

## Keeping FreeBSD Applications Up-To-Date

Richard Bejtlich (taosecurity at gmail dot com)

[www.taosecurity.com/keeping\\_freebsd\\_applications\\_up-to-date.html](http://www.taosecurity.com/keeping_freebsd_applications_up-to-date.html)

[www.taosecurity.com/keeping\\_freebsd\\_applications\\_up-to-date.pdf](http://www.taosecurity.com/keeping_freebsd_applications_up-to-date.pdf)

[www.taosecurity.com/keeping\\_freebsd\\_applications\\_up-to-date.ps](http://www.taosecurity.com/keeping_freebsd_applications_up-to-date.ps)

21 April 2005

---

### Introduction

This article is a sequel to my previous work "Keeping FreeBSD Up-To-Date," available here:

[www.taosecurity.com/keeping\\_freebsd\\_up-to-date.html](http://www.taosecurity.com/keeping_freebsd_up-to-date.html)

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 FreeBSD applications up-to-date. I explain how to install and upgrade several applications on a FreeBSD 5.2.1 RELEASE system. In my previous article "Keeping FreeBSD Up-To-Date," I described how to patch and upgrade the FreeBSD operating system, beginning with FreeBSD 5.2.1 and ending with FreeBSD 5-STABLE. Taken as a pair, these two articles will help system administrators keep their FreeBSD OS and applications current and defensible.

I chose FreeBSD 5.2.1, released in February 2004, as my reference platform because the applications bundled with it have been updated several times in the past ten months. These updates provide good material for the case-based approach used in this article. All of the techniques explained here work on the most recent FreeBSD version, 5.3.

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. [1] 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.

My advice on upgrading these applications 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 applications up-to-date or maintaining secure software. I strongly recommend reading the excellent FreeBSD Handbook as well as the multiple helpful published books on FreeBSD.

Third party applications may be installed using source code, the FreeBSD ports tree, or precompiled packages. Each will be described in

detail, but not exhaustively. The tips here are enough to get the novice to intermediate system administrator managing applications on FreeBSD.

If you are reading this as a hard copy, the right hand side of the page may be cut off. This is important when long URLs or command line statements are involved. Please refer to the HTML version of this document for the authoritative version.

---

## Installation Using Source Code

Source code is typically packaged as an archive processed by the `tar(1)` and `gzip(1)` programs, with the `.tar.gz` or `.tgz` suffixes. In the following example, we install the Snort output reader Barnyard from source code. Until further notified, we take these actions as a user and not as root.

As a good system administration practice, one might want to create a specific directory to hold source code for third party applications. Creating `/usr/local/src` is the method used here.

```
freebsd521$ mkdir /usr/local/src
```

Next we download the Barnyard tarball with `fetch(1)`, after visiting the Barnyard project page and locating a suitable distribution site. [2]

```
freebsd521$ fetch http://kent.dl.sourceforge.net/sourceforge/barnyard/barnyard-0.2.0.tar.gz
Receiving barnyard-0.2.0.tar.gz (161543 bytes): 100%
161543 bytes transferred in 6.6 seconds (23.78 kBps)
```

Now extract the contents of the archive:

```
freebsd521$ tar -xzvf barnyard-0.2.0.tar.gz
barnyard-0.2.0/
barnyard-0.2.0/docs/
barnyard-0.2.0/docs/BUGS
...edited...
barnyard-0.2.0/src/input-plugins/dp_stream_stat.h
barnyard-0.2.0/src/input-plugins/dp_stream_stat.c
```

Once the source code is extracted, we change into the new directory and look for a configure script:

```
freebsd521$ cd barnyard-0.2.0
freebsd521$ ls
AUTHORS          README           config.guess     docs             src
COPYING          acconfig.h      config.h.in      etc             stamp-h.in
LICENSE.QPL      aclocal.m4      config.sub       install-sh
Makefile.am      autoclean.sh    configure        missing
Makefile.in      autojunk.sh     configure.in     mkinstalldirs
```

We execute the configure script and pass an optional parameter to enable MySQL output.

```
freebsd521$ ./configure --enable-mysql
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... yes
checking for working aclocal-1.4... missing
...edited...
checking for strerror... yes
checking for /mysql.h... no

*****
ERROR: unable to find mysql headers (mysql.h)
checked in the following places
    /mysql.h
*****
```

The configure script abruptly stops because it cannot find `mysql.h`. This demonstrates the major weakness of installing software from source code. Administrators must be aware of any dependencies required by the application, and address them prior to installing the new software.

Assume we take care of the dependency (using a method to be demonstrated shortly). We re-run the configure script:

```
freebsd521$ ./configure --enable-mysql
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
...edited...
checking for strerror... yes
checking for /usr/local/include/mysql/mysql.h... yes
checking for mysql_real_connect in -lmysqlclient... no

*****
ERROR: unable to find mysqlclient library
checked in the following places
    /usr/local/lib/mysql
*****
```

Now we see a new error. Although the `mysql.h` file was found, the configure script is not finding the MySQL client library where it expects to find it. A look in `/usr/local/lib/mysql` shows the following:

```
freebsd521$ ls /usr/local/lib/mysql/
libmysqlclient.a      libmysqlclient.so      libmysqlclient.so.10
```

It appears the files needed are in place. This error demonstrates the second weakness of installing software from source. Sometimes it does not work as expected, and administrators must tweak installation files to accommodate their systems.

The resolution to this problem surfaces by doing a Google search. If we modify the configure script as shown, it fixes the problem:

original configure script:

```
LIBS="$LIBS -lz -lssl -lmysqlclient"
```

modified configure script:

```
LIBS="$LIBS -lz -lssl -lcrypto -lmysqlclient"
```

Rerun the configure script:

```
freebsd521$ ./configure --enable-mysql
creating cache ./config.cache
checking for a BSD compatible install... /usr/bin/install -c
...edited...
checking for strerror... (cached) yes
checking for /usr/local/include/mysql/mysql.h... yes
checking for mysql_real_connect in -lmysqlclient... yes
...edited...
creating config.h
config.h is unchanged
```

Because the configure script completed, we can now execute make(1) and begin compiling Barnyard:

```
freebsd521$ make
make all-recursive
Making all in src
Making all in output-plugins
gcc -DHAVE_CONFIG_H -I. -I. -I../.. -I../.. -I../..../src -I/usr/local/include/mysql
-DENABLE_MYSQL -g -O2 -Wall -c op_decode.c
gcc -DHAVE_CONFIG_H -I. -I. -I../.. -I../.. -I../..../src -I/usr/local/include/mysql
-DENABLE_MYSQL -g -O2 -Wall -c op_fast.c
...edited...
gcc -DHAVE_CONFIG_H -I. -I. -I.. -I.. -I/usr/local/include/mysql -DENABLE_MYSQL
-g -O2 -Wall -c barnyard.c
gcc -DHAVE_CONFIG_H -I. -I. -I.. -I.. -I/usr/local/include/mysql -DENABLE_MYSQL
-g -O2 -Wall -c mstring.c
...edited...
ProgVars.c:672: warning: long unsigned int format, time_t arg (arg 3)
```

```
gcc -g -O2 -Wall -L/usr/local/lib/mysql -o barnyard barnyard.o mstring.o
  strlcatu.o strlcpyu.o util.o spool.o sid.o debug.o classification.o
  CommandLineArgs.o ConfigFile.o ProgVars.o output-plugins/libop.a
  input-plugins/libdp.a -lz -lssl -lcrypto -lmysqlclient
```

At this point we assume root privileges and execute 'make install' to copy the Barnyard executable to /usr/local/bin:

```
freebsd521# make install
Making install in src
Making install in output-plugins
Making install in input-plugins
/bin/sh ../mkinstalldirs /usr/local/bin
  /usr/bin/install -c barnyard /usr/local/bin/barnyard
```

On FreeBSD systems where the root shell is tcsh or csh, administrators must run rehash(1) to recompute the hash table for the PATH variable. If one does not do this, the system will not find barnyard(1). For example:

```
freebsd521# barnyard -h
barnyard: Command not found.
freebsd521# ls /usr/local/bin/barnyard
/usr/local/bin/barnyard
freebsd521# rehash
freebsd521# barnyard -h
Barnyard Version 0.2.0 (Build 32)
Usage: barnyard [OPTIONS]...          (continual mode)
      or: barnyard -o [OPTIONS]... FILES... (batch mode)
...truncated...
```

Barnyard is now installed.

Installation from source code has four main weaknesses:

- Administrators must resolve dependencies manually.
- Administrators may have to tweak configuration scripts to accommodate their systems.
- Administrators may have to take additional actions to install code without configure scripts or Makefiles.
- Software installed from source can not usually be managed by the native FreeBSD package management tools.

