifically built on the BSD platform, designed to support the automatic, cross boundary capability to create, change, and modify processes.

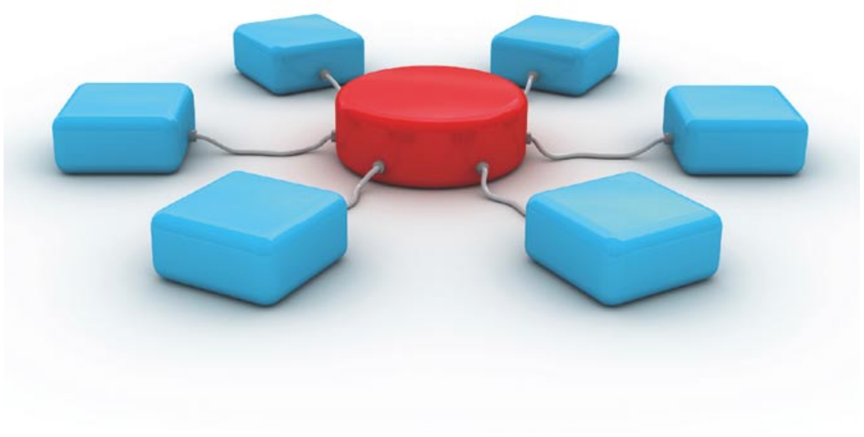
It's a movement that's building momentum today and it's something you need to prepare for in the next 18-24 months. Thousands of organizations worldwide will build out this new capability and quickly discover that they have a distinct market advantage.

## BSD has the Flexability

To provide a reliable infrastructure for applications that are data concourse service aware and ready to facilitate operations among vendors, customers, and suppliers by aggregating discreet data elements, structuring it as information, and providing push up reports with wide visibility.

This new environment will also allow firms to achieve economies of scale upon the creation of a data concourse service that provides clients and their partners the use of standardized operational applications connecting to standardized business management applications which will have the support functions to enact real time process change at an operational level.

The goal is to enable effective, real time information interchange by creation of a data concourse service which will promote the aggregation of data from internal and external sources, correlate it, and then make it available for immediate automated process/rule creation or further analysis. Additionally, it dose something that has never successfully been achieved before at the enterprise level:



connect operational applications and business management applications.

## Creating these Connections is the key to Organizational Flexability and Ultimately Compeditive Advantage

Through effective information interchange (and the associated automatic rule/process creation that results) between organizational levels/business units within the organization, organizations will be able to achieve something never thought possible – automatically creating process connections between strategy and operations, so that changes are automatically incorporated on either side. To achieve this, an organization must have the proper combination of BSD based infrastructure, event aware applications, and flexibility.

## What is BSD Based Information Interchange?

Information interchange enables a clear separation of duties between applications that support standard operations and those applications that require flexibility to handle changes. BSD is platform of choice to support the next generation of aware applications that allow for the successful resolution of semantic differences between unstructured and structured data in use by applications today.

I call the next generation applications: *Operationally Aware Vector Adjusting Applications* (OAVAA). These applications will provide organizational leadership with what's been lacking today, namely real-time shared visibility in aid of effective decision making and ability to automatically react to deltas discovered between operational and business management applications upon synchronization. These interchange features provide the balance needed for enterprise flexibility.

The elements to this approach are:

- Enterprise based, process enabled, operational & business management applications
- Information interchange (based on a robust BSD environment) which facilitates the connection between areas (example: operations and business)
- Audit trail capability
- Multi-site deployment capability with failover & backup capability
- Straight forward licensing model

Operational applications designed for efficiency are insulated from needing to be changed frequently. Business management applications are flexible and easier to change; they support collaboration, analysis, and decision support so to achieve flexibility, the enterprise must have the proper combination of hard and fast business rules and openness to change.

## Flexability?

All enterprises want to be flexible. They want the ability to change with current market conditions however many confuse flexibility with speed to market. They find themselves failing because of the inability to make changes related to those current market conditions. Meanwhile, others are so intent on effectiveness that they throw a blind eye to market conditions.

## It's a Moving Target

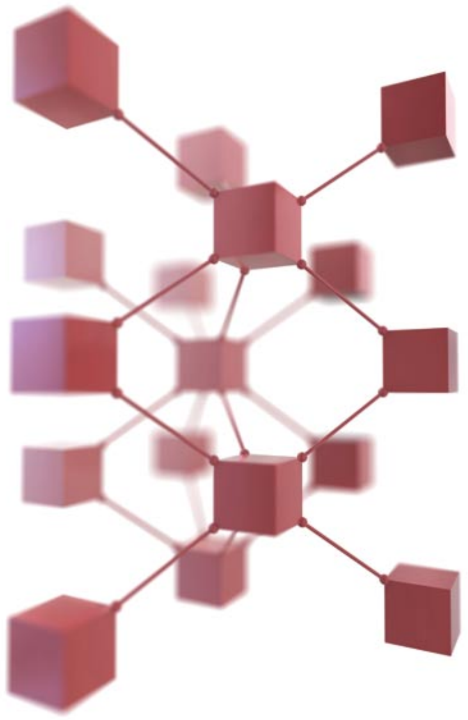
All companies have strategy and all companies have operations. Prevailing wisdom would say that changes in the marketplace should lead to changes in strategy and operations – but this is never the case.

### Fact

Seven of eight large enterprises failed to meet self imposed growth (profitability) targets.

### Fact

More than 95 percent of employees are unaware of or do not understand their enterprise strategy.



## Self-evident Disconnect

Strategy and operations are not communicating effectively. Strategy prepares for the future and chases flexibility, but operations are rigid and designed to be consistent. Strategy requires extreme flexibility in making choices and changes frequently, while operations are complicated and take time and money to change. There is a disconnect between strategy and operations. The disconnect is the roadblock – Why? Because changes in strategy are not reflected in operations fast enough.

Over the years organizations have spent millions of dollars trying to synchronize the two – by investing in enterprise resource planning (ERP), operational support systems, decision support systems, performance management tools, analytics, and dashboards. Talk to an executive on the golf course about these systems – They will be happy to share tales of frustration regarding massive integration contracts, mediocre results, and uncomfortable silences during board meetings. For real excitement, try asking a front line manager *after* they spent all week in training on these systems?

### Fact

Employees at every level of the enterprise spend the majority of their day finding information, without consideration (or) time to care for what it means.

## In the Mean Time

While the applications play catch up, we can prepare by spending our energy building an information interchange. Most applications will adapt but for the ones that don't, we can offer access to the interchange. The Information interchange will play two key roles:

- Manage the semantics so applications can share and integrate information in a plug-and-play manner.
- Operationalize the connection between strategy and operations so that respective applications are able to stay in sync.

This is operationalizing flexibility.

### Warning

Many of today's applications in the operational space play to key a roll in saying *no, your organization can't afford to make that change*, because the created financial projection says you can't afford to change the application.

## Alternative Flexability

Externally to the organization you should accept that change happens so look to build solutions such as information interchanges that offer services to the strategy and operations groups within the organization through the creation of a dedicated BSD based data concourse service infrastructure.

BSD is a stable platform to build any type of transactional processing system. Handle changes in a data concourse layer that resides between business applications and operational applications. Keep in mind that changes occur all over the enterprise: product development, sales, customer support, IT, marketing, and all other functions in an enterprise that operate in an environment of increasing change.

## Historicaly Speaking: Strategy and Operations Differ in Key Ways

Processes closer to strategy are semi structured and constructed on the fly. Processes in operations are clearly defined. The processes that live closer to strategy use structured and unstructured inputs, whereas operations rest on structured data supplied by reliable data sources.

### Questions to Ask

- What processes exist today that touch both strategy and operations?
- How does your organization tell the difference between structured process and a semi-structured process requests?
- Is the leadership team able to see change coming based on top level reports produced by the organization?
- Can front line managers make changes to operations easily?

Take for example enterprise dashboards that monitor a myriad of organizational performance metrics. They provide valuable information to the people that need to know. However, do you have access to the systems required to modify (change) activities without disruption to the enterprise? Probably not.

## Case Study: BSD Network Management Tools

The majority of tools in use today deploy some type of agent to the end node (let's call that a type of business management application) which is done to accept or decline patches and code drops to the end node – a type shielded information interchange that exists between the management node, patch server, reporting server, and end node. If we scale this architecture to the larger business functions of the organization for the singular purpose of increased flexibility, we will see that we can make changes to operational process and implement them with minimal disruption.

## Case Study: BSD Security Patches

Applying security patches to your BSD environment is an important part of maintaining computer software, especially the operating system. For the longest time on FreeBSD, for example, this process was not an easy one. Patches had to be applied to the source code, the code rebuilt into binaries, and then the binaries had to be re-installed. Today you can use a utility called freebsd-update. This is an example of a service that can be offered at the information interchange through the data concourse service.

This utility provides two separate functions. First, it allows for binary security and errata updates to be applied to the FreeBSD base system without disruption (the build and install requirements). Secondly, the utility supports minor and major release upgrades (again with minimal disruption.)

### Tip

Use the cvsup command to obtain and update FreeBSD sources. To use it, you will need to install a port or package like net/cvsup-without-gui. If you are using FreeBSD 6.2-RELEASE or later, you may wish to substitute this with csup(1), which is now part of the base system.

### References

- The FreeBSD Project *www.freebsd.org*
- OpenBSD *www.openbsd.org*
- NetBSD *www.netbsd.org*
- DragonFly BSD *www.dragonflybsd.org*
- *www.InterviewTomorrow.com*
- Operational Tempo BBS: telnet: *//operationaltempo.com*

## Implementation

You know that organizations are inherently resistant to change so your level of success will be determined in large part by your approach. A well thought out plan is required. Take the following implementation approach for organizations with up to 30,000 employees.

- Pilot 1-500 people
- ROI (checkpoint)
- Larger Group
- ROI (checkpoint) 15,000 people
- Organization Wide
- ROI (checkpoint) 30,000 people

## Conclusion

Business is constantly experimenting with new strategies to take advantage of change while minimizing its disruptive effects. The business environment will always be changing and you have the opportunity to build new environments in support of operationalizing flexibility by creating real time, adaptive connections between business units.

This new approach goes beyond simple Enterprise Application Integration EAI, Service Platforms, and Business Process Management. It's a call for a complete rethinking of the connections that exist between and within every core *group* within the organization. Any link that exists between two or more critical business functions is fair game for this new thinking.

Enterprises that can leverage applications to create a balance between standardization and flexibility (and then operationalize flexibility) will have a unique competitive advantage which will allow them to dominate in the marketplace. This is clearly an opportunity for you to be on the forefront of this seismic shift that is about to take place while unleashing the true power of BSD. If you help establish these architectures; the ones that help organizations make this transformation smoothly will most definitely benefit at promotion/bonus time.

### About the Author

Richard C. Batka has held various management and engineering positions with Microsoft, PriceWaterhouseCoopers, Symantec, ThomsonReuters, and JPMorgan Chase. He has spent the last 17 years devoted to the complex issues of enterprise application development, security, infrastructure, data management and regulatory compliance. A graduate of New York University he holds numerous industry certifications. Currently, Mr. Batka is the CEO of a privately funded consulting services firm in New York that provides strategy and engineering services internationally. Mr. Batka can be reached at rbusa1@gmail.com.

# Living The PC-BSD Lifestyle

James T. Nixon III

Some people are Mac, some are Windows, I am PC-BSD. PC-BSD is more than an operating system, it's a lifestyle.