Installation from source code has two main advantages:

- Developers publish their code in source archive form. Therefore, it is typically the freshest version available.
- Administrators have maximum flexibility when working with source code.

My personal preference is to avoid installing source code in this manner if at all possible. I prefer one of the two methods that follow.

## Installation Using the FreeBSD Ports Tree

FreeBSD offers an extremely powerful means of installing software. This system is known as the ports tree. Administrators have the option of adding this structure to their system during installation. Alternatively, a recent copy of the source tree archive can be downloaded from [www.freebsd.org/ports/](http://www.freebsd.org/ports/) and extracted to the /usr/ports directory.

The FreeBSD ports tree is a system to facilitate the installation of third party applications. The FreeBSD Handbook and other references already comprehensively explain this resource, so I will restrict this text to demonstrating the installation of a tool, FreeBSD Update, using the ports tree. In the previous article we used FreeBSD Update to keep the OS up-to-date. As such it may be considered a "security tool."

A quick look at /usr/ports shows multiple files and directories, corresponding to categories of tools:

```
freebsd521# cd /usr/ports
freebsd521# ls
.cvsignore      audio           finance         math            shells
INDEX           benchmarks     french          mbone          sysutils
INDEX-5         biology        ftp             misc           textproc
LEGAL           cad            games          multimedia     ukrainian
MOVED           chinese        german         net            vietnamese
Makefile        comms          graphics       news           www
Mk              converters     hebrew         palm           x11
README          databases     hungarian     picobsd        x11-clocks
README.html     deskutils     irc            polish         x11-fm
Templates       devel         japanese       portuguese     x11-fonts
Tools           distfiles     java           print          x11-servers
arabic          dns            korean        russian        x11-toolkits
archivers       editors       lang           science        x11-wm
astro           emulators     mail           security
```

A quick look inside the security directory shows tools which provide security functions:

```
freebsd521# ls security | head
ADM smb
ADM snmp
IMHear
Makefile
README.html
aafid2
acid
aes crypt
aide
altivore
```

In fact, we see 'freebsd-update' listed, if we use `grep(1)` to narrow down the directory listing:

```
freebsd521# ls security | grep -i update
freebsd-update
```

If we were not able to guess at the location of our tool of interest, we could turn to other resources. Two Web-based options exist. Search functions at [www.freshports.org](http://www.freshports.org) or [www.freebsd.org/ports](http://www.freebsd.org/ports) reveal that the FreeBSD Update program is found in the security/freebsd-update portion of the tree.

The ports tree itself offers two other methods to find tools of interest. From the /usr/ports directory, use `make(1)` to find ports with the "update" keyword:

```
freebsd521# make search key=update | more
...edited...
Port:    freebsd-update-1.4
Path:    /usr/ports/security/freebsd-update
Info:    Fetches and installs binary updates to FreeBSD
Maint:   cperciva@daemonology.net
Index:   security
B-deps:
R-deps:  bsdiff-4.1
...truncated...
```

If we were sure of the tool's name (i.e., "freebsd-update"), we could leverage that knowledge directly to check if the tool is in the ports tree:

```
freebsd521# make search name=freebsd-update
Port:    freebsd-update-1.4
Path:    /usr/ports/security/freebsd-update
Info:    Fetches and installs binary updates to FreeBSD
Maint:   cperciva@daemonology.net
Index:   security
B-deps:
R-deps:  bsdiff-4.1
```

At this point we've used several means to locate `freebsd-update(1)`. Now we install it as user root by running `make(1)` inside the /usr/ports/security/freebsd-update directory. Assume you are taking this action just after FreeBSD 5.2.1 was released, so the `freebsd-update(1)` version shown (1.4) is the newest available:

```
freebsd521# cd security/freebsd-update/
freebsd521# make
>> freebsd-update-1.4.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from http://www.daemonology.net/freebsd-update/.
Receiving freebsd-update-1.4.tar.gz (29567 bytes): 100%
```

```
29567 bytes transferred in 0.1 seconds (236.29 kBps)
===> Extracting for freebsd-update-1.4
>> Checksum OK for freebsd-update-1.4.tar.gz.
/usr/bin/sed -e "s#PREFIX=/usr/local#PREFIX=/usr/local#g"
  /usr/ports/security/freebsd-update/work/freebsd-update-1.4/freebsd-update >
  /usr/ports/security/freebsd-update/work/freebsd-update-1.4/freebsd-update.new
  /bin/mv /usr/ports/security/freebsd-update/work/freebsd-update-1.4/freebsd-update.new
  /usr/ports/security/freebsd-update/work/freebsd-update-1.4/freebsd-update
===> Patching for freebsd-update-1.4
===> Configuring for freebsd-update-1.4
===> Building for freebsd-update-1.4
cc -O -pipe -mcpu=pentiumpro -mcpu=pentiumpro -O3 -I lib -o freebsd-update-verify verify.c
  lib/hashtab.c lib/hash.c lib/fftlut.c lib/fft.c lib/smpa.c lib/numt.c lib/rsa.c
```

We see the freebsd-update-1.4.tar.gz source code archive is retrieved from [www.daemonology.net/freebsd-update](http://www.daemonology.net/freebsd-update). The archive is extracted, patched, configured, and compiled. Next we run 'make install' to install the program:

```
freebsd521# make install
===> Installing for freebsd-update-1.4
===> freebsd-update-1.4 depends on executable: bspatch - not found
===> Verifying install for bspatch in /usr/ports/misc/bsdiff
>> bsdiff-4.1.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
>> Attempting to fetch from http://www.daemonology.net/bsdiff/.
Receiving bsdiff-4.1.tar.gz (7721 bytes): 100%
7721 bytes transferred in 0.0 seconds (156.32 kBps)
===> Extracting for bsdiff-4.1
>> Checksum OK for bsdiff-4.1.tar.gz.
===> Patching for bsdiff-4.1
===> Configuring for bsdiff-4.1
===> Building for bsdiff-4.1
cc -O -pipe -mcpu=pentiumpro -mcpu=pentiumpro -O3 bsdiff.c -o bsdiff
cc -O -pipe -mcpu=pentiumpro -mcpu=pentiumpro -O3 bspatch.c -o bspatch
===> Installing for bsdiff-4.1
===> Generating temporary packing list
===> Checking if misc/bsdiff already installed
install -c -s -m 555 bsdiff bspatch /usr/local/bin
install -c -m 444 bsdiff.1 bspatch.1 /usr/local/man/man1
===> Compressing manual pages for bsdiff-4.1
===> Registering installation for bsdiff-4.1
===> Returning to build of freebsd-update-1.4
===> Generating temporary packing list
===> Checking if security/freebsd-update already installed
install -s -o root -g wheel -m 555 freebsd-update-verify /usr/local/sbin
```

```
install -o root -g wheel -m 555 freebsd-update /usr/local/sbin
install -o root -g wheel -m 444 freebsd-update.conf.5 /usr/local/man/man5
install -o root -g wheel -m 444 freebsd-update.8 /usr/local/man/man8
mkdir /usr/local/freebsd-update
install -o root -g wheel -m 444 freebsd-update.conf /usr/local/etc/freebsd-update.conf.sample
mkdir /usr/local/share/doc/freebsd-update
install -o root -g wheel -m 444 LICENSE README VERSION /usr/local/share/doc/freebsd-update
...edited...
==> Compressing manual pages for freebsd-update-1.4
==> Registering installation for freebsd-update-1.4
```

During this process, something interesting happened. The installation process recognized that `freebsd-update(1)` had an unresolved dependency. `Freebsd-update(1)` requires the `bsdiff(1)` program, but `bsdiff(1)` was not installed. Thanks to the power of the ports tree, the installation process first installed `bsdiff(1)` and then continued with the installation of `freebsd-update(1)`. If `bsdiff(1)` had any unresolved dependencies, the installation process would have taken care of those before declaring `bsdiff(1)` ready. This incredible ports tree feature makes installing software much simpler than compiling source code manually.