Sitting next to my 47" Westinghouse LCD TV is the iXsystems Apollo Workstation. This workstation is powered by the 5500 series of the Intel Xeon processor, an Asus GeForce 9800 GT video card, and 4 gigs of RAM. It came with PC-BSD Galileo Edition (7.1) pre-installed and a handful of applications that immediately increased my quality of life tenfold. Using free software instead of spending hundreds, or even thousands of dollars on commercial software is great, especially because I enjoy dabbling in Photoshop, FL Studio, Sony Music Studio, as well as playing games such as Left 4 Dead, Half-Life 2, and Eve Online.

First things first, can I play my favorite games? The answer (for me) is absolutely! I am a huge fan of Valve and their Steam client because I tend to scratch or lose CD's. I created my Steam account in 2004 when I purchased Half-Life 2, although I never finished the game because my computer could not handle it. 45 minute loading screens do not work for me... So, I forgot about Valve for a while and moved on to other hobbies, namely music and web design. When I was away from my drumset, I was on my PC hacking away at local band websites, doing photomanipulation in Photoshop, or creating Classic Nintendo remixes in Sony Music Studio (formerly Acid Pro). I was also dual-booting random Linux distros (with much displeasure), because I got tired of the constant degredation of performance on my Windows box. I found that I couldn't enjoy most of my computer related hobbies on Linux, and worse yet, most of the websites I was developing or visiting didn't work or look the same. So I forgot about Linux for awhile, too.

Enter PC-BSD, one desktop to rule them all! The PC-BSD operating system truly changed my life. No more Windows, no more Linux, and all (okay most) of my hobbies intact. I replaced Photoshop with GIMP, Sony Music Studio with Ardour, and Dreamweaver with Bluefish. The transition from Windows to PC-BSD was fairly easy. Adapting to a new collection of programs and bugs was the

hardest part, but didn't stop me from pursuing a Windowless lifestyle. I am not against commercial software, I just prefer to spend money on open source software, hardware, and video games. This is where Valve comes back into the picture.

## PC-BSD is for Gamers

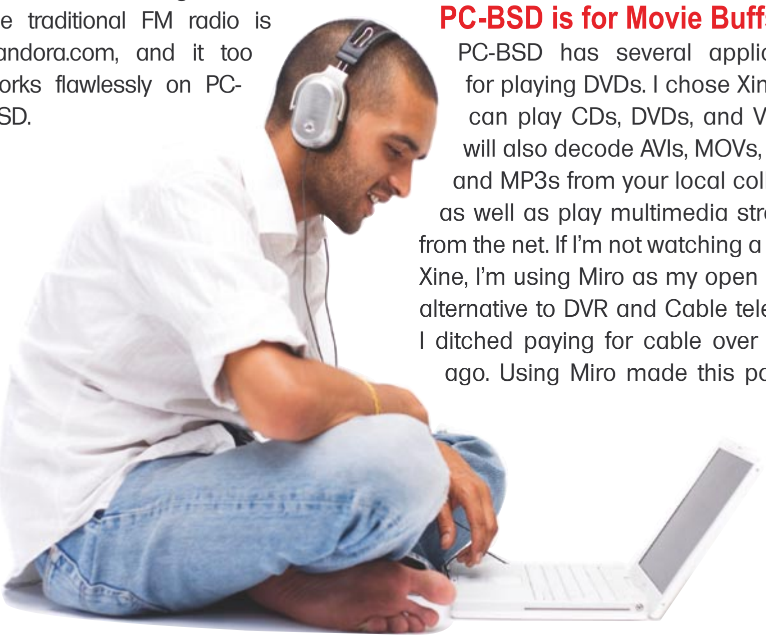
One lazy afternoon, I was bored and thought I'd download the Steam client from steampowered.com and install it on PC-BSD. PC-BSD comes with Wine, so Steam installed without any problems. When I opened Steam and entered my account details, all the games I purchased in 2004 were waiting for me to install. I was feeling pretty lucky at this point, so I chose to install Half-Life 2 first. An hour or so later I launched Half-Life 2. There was only one problem, sound did not work. I was a little sad, but I turned captioning on and played for a minute, saved, loaded, and then quit. After a minute of searching on winehq.org I found out that all I needed to do was set the sound acceleration in winecfg to Emulation.

After doing that, I launched Half-Life 2 again. This time sound worked perfectly. Amazed at how beautiful the game looked, I pressed my luck and maxed out the graphic settings to include full bloom and reflection. Victory! To see what the FPS was, I opened the developer console and typed 'cl_showfps 1' and the result was a steady 300 frames per second. Simply amazing. After playing Half-Life 2 for a few hours I hopped on Deathmatch and CounterStrike: Source. Both worked flawlessly. Pretty pleased with Valve at this point, I went to *http://store.steampowered.com* and purchased Left 4 Dead which also worked flawlessly on PC-BSD. The next Steam game I tried was Overlord II, which didn't work at all, but Assassin's

Creed from Ubisoft played wonderfully. In the end, 4 of the 5 games I tested ended up playing better on my PC-BSD machine than any Windows box I have ever owned.

## PC-BSD is for Music Lovers

Life is not all fun and games. You need to mix things up with music too. There are many choices for audio players on PC-BSD. My personal choice is Amarok. Amarok has an easy to use and intuitive interface and comes with great features like displaying lyrics, downloading album art, and connecting to your Last.fm account, just to name a few. If you have a Last.fm account, you will also enjoy the Last.fm PBI on pbidir.com. The Last.fm application is an easy to use radio alternative. I tend to use it at work when I get bored of my local collection. Another great alternative to the traditional FM radio is Pandora.com, and it too works flawlessly on PC-BSD.