When done, the new applications are installed as packages. This is a crucial point to understand. Any application installed through the ports tree, or as a precompiled package (shown next), ends up as a **package** on the system. For example, the native `pkg_info(1)` tool reveals the packages installed on our test system:

```
freebsd521# pkg_info
bash-2.05b.007      The GNU Bourne Again Shell
bsdiff-4.1         Generates and applies patches to binary files
freebsd-update-1.4 Fetches and installs binary updates to FreeBSD
mysql-client-3.23.58 Multithreaded SQL database (client)
perl-5.6.1_15     Practical Extraction and Report Language
```

Installation using the ports tree has three main weaknesses:

- If a tool you want (like `Barnyard`) is not in the ports tree, you can't install it using the ports tree.
- The versions of applications in the ports tree may lag their source code counterparts.
- The ports tree must be kept up-to-date in order to install the latest applications. (This process is explained later.)
- Although the ports tree resolves dependencies, the dependencies are specified by the ports maintainer. For example, a port may say it needs `Tcl 8.3`, but you may want it to use `Tcl 8.4`. Usually these problems can be resolved manually, but at the cost of "breaking" the tree with your own modifications.

Installation using the ports tree has five main strengths:

- Dependencies are resolved efficiently and with little or no manual intervention.
- Little or no manual tweaking is required, unlike source code installations. The patching process incorporates tweaks done by the port maintainers.
- An application found in the ports tree has some level of guarantee of working on FreeBSD, thanks to the testing and work of the port

- maintainers.
- Browsing the ports tree is a great way to find tools to accomplish many tasks.
- Applications deployed using the ports tree can be maintained using native package tools or other third-party applications.

When possible, I install applications using the ports tree. I find this aspect of FreeBSD to be one of its most compelling features.

---

## Installation Using Precompiled Packages

While installing Barnyard from source code, we came across a dependency for the MySQL client. This section shows how I quickly resolved this dependency by installing the MySQL client as a precompiled package.

Each FreeBSD release (at least for i386) is shipped with a set of packages. These are found on CD-ROMs shipped by vendors like FreeBSDmall.com or on the FreeBSD FTP mirrors. For example, the following FTP directory on FTP mirror nine contains packages for the i386 platform:

```
ftp://ftp9.freebsd.org/pub/FreeBSD/ports/i386/packages-5.2.1-release
```

Within that directory, the 'All' directory shows the actual packages:

```
ftp> cd All
250 "/pub/FreeBSD/ports/i386/packages-5.2-release/All" is new cwd.
ftp> ls freebsd-update*
227 Entering Passive Mode (128,10,252,13,223,67)
150 Data connection accepted from 69.243.15.208:49163; transfer starting.
-rw-r--r--  1 ftpuser  ftpusers      28927 Dec  9  2003 freebsd-update-1.4.tbz
226 Listing completed.
```

The 'Latest' directory is a collection of symbolic links using the base name for each package. Here we see that Latest/freebsd-update.tbz is a link to the real package, which bears a version number of 1.4. This is the same version we installed using the ports tree.

```
ftp> cd ..
250 "/pub/FreeBSD/ports/i386/packages-5.2-release" is new cwd.
ftp> cd Latest
250 "/pub/FreeBSD/ports/i386/packages-5.2-release/Latest" is new cwd.
ftp> ls freebsd-update*
227 Entering Passive Mode (128,10,252,13,223,69)
150 Data connection accepted from 69.243.15.208:49164; transfer starting.
lrwxrwxrwx  1 ftpuser  ftpusers      29 Aug 26 20:58 freebsd-update.tbz -> ../All/freebsd-upc
226 Listing completed.
```

Returning to our problem of resolving Barnyard's dependency on the MySQL client, we can install the program using the `pkg_add(1)` command. The `-v` switch enables verbose output and the `-r` switch denotes fetching the package from a remote location. If we had already

downloaded the package locally, -r is not needed. I prefer to install all packages using -r, because the installation process will also download any dependencies:

```
freebsd521# pkg_add -vr mysql-client
looking up ftp.freebsd.org
connecting to ftp.freebsd.org:21
setting passive mode
opening data connection
initiating transfer
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5.2.1-release/Latest/mysql-client
+COMMENT
+DESC
+MTREE_DIRS
man/man1/isamchk.1.gz
man/man1/isamlog.1.gz
man/man1/mysql.1.gz
...edited...
lib/mysql/libmysqlclient.so.10
tar command returns 0 status
Done.
Package 'mysql-client-3.23.58' depends on 'perl-5.6.1_15' with 'lang/perl5' origin.
- already installed.
extract: Package name is mysql-client-3.23.58
extract: CWD to /usr/local
extract: /usr/local/man/man1/isamchk.1.gz
extract: /usr/local/man/man1/isamlog.1.gz
...edited...
extract: /usr/local/lib/mysql/libmysqlclient.so.10
extract: execute '/sbin/ldconfig -m /usr/local/lib/mysql'
extract: CWD to .
Running mtree for mysql-client-3.23.58..
mtree -U -f +MTREE_DIRS -d -e -p /usr/local >/dev/null
Attempting to record package into /var/db/pkg/mysql-client-3.23.58..
Trying to record dependency on package 'perl-5.6.1_15' with 'lang/perl5' origin.
Package mysql-client-3.23.58 registered in /var/db/pkg/mysql-client-3.23.58
```

First the pkg\_add(1) process retrieves the mysql-client.tbz package from the "All" directory. This package name is really a symlink to the mysql-client-3.23.58.tbz package.

The package installation process also resolves dependencies automatically. The mysql-client-3.23.58 package depends on perl-5.6.1\_15. Since Perl is already installed, the package deployment continues. When done, we have our MySQL client and Barnyard can be installed from source code.

Installation using precompiled packages has five main weaknesses:

- Because the software is precompiled, it uses a "least common denominator" approach. Some customizations used by your system will not be applied to the software installation process.
- Packages tend to lag the ports tree and source code by several weeks.
- Some applications are not available as packages. Programs like OpenOffice.org take such a long time to compile that they put an unwanted strain on the FreeBSD package generation cluster. Others like the Java JDK can not be redistributed in package form. For some architectures, packages may not be available.
- Package dependencies can be more onerous than port dependencies. In other words, packages tend to be stricter about the dependencies they require. While you can tweak the dependencies needed by the ports tree manually, there is no similar capability when using precompiled packages.
- When you install a package, you trust the package source to not provide trojaned code.

Installation using precompiled packages has four main strengths:

- Package installation is much faster than compiling from source, using normal archives or the ports tree.
- Code that may not compile in the ports tree due to problems with your system or the tree may be available as a package.
- Slow systems that could spend hours or days compiling software can be maintained fairly inexpensively by using precompiled packages.
- Like tools installed with the ports tree, precompiled packages can be administered with native or third-party package management tools.

For initial system setup I tend to install what I need using precompiled packages. Once the system is running, I tend to use the ports tree for most situations.

---

## Updating Applications Installed from Source Code

The most direct way to update an application installed from source code is to uninstall the old version and install the new version. First, execute 'make uninstall' in the directory from which you executed 'make install'. For example:

```
freebsd521# cd /usr/local/src/barnyard-0.2.0
freebsd521# make uninstall
Making uninstall in src
Making uninstall in output-plugins
Making uninstall in input-plugins
list='barnyard'; for p in $list; do rm -f /usr/local/bin/'echo $p|sed 's/$//'|sed 's,x,x,'|sed
freebsd521# ls /usr/local/bin/barnyard
ls: /usr/local/bin/barnyard: No such file or directory
```

Now, download the new version of Barnyard, extract it, and follow the installation instructions already posted.

---

## Updating Packages by Deletion and Addition

Now we turn to keeping packages up-to-date. There is no section on keeping "ports" up-to-date or "packages" up-to-date. To reiterate, applications installed using the ports tree or precompiled packages all end up as **packages** on the system. While we will see tools with "port" or "pkg" in their names, all tend to act on packages installed on FreeBSD.