Pandora allows you to create a radio station based on your personal tastes. As you give songs a *thumbs up* or *thumbs down*, Pandora takes into account several attributes and plays similar songs. For example, the acoustic version of Creep by Radiohead has pop rock qualities, acoustic sonority, repetitive melodic phrasing, major key tonality, and a dynamic male vocalist. Mix this with say, Cannibal Corpse, and you'll have a unique blend of music in constant rotation. After listening to music for awhile I tend to get the musician's itch. If this happens and I don't have a band to jam with, I open up Ardour and start recording, editing, and mixing my own music. Ardour is very similar to Cubase, Nuendo, Adobe Audition, etc... But like any program, it has its quirks. After a few hours of use you'll feel right at home. Ardour does multichannel recording, non-linear, non- destructive region based editing with unlimited undo and redo capabilities. It also features full automation support, an amazing mixer, and plenty of plugins to tweak and shape sound to your heart's content. I've had it crash a few times, but I found that turning off auto-crossfade solved this problem. In the near future I am going to set up a completely open source recording studio for my fellow musical geeks and I to create "open music" for the masses.

## PC-BSD is for Movie Buffs

PC-BSD has several applications for playing DVDs. I chose Xine. Xine can play CDs, DVDs, and VCDs. It will also decode AVIs, MOVs, WMVs, and MP3s from your local collection, as well as play multimedia streamed from the net. If I'm not watching a DVD in Xine, I'm using Miro as my open source alternative to DVR and Cable television. I ditched paying for cable over a year ago. Using Miro made this possible.

I have all my favorite television shows auto-download as they are released using various RSS torrent feeds. Miro can play most video files and offers over 6,000 free internet TV shows and video podcasts. Watching Lost in HD on my 1080p 47" television is a wonderful experience. Occasionally, I want to watch a TV show instantly. That's when I go to Hulu.com. Hulu.com is an amazing site that streams HD television shows and movies over the internet using Flash. All of this is done while I am lounging on my couch using a wireless mouse and keyboard on my coffee table. And if watching movies isn't enough, editing video is a snap with Kdenlive. Kdenlive is a non-linear video editor for PC-BSD that is designed for basic or semi-professional video editing. It supports DV, AVCHD (which is considered experimental), and HDV editing. There are other video editors out there, but Kdenlive was the easiest to get the job done.

## PC-BSD is for Everyone!

Whether you're a gamer, music connoisseur, movie enthusiast, or all of those, PC-BSD is the operating system for you. For more information or to download PC-BSD, visit *http://pcbsd.org*. To download PC-BSD software, visit *http://pbidir.com*.

# Tips and tricks

BSD Tips&trics by Dru Lavigne

In this issue of BSD Tips and Tricks, readers share some of their favourite tips for solving problems and saving time.

## Keep /home Intact During OpenBSD Upgrade

Denny White of OpenBSD101 has several tips available at *http://polarwave.openbsd101.com*. One of his favourite tricks shows how to keep the `/home` partition intact during an in-place upgrade.

NOTE: Before any upgrade, always backup your data first, just in case!

The gist of this trick as that you tell disklabel to ignore the `/home` partition so it is not reformatted during the upgrade. To do this, go through the normal install routine until you get to the disklabel section. If you accept the defaults for each partition, your screen will look something like the output in Figure 1.

Note that the default is to press enter for each partition, meaning each partition will be formatted. Instead, you want to type in the word *none* when you get to the `/home` partition so it looks like Figure 2 instead.

You can then continue through the installation as usual.

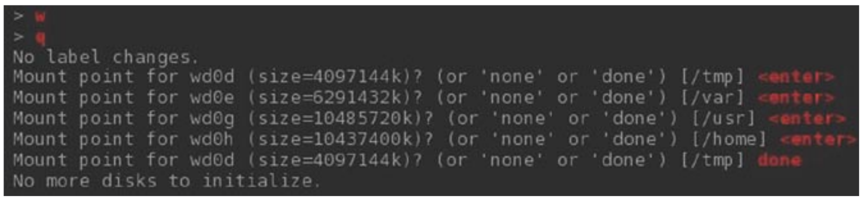
Once the installation is complete, the upgraded system won't be aware that you have an existing /home partition that you would like to mount at boot time. You can fix this by editing /etc/fstab to re-add your `/home` partition.

## cd to an Unknown Directory

Jan Schaumann of netmeister.org offers the following tip. To let the shell figure out where the package to install is, type:

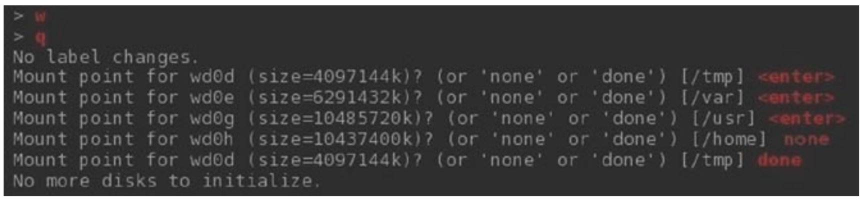
```
cd /usr/pkgsrc/*/package
```

This logic works on any system. Figure 3 shows example output from a PC-BSD system. In this example, I wanted to cd to the build directory for firefox and the shell figured out for me that it was a subdirectory of `/usr/ports/www`. I then wanted to cd to the build directory for gimp, and the shell figured out it was a subdirectory of `/usr/ports/graphics`.