The most direct way to "update" a package is to remove it and install a new version. Early we used `pkg_info` to show installed packages. A check in `/var/db/pkg` also shows the packages installed.

```
freebsd521# cd /var/db/pkg
freebsd521# ls
bash-2.05b.007          freebsd-update-1.4      perl-5.6.1_15
bsdiff-4.1             mysql-client-3.23.58
```

I prefer to use `pkg_delete(1)` from this directory, because I can use tab-completion to specify the entire package name:

```
freebsd521# pkg_delete -v mysql-client-3.23.58
Change working directory to /usr/local
Delete file /usr/local/man/man1/isamchk.1.gz
Delete file /usr/local/man/man1/isamlog.1.gz
...edited...
Delete file /usr/local/man/man1/safe_mysqlld.1.gz
Execute 'rm -f /usr/local/man/cat1/isamchk.1 /usr/local/man/cat1/isamchk.1.gz'
Execute 'rm -f /usr/local/man/cat1/isamlog.1 /usr/local/man/cat1/isamlog.1.gz'
...edited...
Execute 'rm -f /usr/local/man/cat1/safe_mysqlld.1 /usr/local/man/cat1/safe_mysqlld.1.gz'
Delete file /usr/local/bin/mysql
Delete file /usr/local/bin/mysqladmin
...edited...
Delete file /usr/local/lib/mysql/libmysqlclient.so.10
Delete directory /usr/local/include/mysql
Delete directory /usr/local/lib/mysql
Execute 'if [ -f /usr/local/info/dir ]; then if sed -e '1,/Menu:/d'
  /usr/local/info/dir | grep -q '^[*] ' ; then true; else rm /usr/local/info/dir; fi; fi'
Execute '/sbin/ldconfig -R'
Change working directory to .
Trying to remove dependency on package 'perl-5.6.1_15' with 'lang/perl5' origin.
```

The package is gone, according to `pkg_info(1)` and a listing of `/var/db/pkg`:

```
freebsd521# pkg_info
bash-2.05b.007      The GNU Bourne Again Shell
bsdiff-4.1         Generates and applies patches to binary files
freebsd-update-1.4 Fetches and installs binary updates to FreeBSD
perl-5.6.1_15      Practical Extraction and Report Language
```

```
freebsd521# ls
bash-2.05b.007      freebsd-update-1.4
bsdiff-4.1         perl-5.6.1_15
```

Now that the package is gone, we must look for a newer version. The package we deleted, `mysql-client-3.23.58`, was the version shipped with FreeBSD 5.2.1 RELEASE. It was found in the following FTP directory. We show the contents of FTP mirror 9, but you are free to use whatever mirror you like;

```
ftp://ftp9.freebsd.org/pub/FreeBSD/ports/i386/packages-5.2.1-release/All
```

Packages compiled from the latest source code on the most recent version of FreeBSD, 5-STABLE, are found here:

```
ftp://ftp9.freebsd.org/pub/FreeBSD/ports/i386/packages-5-stable/Latest
```

In this directory we find a new version of the MySQL client, namely `mysql-client-3.23.58_3.tbz`. We can install this version using `pkg_add(1)` if we change our `PACKAGESITE` environment variable:

```
freebsd521# setenv PACKAGESITE ftp://ftp9.freebsd.org/pub/FreeBSD/ports/i386/packages-5-stable/I
```

We check to see what packages are available there:

```
ftp> pwd
257 "/pub/FreeBSD/ports/i386/packages-5-stable/Latest" is cwd.
ftp> ls mysql*client*
227 Entering Passive Mode (128,10,252,13,224,196)
150 Data connection accepted from 69.243.15.208:49184; transfer starting.
lrwxrwxrwx   1 ftpuser  ftpusers      33 Nov 17 21:01 mysql323-client.tbz -> ../All/mysql-clie
lrwxrwxrwx   1 ftpuser  ftpusers      30 Nov 17 21:01 mysql40-client.tbz -> ../All/mysql-clier
lrwxrwxrwx   1 ftpuser  ftpusers      29 Nov 17 21:01 mysql41-client.tbz -> ../All/mysql-clier
lrwxrwxrwx   1 ftpuser  ftpusers      29 Nov 17 21:01 mysql50-client.tbz -> ../All/mysql-clier
226 Listing completed.
```

Notice that the various MySQL clients now all have version numbers. We are interested in using `mysql323-client.tbz` with Barnyard.

Now we use `pkg_add(1)` to install the newest MySQL 3.x client listed in the STABLE package directory:

```
freebsd521# pkg_add -vr mysql323-client
looking up ftp9.freebsd.org
connecting to ftp9.freebsd.org:21
setting passive mode
opening data connection
initiating transfer
Fetching ftp://ftp9.freebsd.org/pub/FreeBSD/ports/i386/packages-5-stable/Latest/mysql323-client.
```

```
...edited...
extract: /usr/local/lib/mysql/libmysqlclient_r.so.10
extract: execute '/sbin/ldconfig -m /usr/local/lib/mysql'
extract: CWD to .
Runningmtree for mysql-client-3.23.58_3..
mtree -U -f +MTREE_DIRS -d -e -p /usr/local >/dev/null
Attempting to record package into /var/db/pkg/mysql-client-3.23.58_3..
Package mysql-client-3.23.58_3 registered in /var/db/pkg/mysql-client-3.23.58_3
```

When done, the new version is installed:

```
freebsd521# pkg_info | grep mysql
mysql-client-3.23.58_3 Multithreaded SQL database (client)
```

This process seems fairly simple, but there are problems. First, we had to manually verify that a new version of the MySQL client was available. Then we deleted it and reinstalled it. If any other packages (not source code like Barnyard) required mysql-client as a dependency, pkg\_delete(1) would have complained and not let us delete mysql-client. We could have forced deinstallation, but that's a sloppy system administration practice.

Should other applications have required the MySQL client, we could have deleted them, then deleted mysql-client, and reinstalled everything. Again, that is a lot of work. Fortunately, alternatives (described next) exist.

Incidentally, you can use sysutils/pkg\_tree to see package dependencies. Here we see what dependencies freebsd-update has:

```
freebsd521# pkg_tree freebsd-update
freebsd-update-1.4
  \__ bsdiff-4.1
```

We now know that if we tried to pkg\_delete(1) the bsdiff package, the system would complain because freebsd-update(1) depends on bsdiff(1):

```
freebsd521# cd /var/db/pkg
freebsd521# pkg_delete bsdiff-4.1/
pkg_delete: package 'bsdiff-4.1' is required by these other packages
and may not be deinstalled:
freebsd-update-1.4
```

A tool one can use to trim installed packages is sysutils/pkg\_cutleaves. This tool begins with "leaf" packages, asking if you want to remove them, followed by the packages upon which they depend. In the following example, we hit [return] whenever we see a package we want to keep. When we come to nmap(1), we decide to remove it with 'd':

```
drury:/root# pkg_cutleaves
Package 1 of 42:
```

```

XFree86-4.3.0,1 - X11/XFree86 core distribution (complete, using mini/meta-ports)
XFree86-4.3.0,1 - [keep]/(d)delete/(f)lush marked pkgs/(a)bort?
** Keeping XFree86-4.3.0,1.
...edited...
Package 21 of 42:
nmap-3.77 - Port scanning utility for large networks
nmap-3.77 - [keep]/(d)delete/(f)lush marked pkgs/(a)bort? d
** Marking nmap-3.77 for removal.
...edited...
Deleting nmap-3.77 (package 1 of 1).
---> Deinstalling 'nmap-3.77'
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 104 packages found (-1 +0) (...) dc

```

At this point, `pkg_cutleaves(1)` asks if we want to continue. If we do, we have the option of deleting a `nmap(1)` dependency, `pcre`:

```
Go on with new leaf packages ((y)es/[no])? y
```

```

Package 1 of 1:
pcre-5.0 - Perl Compatible Regular Expressions library
pcre-5.0 - [keep]/(d)delete/(f)lush marked pkgs/(a)bort? d
** Marking pcre-5.0 for removal.

```

```

Deleting pcre-5.0 (package 1 of 1).
---> Deinstalling 'pcre-5.0'
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 103 packages found (-1 +0) (...) dc
** Didn't find any new leaves to work with, exiting.
** Deinstalled packages:
nmap-3.77
pcre-5.0
** Number of deinstalled packages: 2

```

Since there are no longer any dependencies for `nmap(1)`, the process ends. Using a tool like `pkg_cutleaves(1)` allows us to trim down unnecessary packages very easily.

---

## Updating the Ports Tree, Part 1

The first step towards better package management is updating the FreeBSD ports tree. The contents of `/usr/ports` will remain the same as they are on the installation CD unless an administrator updates them. To let the system know what port versions are available, we must update the ports tree. This differs from other systems that contact a third party server to check for updates. Without updating the ports tree, the system only knows if installed packages are what it thinks of as "up-to-date."

We start by using `pkg_version(1)` to show what the system thinks of its package status:

```

freebsd521# pkg_version -v
bash-2.05b.007           = up-to-date with port
bsdiff-4.1              = up-to-date with port
cvsup-without-gui-16.1h = up-to-date with port
freebsd-update-1.4      = up-to-date with port
mysql-client-3.23.58_3  > succeeds port (port has 3.23.58)
perl-5.6.1_15          = up-to-date with port

```

The odd results for `mysql-client.3.23.58_3` are a result of deinstalling the package shipped with FreeBSD 5.2.1 and installing a new package for FreeBSD 5-STABLE. The `pkg_version(1)` tool sees the MySQL client version is newer than that found in the ports tree. It does this by checking version information in the Makefile found in the ports tree. If that information is unavailable, it looks in the `/usr/ports/INDEX-5` file (explained below).

The tool used to keep the ports tree up-to-date is the same tool used to keep the FreeBSD source tree up-to-date -- CVSup.

Here we add the GUI-less version of CVSup using `pkg_add(1)`. Note we are adding the package that ships with FreeBSD 5.2.1 RELEASE, not the version available in the 5-STABLE tree.

```

freebsd521# pkg_add -vr cvsup-without-gui
looking up ftp.freebsd.org
connecting to ftp.freebsd.org:21
setting passive mode
opening data connection
initiating transfer
Fetching ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/packages-5.2.1-release/Latest/cvsup-withou
...edited...
extract: /usr/local/share/cvsup/License
extract: CWD to .
Running mtree for cvsup-without-gui-16.1h..
mtree -U -f +MTREE_DIRS -d -e -p /usr/local >/dev/null
Attempting to record package into /var/db/pkg/cvsup-without-gui-16.1h..
Package cvsup-without-gui-16.1h registered in /var/db/pkg/cvsup-without-gui-16.1h

```

We must provide a suitable configuration file for CVSup, so we create `/usr/local/etc/ports-supfile` with the following contents:

```

*default host=cvsup9.FreeBSD.org
*default base=/var/db
*default prefix=/usr
*default release=cvs tag=.
*default delete use-rel-suffix

*default compress

```

```
ports-all
```

Notice the CVS tag is "." (dot). This shows that the ports tree is not tied to any particular FreeBSD version. While we could replace the . with a specific date, it makes more sense to upgrade to the latest ports tree available.

Before running CVSup, we should take a look at the two INDEX files in /usr/ports. The INDEX file is for FreeBSD 4.x, while the INDEX-5 is for FreeBSD 5.x. A look at the entry for nmap(1) in each is illustrative of the file's purpose:

```
freebsd521# ls -al IND*
-rw-r--r--  1 root  wheel  4251062 Nov 15   2003 INDEX
-rw-r--r--  1 root  wheel  4436222 Feb 14   2004 INDEX-5
```

```
freebsd521# grep "^nmap-" INDEX
nmap-3.48_1|/usr/ports/security/nmap|/usr/local|Port scanning utility for large
networks|/usr/ports/security/nmap/pkg-descr|eik@FreeBSD.org|security ipv6|
pcre-4.4|pcre-4.4|http://www.insecure.org/nmap/
```

```
freebsd521# grep "^nmap-" INDEX-5
nmap-3.48_1|/usr/ports/security/nmap|/usr/local|Port scanning utility for large
networks|/usr/ports/security/nmap/pkg-descr|eik@FreeBSD.org|security ipv6|
pcre-4.4|pcre-4.4|http://www.insecure.org/nmap/
```

Each entry shows the port name, followed by its location in the ports tree, a description, port maintainer, category, dependencies, and author URL. If there were different information for each version of FreeBSD, we would see it. Here there are no differences.

Now we run CVSup to sync our local copy of the ports tree with the version on the CVSup9 mirror. Again, this is an arbitrary choice of mirrors. The process makes every file in our /usr/ports directory the same as that on the CVSup9 mirror:

```
freebsd521# cvsup -g -L 2 /usr/local/etc/ports-supfile
Parsing supfile "/usr/local/etc/ports-supfile"
Connecting to cvsup9.FreeBSD.org
Connected to cvsup9.FreeBSD.org
Server software version: SNAP_16_1h
Negotiating file attribute support
Exchanging collection information
Establishing multiplexed-mode data connection
Running
Updating collection ports-all/cvs
Checkout ports/CHANGES
Delete ports/INDEX
Delete ports/INDEX-5
...edited...
Edit ports/Tools/scripts/plist
```

```
Add delta 1.6 2004.02.27.21.01.02 green
Edit ports/Tools/scripts/release/doi.sh
Add delta 1.5 2004.06.08.21.57.01 murray
Edit ports/Tools/scripts/release/scrubindex.pl
Add delta 1.2 2004.06.08.21.57.01 murray
Checkout ports/Tools/scripts/security-check.awk
Checkout ports/Tools/scripts/sunshar/Makefile
...edited...
Delete ports/x11-wm/xwmm/pkg-plist
Edit ports/x11-wm/yawm/distinfo
Add delta 1.2 2004.01.27.16.12.36 trevor
Applying fixups for collection ports-all/cvs
Fixup ports/textproc/xerces-c2/files/patch-ab
Shutting down connection to server
Finished successfully
```

On a FreeBSD 5.2.1 installation, this process takes a while. CVSup has to merge any changes between the ports tree of a 5.2.1 system with the latest version available from FreeBSD.org.

Notice that the process deleted the INDEX and INDEX-5 files. The easiest way to rebuild the INDEX-5 file is to execute 'make fetchindex' in /usr/ports:

```
freebsd521# make fetchindex
Receiving INDEX-5.bz2 (612684 bytes): 100%
612684 bytes transferred in 2.0 seconds (292.50 kBps)
freebsd521# ls -al INDEX-5
-rw-r--r--  1 root  wheel  5974063 Dec 23 15:09 INDEX-5
```

A FreeBSD 5.x system does not need the INDEX file used by FreeBSD 4.x.

An alternative to using 'make fetchindex' is Matthew Seaman's tool Portindex (sysutils/p5-FreeBSD-Portindex). I find 'make fetchindex' suits my needs, so I do not use Portindex.

An important file to reference whenever you consider upgrading installed applications is /usr/ports/UPDATING.

For example, here is the beginning of that file at the time of writing this article:

```
This file documents some of the problems you may encounter when
upgrading your ports. We try our best to minimize these disruptions,
but sometimes they are unavoidable.
```

```
You should get into the habit of checking this file for changes each
time you update your ports collection, before attempting any port
```

upgrades.

20041222:

```
AFFECTS: users of security/clamav, security/clamav-devel
AUTHOR: jylefort@brutele.be
```

The ClamAV database path has changed from /usr/local/share/clamav to /var/db/clamav. You should update the DatabaseDirectory keyword in /usr/local/etc/clamd.conf and /usr/local/etc/freshclam.conf.

If you carefully heed the advice in UPDATING your upgrades will proceed more smoothly.

---

## Manually Updating a Package Using the Ports Tree

Now that our ports tree is up-to-date, we can use `pkg_version(1)` to see what packages require updating:

```
freebsd521# pkg_version -v
bash-2.05b.007          <  needs updating (port has 2.05b.007_2)
bsdiff-4.1             <  needs updating (port has 4.2)
cvsup-without-gui-16.1h =  up-to-date with port
freebsd-update-1.4     <  needs updating (port has 1.6_1)
mysql-client-3.23.58_3 =  up-to-date with port
perl-5.6.1_15         =  up-to-date with port
```

It seems several are outdated. Earlier we showed how to use `pkg_delete(1)` to remove an outdate MySQL client, and `pkg_add(1)` to install a precompiled package of a newer version. Here we'll demonstrate a method using the ports tree to update `bash(1)`.

First we deinstall shells/bash2:

```
freebsd521# cd /usr/ports/shells/bash2
freebsd521# make deinstall
==>  Deinstalling for shells/bash2
==>  Deinstalling bash-2.05b.007
updating /etc/shells
```

Now reinstall it:

```
freebsd521# make reinstall
==>  Vulnerability check disabled
=> bash-2.05b.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch from http://ftp.gnu.org/gnu/bash/.
Receiving bash-2.05b.tar.gz (1956216 bytes): 78%
```

```
...edited...
==> Compressing manual pages for bash-2.05b.007_2
==> Registering installation for bash-2.05b.007_2
freebsd521# pkg_version -v | grep bash
bash-2.05b.007_2          = up-to-date with port
```

For this example I chose a port that did not have any dependencies to satisfy. Like the earlier example showing the `pkg_delete(1)` and `pkg_add(1)` of the MySQL client, this approach isn't as simple as one might like.

---

## Updating Packages with Portupgrade, Part 1

The best method created thus far to keep packages up-to-date requires the Portupgrade tool. You can install it using the ports tree as it is found in `sysutils/portupgrade`, or you can install it as a precompiled package. On a slow system, I recommend installing Portupgrade with the precompiled package. Portupgrade's Ruby dependencies are fairly heavy.

Portupgrade is well-documented, so here I chose to show how I use it. [3] This allows you to compare Portupgrade to the methods already demonstrated.