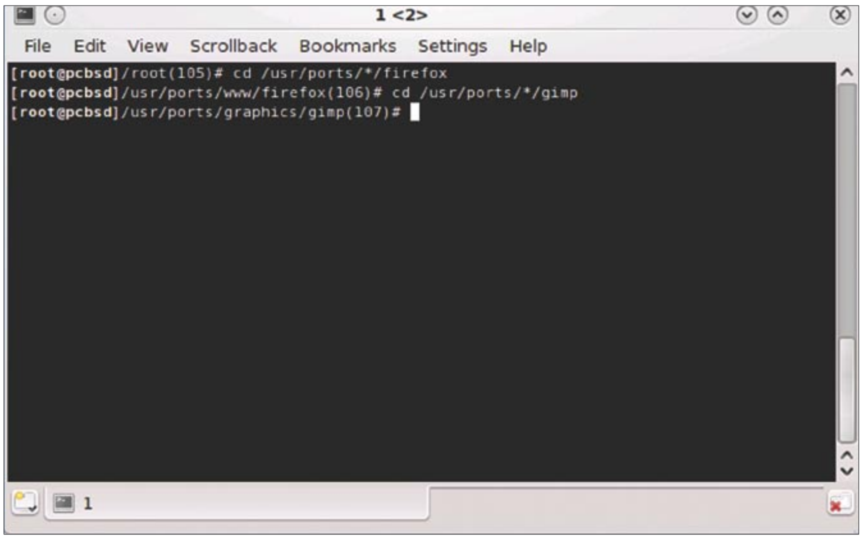
```
> w
> q
No label changes.
Mount point for wd0d (size=4097144k)? (or 'none' or 'done') [/tmp] <enter>
Mount point for wd0e (size=6291432k)? (or 'none' or 'done') [/var] <enter>
Mount point for wd0g (size=10485720k)? (or 'none' or 'done') [/usr] <enter>
Mount point for wd0h (size=10437400k)? (or 'none' or 'done') [/home] <enter>
Mount point for wd0d (size=4097144k)? (or 'none' or 'done') [/tmp] done
No more disks to initialize.
```

Figure 1. Default Disklabel Screen



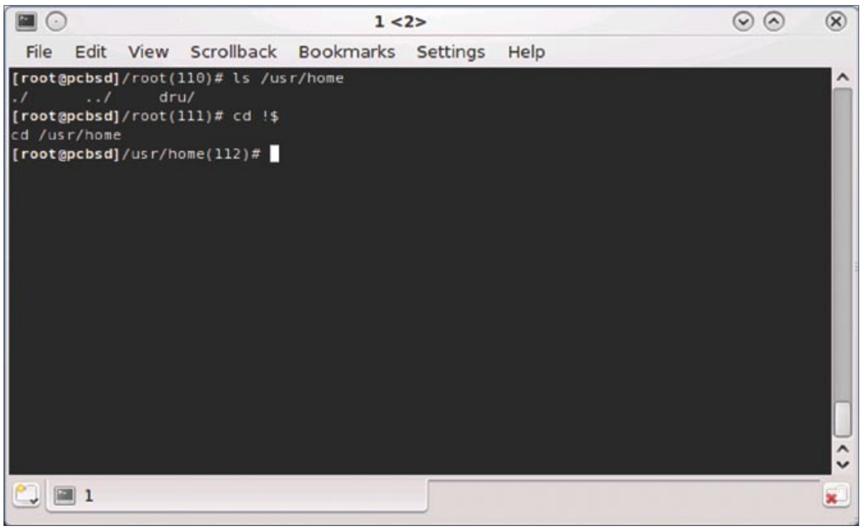
```
> w
> q
No label changes.
Mount point for wd0d (size=4097144k)? (or 'none' or 'done') [/tmp] <enter>
Mount point for wd0e (size=6291432k)? (or 'none' or 'done') [/var] <enter>
Mount point for wd0g (size=10485720k)? (or 'none' or 'done') [/usr] <enter>
Mount point for wd0h (size=10437400k)? (or 'none' or 'done') [/home] none
Mount point for wd0d (size=4097144k)? (or 'none' or 'done') [/tmp] done
No more disks to initialize.
```

Figure 2. Instructing disklabel to Ignore /home



```
[root@pcbsd]/root(105)# cd /usr/ports/*/firefox
[root@pcbsd]/usr/ports/www/firefox(106)# cd /usr/ports/*/gimp
[root@pcbsd]/usr/ports/graphics/gimp(107)#
```

Figure 3. cd to Unknown Directory



```
[root@pcbsd]/root(110)# ls /usr/home
./      ../     dru/
[root@pcbsd]/root(111)# cd !$
cd /usr/home
[root@pcbsd]/usr/home(112)#
```

Figure 4. Last Parameter Substitution

## Adding Notification of RAID Status to Daily Output

Charles Sprickman of NYCBUG has a shell script he added to /usr/local/etc/periodic/daily to put RAID status in his daily emails: see Listing 1.

## !$ Substitution

Francisco Reyes of NYCBUG reminds us how handy !$ can be. Figure 4 shows an example usage. In this example, the shell remembered that the value of the last parameter in the `ls /usr/home` command was `/usr/home`. In the second command, I asked to cd to that last value (represented by the variable `!$`), meaning that the shell interpreted this command as `cd /usr/home`.

## Miscellaneous FreeBSD Tips

George Rosamond of NYCBUG has several tips he uses on his FreeBSD systems. He typically adds the following lines to `/etc/rc.conf`: see Listing 2.

For servers that aren't running X11, add the following line to `/etc/make.conf` before installing any ports:

```
WITHOUT_X11=yes
```

If you haven't heard of src.conf, read "man src.conf" to see if any settings are useful to your environment.

If you'd like to be notified when a task or script is complete on a remote system, add `&& mail` to the command.

## NetBSD 5 in Parallels 4

Michael Hernandez of NYCBUG was able to get NetBSD 5.x to work in Parallels by configuring the *guest OS* section as *Solaris* after discovering that choosing *Other* or *FreeBSD* did not properly configure networking.

## dtrace

Pete Wright and Sahil Tandon of NYCBUG have some tips for those of you who have been wanting to give dtrace a try. The following scripts are available on OSX 10.5:

```
iosnoop
iotop
iopattern
iopending
opensnoop
```

`man -k dtrace` or `apropos dtrace` on OSX shows a bunch of other precooked scripts.

The DtraceToolkit, available in the sysutils section of the FreeBSD ports collection, provides the same functionality.

## CARP

Ike Levy and Okan Demirmen from NYCBUG have some suggestions for those of you using carp(4) for redundant routers or firewalls. On FreeBSD, the lagg(4) interface makes it extremely easy to setup link failover or link aggregation using ifconfig.

On OpenBSD, use the trunk(4) interface. Both interfaces support load balancing, the LACP protocol, and `EtherChannel`.