When you install Portupgrade, you will receive several tools that perform functions similar to those found in the native package management tools. The first is `portsdb(1)`, which builds the `INDEX.db` file used by Portupgrade:

```
janney:/usr/ports# portsdb -u
[Updating the portsdb <format:bdb1_btree> in /usr/ports ... - 12144 port entries found
.....2000.....3000.....4000.....5000.....6000.....7000...
.....8000.....9000.....10000.....11000.....12000. .... done]
```

Note: When writing this article using FreeBSD 5.2.1, the version of `portsdb(1)` packaged with Portupgrade did not like the format of the `INDEX-5` created by `'make fetchindex'`. I recommend using an updated version of the Portupgrade suite to keep your applications up-to-date. If you are using FreeBSD 5.3 RELEASE or higher, you should not have a problem. I have not encountered any problems with `'make fetchindex'` on FreeBSD 5.3.

Here is the new file:

```
janney:/usr/ports# ls -al INDEX.db
-rw-r--r--  1 root  wheel  12296192 Dec 23 18:23 INDEX.db
```

We now use the `portversion(1)` tool to see which packages need updating, similar to using `pkg_version(1)`:

```
freebsd521# portversion -v
[Rebuilding the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 10 packages found (-0 +10) .....
bash-2.05b.007_2          = up-to-date with port
bsdiff-4.1                < needs updating (port has 4.2)
```

```
cvsup-without-gui-16.1h      = up-to-date with port
frebsd-update-1.4           < needs updating (port has 1.6_1)
mysql-client-3.23.58_3     = up-to-date with port
perl-5.6.1_15              = up-to-date with port
portupgrade-20030723       < needs updating (port has 20041224)
ruby-1.6.8.2003.10.15     < needs updating (port has 1.6.8.2004.07.28_1)
ruby-bdb1-0.2.1           < needs updating (port has 0.2.2)
ruby-shim-ruby18-1.8.1.p2 < needs updating (port has 1.8.1.p3)
```

At this point, before actually updating these packages, I recommend consulting `/usr/ports/UPDATING`. Watch for advice on any of the packages you need to update. Here we find information on Ruby and Portupgrade:

20040226:

```
AFFECTS: i386 users of ruby and portupgrade
AUTHOR: knu@FreeBSD.org
```

Change the default version of ruby to 1.8 for i386.

If you are a ruby developer and want to keep ruby 1.6 as default, please add `RUBY_DEFAULT_VER=1.6` to `/etc/make.conf`.

Otherwise, please run the following series of commands to migrate to ruby 1.8:

- 1) Reinstall portupgrade manually (and as a result ruby 1.8 will be installed):

```
pkg_delete portupgrade-\*
(cd /usr/ports/sysutils/portupgrade; make install clean)
```
- 2) Reinstall everything that depends on ruby 1.6 to use ruby 1.8 instead:

```
portupgrade -fr lang/ruby16
```
- 3) Reinstall ruby 1.8 (because the previous step kills symlinks):

```
portupgrade -f lang/ruby18
```
- 4) Deinstall ruby 1.6 stuff (if you are paranoia):

```
pkg_deinstall -ri lang/ruby16
```
- 5) If the above commands do now work somehow and portupgrade starts causing `LoadError`, please reinstall portupgrade manually again. Whenever you get confused, you can always deinstall portupgrade and all the ruby stuff (run `"pkg_delete -r ruby-\*"`) and

reinstall portupgrade as a last resort.

This is very important advice, which we follow in the next example. Normally we do not have to observe such seemingly convoluted instructions to use Portupgrade to update our installed applications. However, when the package to be updated is Portupgrade itself, or one of its dependencies, extra attention is required.

We begin following the aforementioned instructions by removing Portupgrade from the system:

```
freebsd521# cd /var/db/pkg
freebsd521# pkg_delete portupgrade-20030723/
```

Now we reinstall Portupgrade and see that Ruby 1.8 is installed as a dependency for the new version:

```
freebsd521# cd /usr/ports/sysutils/portupgrade/
freebsd521# make install
==> Vulnerability check disabled
=> pkgtools-20041224.tar.bz2 doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch from ftp://ftp.iDaemons.org/pub/distfiles/.
Receiving pkgtools-20041224.tar.bz2 (104308 bytes): 100%
104308 bytes transferred in 3.7 seconds (27.36 kBps)
==> Extracting for portupgrade-20041224
=> Checksum OK for pkgtools-20041224.tar.bz2.
==> portupgrade-20041224 depends on file: /usr/local/bin/ruby18 - not found
==> Verifying install for /usr/local/bin/ruby18 in /usr/ports/lang/ruby18
Dependency warning: used OpenSSL version contains known vulnerabilities
Please update or define either WITH_OPENSSL_BASE or WITH_OPENSSL_PORT
*** Error code 1
```

```
Stop in /usr/ports/lang/ruby18.
*** Error code 1
```

```
Stop in /usr/ports/sysutils/portupgrade.
```

Here we have encountered a problem because we are running an unpatched FreeBSD 5.2.1 RELEASE system. Thanks to our knowledge gained from the previous article, we use FreeBSD Update to take care of the reported OpenSSL vulnerability, then try installing Portupgrade again.

```
freebsd521# cd /usr/ports/sysutils/portupgrade/
freebsd521# make install
==> Vulnerability check disabled
==> Extracting for portupgrade-20041224
=> Checksum OK for pkgtools-20041224.tar.bz2.
==> portupgrade-20041224 depends on file: /usr/local/bin/ruby18 - not found
```

```

===>   Verifying install for /usr/local/bin/ruby18 in /usr/ports/lang/ruby18
===>   Vulnerability check disabled
=> ruby-1.8.2-preview4.tar.gz doesn't seem to exist in /usr/ports/distfiles/ruby.
=> Attempting to fetch from ftp://ftp.iij.ad.jp/pub/lang/ruby/1.8/.
Receiving ruby-1.8.2-preview4.tar.gz (3602003 bytes): 20%
...edited...
===>   Registering installation for portupgrade-20041224

freebsd521# portupgrade -vfr lang/ruby16
--->   Session started at: Thu, 23 Dec 2004 20:22:11 -0500
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 13 packages found (-1 +4) (...).]
--->   Upgrade of lang/ruby16 started at: Thu, 23 Dec 2004 20:22:35 -0500
--->   Upgrading 'ruby-1.6.8.2003.10.15' to 'ruby-1.6.8.2004.07.28_1' (lang/ruby16)
--->   Build of lang/ruby16 started at: Thu, 23 Dec 2004 20:22:35 -0500
--->   Building '/usr/ports/lang/ruby16'
===>   Cleaning for ruby-1.6.8.2004.07.28_1
===>   Vulnerability check disabled
=> ruby-1.6.8-2004.07.28.tar.bz2 doesn't seem to exist in /usr/ports/distfiles/ruby.
=> Attempting to fetch from ftp://ftp.iij.ad.jp/pub/lang/ruby/snapshots/.
...edited...
===>   Installing for ruby18-bdb1-0.2.2
===>   ruby18-bdb1-0.2.2 depends on file: /usr/local/bin/ruby18 - found
===>   Generating temporary packing list
install -c -p -m 0755 bdb1.so /usr/local/lib/ruby/site_ruby/1.8/i386-freebsd5
===>   Registering installation for ruby18-bdb1-0.2.2
===>   Cleaning for ruby-1.8.2.p4
===>   Cleaning for ruby18-bdb1-0.2.2
--->   Removing temporary backup files
--->   Installation of databases/ruby-bdb1 ended at: Thu, 23 Dec 2004 20:35:23 -0500 (consumed 00:01:00)
--->   Cleaning out obsolete shared libraries
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 12 packages found (-1 +1) (...).]
--->   Upgrade of databases/ruby-bdb1 ended at: Thu, 23 Dec 2004 20:35:30 -0500 (consumed 00:01:00)
--->   Listing the results (+:done / -:ignored / *:skipped / !:failed)
      + lang/ruby16 (ruby-1.6.8.2003.10.15)
      + lang/ruby16-shim-ruby18 (ruby-shim-ruby18-1.8.1.p2)
      + databases/ruby-bdb1 (ruby-bdb1-0.2.1)
--->   Packages processed: 3 done, 0 ignored, 0 skipped and 0 failed
--->   Session ended at: Thu, 23 Dec 2004 20:35:33 -0500 (consumed 00:13:21)
freebsd521# portupgrade -vf lang/ruby18
--->   Session started at: Thu, 23 Dec 2004 21:41:17 -0500
--->   Reinstallation of lang/ruby18 started at: Thu, 23 Dec 2004 21:41:22 -0500
--->   Reinstalling 'ruby-1.8.2.p4' (lang/ruby18)
--->   Build of lang/ruby18 started at: Thu, 23 Dec 2004 21:41:22 -0500