## Keeping Output Headers with Sed

Giorgos Keramidas (*http://keramida.wordpress.com/*) has a good tip can that be used to filter through the output

a d v e r t i s e m e n t

RootBSD

PREMIERE VPS HOSTING

Latest FreeBSD
Full Root Access
Starting at $20/mo
VPS and Dedicated
Multiple Datacenter Locations
Friendly, Knowledgeable Support Staff

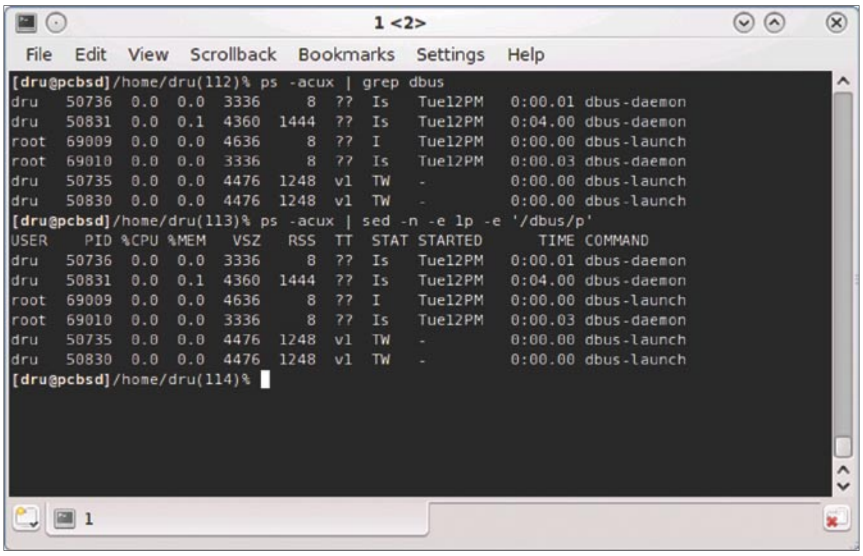
WWW.ROOTBSD.NET

Listing 1. Adding Notification of RAID Status to Daily Output

```
------------------------------------------------------------------------
#!/bin/sh

# show number of non-optimal drives attached to mpt raid card

NONOPT='/sbin/sysctl -n dev.mpt.0.nonoptimal_volumes'

echo
echo "Checking MPT RAID array"
echo

if [ $NONOPT -eq 0 ]; then
         echo "No non-optimal volumes: ($NONOPT)"
elif [ $NONOPT -ne 0 ]; then
         echo "WARNING, $NONOPT non-optimal volumes!"
fi
------------------------------------------------------------------------
```

Listing 2. Miscellaneous FreeBSD Tips

```
rc_debug="YES"
rc_info="YES"
From "man rc.conf":
rc_debug     (bool) If set to ''YES'', enable output of debug messages
                 from rc scripts.  This variable can be helpful in
diagnosing
                 mistakes when editing or integrating new scripts.  Beware
                 that this produces copious output to the terminal and
                 syslog(3).

rc_info      (bool) If set to ''NO'', disable informational messages from
                 the rc scripts.  Informational messages are displayed
when a
                 condition that is not serious enough to warrant a warning
or
                 an error occurs.
```



```
[dru@pcbsd]/home/dru(112)% ps -acux | grep dbus
dru    50736  0.0  0.0  3336     8  ??  Is   Tue12PM    0:00.01 dbus-daemon
dru    50831  0.0  0.1  4360  1444  ??  Is   Tue12PM    0:04.00 dbus-daemon
root   69009  0.0  0.0  4636     8  ??  I    Tue12PM    0:00.00 dbus-launch
root   69010  0.0  0.0  3336     8  ??  Is   Tue12PM    0:00.03 dbus-daemon
dru    50735  0.0  0.0  4476  1248  v1  TW   -          0:00.00 dbus-launch
dru    50830  0.0  0.0  4476  1248  v1  TW   -          0:00.00 dbus-launch
[dru@pcbsd]/home/dru(113)% ps -acux | sed -n -e 1p -e '/dbus/p'
USER     PID %CPU %MEM   VSZ   RSS  TT  STAT STARTED       TIME COMMAND
dru    50736  0.0  0.0  3336     8  ??  Is   Tue12PM    0:00.01 dbus-daemon
dru    50831  0.0  0.1  4360  1444  ??  Is   Tue12PM    0:04.00 dbus-daemon
root   69009  0.0  0.0  4636     8  ??  I    Tue12PM    0:00.00 dbus-launch
root   69010  0.0  0.0  3336     8  ??  Is   Tue12PM    0:00.03 dbus-daemon
dru    50735  0.0  0.0  4476  1248  v1  TW   -          0:00.00 dbus-launch
dru    50830  0.0  0.0  4476  1248  v1  TW   -          0:00.00 dbus-launch
[dru@pcbsd]/home/dru(114)%
```

Figure 5. grep vs. sed

of commands when matching specific patterns, without losing the header line in the output:

```
command | sed -n -e 1p -e '/PATTERN/p'
```

Figure 5 shows the difference between grepping the output of a command vs. using the sed pattern trick to filter the same output:

Notice that in the grep output, the beginning header line is stripped away (`USER`, `PID`, `%CPU`, etc.) making the results less meaningful than the sed output which includes the header information. This sed trick will work with other sorts of commands such as ps(1) or iostat(1) output, or any other command that outputs a header before numeric stats.

We hope that you have enjoyed the tips in this column. If you have any tricks of your own, send them to *dru@osbr.ca* to be included in a future edition of the column.

## About the Author

Dru Lavigne is a network and systems administrator, IT instructor, author and international speaker. She has over a decade of experience administering and teaching Netware, Microsoft, Cisco, Checkpoint, SCO, Solaris, Linux and BSD systems. She is author of the FreeBSD Basics column for O'Reilly, BSD Hacks, The Best of FreeBSD Basics, and the upcoming Definitive Guide to PC-BSD.

She is currently the Editor-in-Chief of the Open Source Business Resource, a free monthly publication covering open source and the commercialization of open source assets. She is founder and current Chair of the BSD Certification Group Inc., a non-profit organization with a mission to create the standard for certifying BSD system administrators. She recently joined the Board of the FreeBSD Foundation.

## How to Rename Ethernet Interfaces Under FreeBSD by Mikel King

I haven't written about things like this in a while but the question was put to me and I thought it'd be worth jotting something down.

Perhaps you prefer something like the generic `eth0` used on your Linux boxes or `en0` as commonly found on Mac OS X servers, or maybe something as short as `e0` typically found on Cisco and Adtran routers and switches. Then again maybe you just want to name them something specific like public, private or DMZ.

So first you are probably asking yourself why would you ever want to change the name of your `bge0` to something else? To answer it simply comes down to keeping things simple. Redundant no? Honestly if you have a set of standard ipfw firewall rules for instance that you wish to roll out to all of your machines however they all have NIC cards from different manufacturers then this will require quite a lot of work. Therefore why not just make it part of your initial setup to generic things up a bit?

Honestly, if you take a few minutes to prepare your machines ahead of time then you can use some sort of version control tools like svn to hold a single copy of your base firewall rules. Then you can perform a simple checkout and raise your shields in seconds. I quick change to the base checked back in and then if you had all machines on a trigger system they can checkout the current versions effectively remodulating the shield frequencies. Ok perhaps that was a bit too Star Trekky for most people.

So here's how to do it. On the command line as root or via sudo you can invoke ifconfig directly as follows;

```
ifconfig bge1 name e1
```

Here is the basic ifconfig output prior to executing the above command: see Listing 1.

And the same after executing the command: see Listing 2.

Notice that the only change was the name identifying the second ethernet interface. Of course being able to manually manipulate the ethernet interface names is all well and good. I suppose you could also write your own script and stuff it into the rc.network startup somewhere but that'd be a total waste of effort when tyou can just use the built in rc.conf as follows to make the same change occur at startup.

You would make a change similar to the following in `/etc/rc.conf`

```
ifconfig_bge0_name="e0"
ifconfig_e0="inet 10.10.10.13 netmask
255.255.255.0"
```

After a reboot you would see the following ifconfig output: see Listing 3.

Observe that the interface formerly known as bge0 is now simply `e0`. I shall leave that up to you imagination as to why the name of e1 has reverted back to `bge1`.

Honestly FreeBSD allows you the power to name the interfaces whatever you like. Maybe, just maybe you are one of those individuals that like to name things after your favorite flavor of ice cream, or after your favorite characters or Dune. Now that you know how the choice is entirely up to you. Go have fun with it! I hope that this little technical note has been helpful.

**Listing 1.** Original named interface ifconfig output

```
bge0: flags=8843 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c3
inet 10.10.10.13 netmask 0xffffff00 broadcast
10.10.10.255
media: Ethernet autoselect (100baseTX )
status: active
bge1: flags=8802 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c2
media: Ethernet autoselect (none)
status: no carrier
lo0: flags=8049 metric 0 mtu 16384
inet 127.0.0.1 netmask 0xff000000
```

**Listing 2.** Renamed interface ifconfig output

```
bge0: flags=8843 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c3
inet 10.10.10.13 netmask 0xffffff00 broadcast
10.10.10.255
media: Ethernet autoselect (100baseTX )
status: active
e1: flags=8802 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c2
media: Ethernet autoselect (none)
status: no carrier
lo0: flags=8049 metric 0 mtu 16384
inet 127.0.0.1 netmask 0xff000000
```

**Listing 3.** Renamed interface ifconfig output after rebooting

```
e0: flags=8843 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c3
inet 204.107.76.13 netmask 0xffffff00 broadcast
10.10.10.255
media: Ethernet autoselect (100baseTX )
status: active
bge1: flags=8802 metric 0 mtu 1500
options=9b
ether 00:0b:cd:f2:d8:c2
media: Ethernet autoselect (none)
status: no carrier
lo0: flags=8049 metric 0 mtu 16384
inet 127.0.0.1 netmask 0xff000000
```

### About the Author

Mikel King (http://twitter.com/mikelking) has been working in the Information Services field for over 20 years. He is currently the CEO of Olivent Technologies, a professional creative services partnership in NY. Additionally he is currently serving as the Secretary of the BSD Certification group as well as a Senior Editor for Daemon News. Recently Mikel was selected as the NYC Technology Writer for the Examiner.

# Year 40 of the UNIX epoch begins

Brian D'Arcangelo, MCSE, Lynn Community Health Center, Lynn, MA, USA

As many UNIX/Linux users know, all UNIX like operating systems start the count of time at January 1, 1970, the start of the UNIX epoch. Yes, I know that this is not precisely when the UNIX operating system was born but for our purposes it will do. It is similar to the idea that January 1, 2010 A.D. does not really represent the precise time since the birth of Christ (astronomers have proven this to be off by a few years) but we still use it as a time marker.

We do know that it was sometime in 1970 that the operating system got its name. Since it was derived from the abandoned MULTICS project at AT&T/Bell Labs it is said that Ken Thompson (one of its creators) chose the name *Unics* as a pun on the name *Multics*. Since the name *Unics* phonetically sounds like it ends in an *x* the name UNIX emerged and stuck.

The summary I am interested in giving is not so much about the chronological history of UNIX over the past 40 years but, instead, the profound impact that this landmark operating system has had on all of the operating systems that would follow. In my estimation, nearly every modern computerized technology that we use today can be in some way traced back to UNIX. We today can feel the same way towards the creators of UNIX as Winston Churchill felt about the pilots in the RAF during WWII when he said, *Never has so much been owed by so many to so few.*

Start with the technology that no one could imagine being without today – the Internet. Even many non-technical persons are aware that the Internet is completely dependent on the TCP/IP networking protocol. But where did TCP/IP come from? It was first developed on the UNIX operating system. For that matter, when the Internet was in its infancy and was know as the DARPAnet, the entire backbone for it was built almost exclusively on UNIX.