```

```
---> Building '/usr/ports/lang/ruby18'
===> Cleaning for ruby-1.8.2.p4
===> Vulnerability check disabled
===> Extracting for ruby-1.8.2.p4
=> Checksum OK for ruby/ruby-1.8.2-preview4.tar.gz.
...edited...
---> Installation of lang/ruby18 ended at: Thu, 23 Dec 2004 22:01:08 -0500 (consumed 00:07:04)
---> Cleaning out obsolete shared libraries
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 12 packages found (-0 +1) . done]
---> Reinstallation of lang/ruby18 ended at: Thu, 23 Dec 2004 22:01:21 -0500 (consumed 00:19:59)
---> Listing the results (+:done / -:ignored / *:skipped / !:failed)
      + lang/ruby18 (ruby-1.8.2.p4)
---> Packages processed: 1 done, 0 ignored, 0 skipped and 0 failed
---> Session ended at: Thu, 23 Dec 2004 22:01:24 -0500 (consumed 00:20:07)

freebsd521# pkg_deinstall -vri lang/ruby16
...edited...
---> Listing the results (+:done / -:ignored / *:skipped / !:failed)
      + ruby16-shim-ruby18-1.8.1.p3
      + ruby-1.6.8.2004.07.28_1
---> Packages processed: 2 done, 0 ignored, 0 skipped and 0 failed
```

We have reached the end of the update process for Portupgrade and Ruby, advocated by /usr/ports/UPDATING. Remember that was an involved process because we were upgrading the very package-upgrade infrastructure itself.

---

## Updating Packages with Portupgrade, Part 2

The last invocation of Portupgrade was complicated. In this section I show the way Portupgrade is more commonly used.

The following represents a far more likely scenario: you run portversion(1) to discover what packages require updating. The -v switch means "verbose" and the -l "<" syntax tells portversion(1) to show only those packages needing to be updated:

```
freebsd521# portversion -v -l "<"
bsdifff-4.1          <  needs updating (port has 4.2)
freebsd-update-1.4  <  needs updating (port has 1.6_1)
```

Here we see bsdifff(1) and freebsd-update(1) require updating.

The easiest way to proceed is to tell Portupgrade to upgrade all old packages. We invoke it with switches best described by Dru Lavigne: "the -R will check the build dependencies and the -r will check the applications that depend upon the port being upgraded. This will prevent your system from having outdated dependencies and software incompatibilities." The -v switch means be verbose and the -a says act on all packages needing to be upgraded:

```
freebsd521# portupgrade -varR
---> Session started at: Thu, 23 Dec 2004 22:24:42 -0500
** No need to upgrade 'ruby-1.8.2.p4' (>= ruby-1.8.2.p4). (specify -f to force)
** No need to upgrade 'cvsup-without-gui-16.1h' (>= cvsup-without-gui-16.1h). (specify -f to for
** No need to upgrade 'mysql-client-3.23.58_3' (>= mysql-client-3.23.58_3). (specify -f to force
---> Upgrade of misc/bsdiff started at: Thu, 23 Dec 2004 22:25:24 -0500
---> Upgrading 'bsdiff-4.1' to 'bsdiff-4.2' (misc/bsdiff)
---> Build of misc/bsdiff started at: Thu, 23 Dec 2004 22:25:24 -0500
---> Building '/usr/ports/misc/bsdiff'
==> Cleaning for bsdiff-4.2
==> Vulnerability check disabled
=> bsdiff-4.2.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
=> Attempting to fetch from http://www.daemonology.net/bsdiff/.
Receiving bsdiff-4.2.tar.gz (7686 bytes): 100%
7686 bytes transferred in 0.0 seconds (354.67 kBps)
==> Extracting for bsdiff-4.2
=> Checksum OK for bsdiff-4.2.tar.gz.
==> Patching for bsdiff-4.2
...edited...
---> Installation of security/freebsd-update ended at: Thu, 23 Dec 2004 22:26:53 -0500 (consume
---> Cleaning out obsolete shared libraries
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 10 packages found (-0 +1) . done]
---> Upgrade of security/freebsd-update ended at: Thu, 23 Dec 2004 22:26:59 -0500 (consumed 00:
** No need to upgrade 'perl-5.6.1_15' (>= perl-5.6.1_15). (specify -f to force)
---> Listing the results (+:done / -:ignored / *:skipped / !:failed)
    - lang/ruby18 (ruby-1.8.2.p4)
    - net/cvsup-without-gui (cvsup-without-gui-16.1h)
    - databases/mysql323-client (mysql-client-3.23.58_3)
    + misc/bsdiff (bsdiff-4.1)
    - databases/ruby-bdb1 (ruby18-bdb1-0.2.2)
    - sysutils/portupgrade (portupgrade-20041224)
    - shells/bash2 (bash-2.05b.007_2)
    - misc/screen (screen-4.0.2_1)
    + security/freebsd-update (freebsd-update-1.4)
    - lang/perl5 (perl-5.6.1_15)
---> Packages processed: 2 done, 8 ignored, 0 skipped and 0 failed
---> Session ended at: Thu, 23 Dec 2004 22:27:03 -0500 (consumed 00:02:21)
```

This process updated freebsd-update(1) and bsdiff(1).

A final re-run of portversion(1) shows all packages are up-to-date with the installed ports tree:

```
freebsd521# portversion -v -l "<"
```

```
freebsd521#
```

---

## Updating the Ports Tree, Part 2

Earlier I presented a way to keep the ports tree up-to-date using CVSup. I recently became a fan of a new tool, written by FreeBSD Update author Colin Percival. It's called Portsnap, and it resides in `sysutils/portsnap`. [\[4\]](#) Colin's documentation is thorough, so I will demonstrate a sample run.

Install Portsnap using your favorite mechanism. Be sure to make a copy of the `/usr/local/etc/portsnap.conf` file as described by the tool's installation process.

Once installed, run 'portsnap fetch':

```
freebsd521# portsnap fetch
Fetching public key... done.
Fetching snapshot tag... done.
Fetching snapshot generated at Wed Dec 22 19:32:14 EST 2004:
Receiving 7dd7712493ec82aa7214e257d5aa9a9f5129af0 (31939389 bytes): 5%
31939389 bytes transferred in 137.9 seconds (226.17 kBps)
Extracting snapshot... done.
Verifying snapshot integrity... done.
Fetching updated snapshot tag... done.
Updating from Wed Dec 22 19:32:14 EST 2004 to Thu Dec 23 22:31:53 EST 2004.
Attempting to generate index via delta compression... success.
Generating list of updates needed... 396 files or ports need to be updated.
Attempting to fetch 370 patches... 370 fetched.
Attempting to apply patches... done.
Attempting to fetch 26 new files or ports... done.
```

The first time 'portsnap fetch' is run, the process retrieves a 30 MB compressed snapshot of the ports tree. Now we extract it:

```
freebsd521# portsnap extract
/usr/ports/.cvsignore
/usr/ports/CHANGES
/usr/ports/LEGAL
/usr/ports/MOVED
/usr/ports/Makefile
/usr/ports/Mk/bsd.autotools.mk
/usr/ports/Mk/bsd.emacs.mk
/usr/ports/Mk/bsd.gnome.mk
...edited...
/usr/ports/x11/yalias/
```

```
/usr/ports/x11/yelp/  
/usr/ports/x11/zenity/
```

Note: for future runs, you do not have to execute 'portsnap extract' again. Run 'portsnap update' instead. This will be demonstrated in the following section.

Next, execute 'make fetchindex' to download a new INDEX-5 and 'portsdb -u' to update the INDEX.db file:

```
freebsd521# cd /usr/ports  
freebsd521# make fetchindex  
Receiving INDEX-5.bz2 (612294 bytes): 100%  
612294 bytes transferred in 2.4 seconds (250.72 kBps)  
freebsd521# ls -al /usr/ports/IND*  
-rw-r--r--  1 root  wheel  5970410 Dec 24 09:05 /usr/ports/INDEX-5  
-rw-r--r--  1 root  wheel    2048 Dec 23 18:21 /usr/ports/INDEX.db  
freebsd521# portsdb -u  
[Updating the portsdb <format:bdb1_btree> in /usr/ports ... - 12148 port entries found  
.....1000.....2000.....3000.....4000.....5000.....6000.....7000...  
.....8000.....9000.....10000.....11000.....12000. .... done]  
freebsd521# ls -al /usr/ports/IND*  
-rw-r--r--  1 root  wheel  5970410 Dec 24 09:05 /usr/ports/INDEX-5  
-rw-r--r--  1 root  wheel 12295168 Dec 24 10:07 /usr/ports/INDEX.db
```

When done we run portversion(1) and find our packages are up-to-date:

```
freebsd521# portversion -v -l "<"  
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 12 packages found (-0 +2) .. done]  
freebsd521#
```

---

## My Common Package Update Process

Now that we've set up Portsnap, let's invoke it again. Assume 24 hours have passed, so port maintainers have made updates to the ports tree. The process which follows is the one I use to keep my packages up-to-date.