We also owe our ability to use the Internet in a human friendly way in large part to UNIX. Whenever we enter an easy to remember URL into a web browser (such as *http://www.bsdmag.org*) DNS does the dirty work of translating that name into a network address and finding it. DNS was of course first developed and run on UNIX when it was known as BIND (Berkeley Internet Name Domain system) and became part of the BSD version of UNIX.

While we are on the subject of DNS, it is noteworthy that some of the most important servers that are really the backbone of the entire Domain Naming system on the Internet are in fact running on a UNIX operating system.

Indeed, to this day, not just DNS, but many of the bread-and-butter services we use on the Internet such as search engines (i.e. Google,) email, web servers, and so on, continue to be run on some flavor of UNIX.

A little known piece of operating system history is the contribution that UNIX made to the growth of Microsoft in its early days. If you worked at Microsoft in the early 1980's you would have been quite familiar with UNIX. Microsoft's entire corporate network infrastructure was built on the Xenix flavor of UNIX and it remained so for quite some time. At that time, Microsoft believed that UNIX (i.e. Xenix) would emerge as its flagship product. Later, when Microsoft switched in directing its attention to MS-DOS, the legacy of UNIX persisted when artifacts from UNIX such as piping and redirection were incorporated into DOS.

UNIX is not only an important part of Microsoft's legacy, but Apple Computers as well. You would be surprised to know how many Mac OS X users are not even aware that the operating system they are using is really UNIX underneath the hood. The legacy of UNIX even persists in such devices as Apples iPod. You will see forums on the Internet littered with questions from stumped owners wondering why their iPod has mysteriously drifted to the date 1st January 1970.

While 1st January 1970 may be a mysterious date to the uninitiated, it was and remains the date from which time began as far as the UNIX community is concerned. May all of us today remember how much so many of us owe to the so few geniuses that gave us the still living legacy of the UNIX operating system.

# SAINT®

## Integrated Vulnerability Assessment and Penetration Testing

### Examine, expose, and exploit your vulnerabilities before an attacker does

Examine your network with the SAINT® vulnerability scanner, and expose the areas where an attacker could breach your network. Then, take the next step and exploit the vulnerability. This allows you to focus on the high-severity vulnerabilities and provides a starting point for prioritizing remediation efforts.

**What SAINT can do for you –**

- ✔ Lets you exploit vulnerabilities found by the vulnerability scanner
- ✔ Show you if your network is compliant with PCI security standards. SAINT is a PCI DSS approved scanning vendor.
- ✔ Show you where to begin remediation—with the exploitable vulnerabilities
- ✔ Let you scan and exploit both IPv4 and IPv6 addresses
- ✔ Give you cross references to CVE, CVSS, IAVA, OSVDB, and SANS/FBI Top 20
- ✔ Allows you to generate colorful and easy-to-read reports

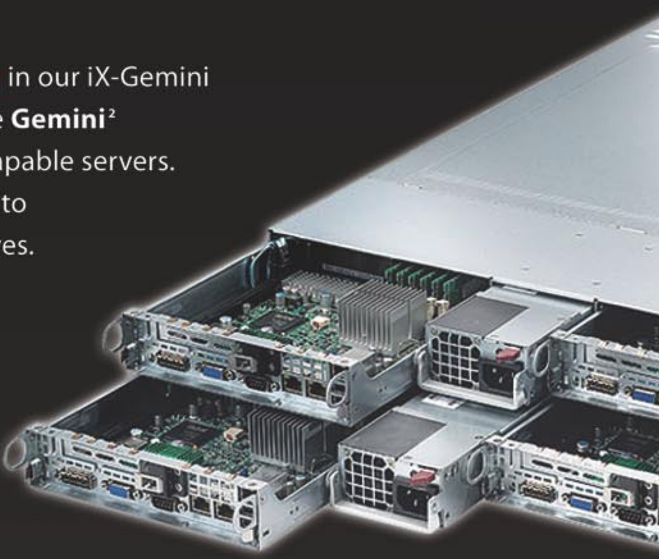
Get free white papers and more information at www.saintcorporation.com

Contact SAINT's sales team at 1-800-596-2006 x0119 or sales@saintcorporation.com

Extreme Network Security

# Gemini²: The Fantastic Four

**IXsystems is proud to introduce the latest offering** in our iX-Gemini line, the **Gemini²**. Cleverly disguised as any other 2U server, the **Gemini²** secretly houses 4 highly efficient, extremely powerful RAID 5 capable servers. Each node supports the latest Intel® 5500 series processors, up to 48GB of DDR3 memory, and three 3.5" hot-swappable hard drives.



This system architecture achieves breakthrough x86 server performance-per-watt (375 GFLOPS/kW) to further satisfy the ever-increasing demands for efficiency, density and low-TCO of today's high performance computing (HPC) clusters and data centers. For more information and pricing, please visit our website at http://www.iXsystems.com/gemini2.

## Features

### Four hot-pluggable systems (nodes) in a 2U form factor

**Each node supports the following:**

- Dual 64-Bit Socket 1366 Quad-Core or Dual-Core Intel® Xeon® Processor 5500 Series
- 3 x 3.5" SAS/SATA Hot-swappable Drive Bays
- Intel® 5520 Chipset with QuickPath Interconnect (QPI)
- Up to 48GB DDR3 1333/1066/800 SDRAM ECC Registered Memory
- 1 (x16) PCI-E (Low Profile)
- Matrox G200eW 8 MB DDR2 Memory Video
- Integrated Remote Management - IPMI 2.0 + IP-KVM with dedicated LAN
- All four nodes share a Redundant 1200W High-efficiency Power Supply (Gold Level 92%+ power efficiency)

iX systems™

**800-820-BSDI**
http://www.iXsystems.com
Enterprise Servers for Open Source

intel Xeon inside™
Powerful. Intelligent.