I never run it as an automated process, say via cron(1). Why? First, it is important to check /usr/ports/UPDATING to see if any of your applications require special handling. Second, if the upgrade doesn't go smoothly, I prefer to be there to handle it.

In brief, the process is as follows:

```
cd /usr/ports  
portsnap fetch  
portsnap update
```

```
make fetchindex
portsdb -u
portversion -v -l "<"
Check /usr/ports/UPDATING for information relating to my applications
portupgrade -varR
```

Here is a sample run showing the www/sarg package requiring an upgrade:

```
freebsd521# cd /usr/ports
freebsd521# portsnap fetch
Fetching updated snapshot tag... done.
Updating from Thu Dec 23 22:31:53 EST 2004 to Fri Dec 24 09:32:12 EST 2004.
Attempting to generate index via delta compression... success.
Generating list of updates needed... 47 files or ports need to be updated.
Attempting to fetch 46 patches... 46 fetched.
Attempting to apply patches... done.
Attempting to fetch 1 new files or ports... done.
```

```
freebsd521# portsnap update
Removing old files and directories... done.
Extracting new files:
/usr/ports/Mk/bsd.emacs.mk
/usr/ports/audio/sox/
/usr/ports/comms/obexapp/
/usr/ports/databases/fastdb/
...edited...
/usr/ports/x11-fonts/dejavu/
/usr/ports/x11-toolkits/open-motif/
```

```
freebsd521# portsdb -u
[Updating the portsdb <format:bdb1_btree> in /usr/ports ... - 12148 port entries found
.....1000.....2000.....3000.....4000.....5000.....6000.....7000...
.....8000.....9000.....10000.....11000.....12000. .... done]
```

```
freebsd521# portversion -v -l "<"
sarg-1.4.1          <  needs updating (port has 2.0.2)
```

Checking /usr/ports/UPDATING shows no issue for updating www/sarg.

```
freebsd521# portupgrade -varR
--->  Session started at: Fri, 24 Dec 2004 10:17:52 -0500
--->  Upgrade of www/sarg started at: Fri, 24 Dec 2004 10:18:00 -0500
--->  Upgrading 'sarg-1.4.1' to 'sarg-2.0.2' (www/sarg)
```

```
---> Build of www/sarg started at: Fri, 24 Dec 2004 10:18:00 -0500
---> Building '/usr/ports/www/sarg'
===> Cleaning for sarg-2.0.2
===> Vulnerability check disabled
=> sarg-2.0.2.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
...edited...
---> Installation of www/sarg ended at: Fri, 24 Dec 2004 10:24:37 -0500 (consumed 00:00:13)
---> Cleaning out obsolete shared libraries
[Updating the pkgdb <format:bdb1_btree> in /var/db/pkg ... - 13 packages found (-0 +1) . done]
---> Upgrade of www/sarg ended at: Fri, 24 Dec 2004 10:24:43 -0500 (consumed 00:06:43)
** No need to upgrade 'freebsd-sha1-5.3' (>= freebsd-sha1-5.3). (specify -f to force)
** No need to upgrade 'perl-5.6.1_15' (>= perl-5.6.1_15). (specify -f to force)
---> Listing the results (+:done / -:ignored / *:skipped / !:failed)
    + www/sarg (sarg-1.4.1)
    - sysutils/freebsd-sha1 (freebsd-sha1-5.3)
    - lang/ruby18 (ruby-1.8.2.p4)
    - net/cvsup-without-gui (cvsup-without-gui-16.1h)
    - misc/bsdif (bsdif-4.2)
    - security/freebsd-update (freebsd-update-1.6_1)
    - sysutils/portsnap (portsnap-0.3.1)
    - databases/mysql323-client (mysql-client-3.23.58_3)
    - databases/ruby-bdb1 (ruby18-bdb1-0.2.2)
    - sysutils/portupgrade (portupgrade-20041224)
    - shells/bash2 (bash-2.05b.007_2)
    - misc/screen (screen-4.0.2_1)
    - lang/perl5 (perl-5.6.1_15)
---> Packages processed: 1 done, 12 ignored, 0 skipped and 0 failed
---> Session ended at: Fri, 24 Dec 2004 10:25:03 -0500 (consumed 00:07:10)
```

That's it -- everything needing an upgrade has been upgraded.

---

## Creating Packages on One System and Installing Them Elsewhere

Imagine you run FreeBSD on a slow laptop (named "orr") and on a relatively fast server (named "janney"). It doesn't make sense to let the laptop labor while it builds packages if you could build them on the server. The following explains how I often build packages on a server and then install them on my laptop.

First, on the server, take the steps indicated earlier to update the ports tree, INDEX-5, and INDEX.db. You run the portversion(1) command and you see that three packages need to be upgraded:

```
janney:/usr/ports# portversion -v -l "<"
bash-3.0.15          <  needs updating (port has 3.0.16_1)
```

```

freebsd-update-1.6          <  needs updating (port has 1.6_1)
sudo-1.6.8.1                <  needs updating (port has 1.6.8.4)

```

You use all three on the server and the laptop. After reading /usr/ports/UPDATING, you invoke Portupgrade with an extra switch: -p

```
janney:/usr/ports# portupgrade -varRp
```

The -p switch tells Portupgrade to build packages for the applications it upgrades.

When done I had three new packages in /usr/ports/packages/All. This is where packages built during the upgrade or installation process are stored:

```

janney:/usr/ports/packages/All# ls -alt
total 135276
drwxr-xr-x  21 root  wheel      512 Nov 25 11:03  .
-rw-r--r--   1 root  wheel    545101 Nov 25 11:03  bash-3.0.16_1.tbz
drwxr-xr-x   2 root  wheel    1536 Nov 25 11:03  .
-rw-r--r--   1 root  wheel    10097 Nov 25 10:59  portsnap-0.2_1.tbz
-rw-r--r--   1 root  wheel    29219 Nov 25 10:58  freebsd-update-1.6_1.tbz
-rw-r--r--   1 root  wheel    99157 Nov 25 10:58  sudo-1.6.8.4.tbz

```

I expected bash, freebsd-update, and sudo to be there. These three were the packages identified by portversion(1) as being out-of-date. A new portsnap package was created as part of the upgrade process for freebsd-update, since portsnap depends upon freebsd-update to function.

Now that I had these new packages, I turned to updating my laptop, named "orr." I followed the five steps I first did for janney, and a 'portversion -v -l "<"' to see what needed updating. Laptop orr has a lot more packages installed compared to janney. If I want to avoid updating packages on orr via building from source through the ports tree, I need to obtain updated packages either from the FreeBSD project or from a system that's built the same packages.

Standard FreeBSD systems do not seem to have a means to build packages without installing them. (I believe OpenBSD has this capability.) Since I prefer to not maintain a "package builder" with every application I need, I take a dual-pronged approach. First, I wait until the FreeBSD project provides updated packages. Second, for critical applications that tend to be installed on many systems, I create my own packages. This second approach is what this process describes.

Based on my update on server janney, I know I can provide updated bash, sudo, and freebsd-update packages for laptop orr. To do that I NFS mount /usr/ports/packages/All on janney to orr, then run portupgrade(1). I use the -PP switch to tell portupgrade(1) to only use packages to update applications:

```

orr:/usr/ports# mount -t nfs janney:/usr/ports/packages/All /usr/ports/packages/All
orr:/usr/ports# portupgrade -varRPP
---> Session started at: Thu, 25 Nov 2004 11:26:02 -0500
** No need to upgrade 'tcpflow-0.21' (>= tcpflow-0.21). (specify -f to force)
---> Checking for the latest package of 'security/sudo'

```

```
---> Found a package of 'security/sudo': sudo-1.6.8.4.tbz (sudo-1.6.8.4)
---> Upgrade of security/sudo started at: Thu, 25 Nov 2004 11:26:09 -0500
---> Upgrading 'sudo-1.6.8.1' to 'sudo-1.6.8.4' (security/sudo) using a package
---> Updating dependency info
---> Uninstallation of sudo-1.6.8.1 started at: Thu, 25 Nov 2004 11:26:11 -0500
---> Fixing up dependencies before creating a package
---> Backing up the old version
---> Uninstalling the old version
---> Deinstalling 'sudo-1.6.8.1'
[Updating the pkgdb in /var/db/pkg ... - 167 packages found (-1 +0) (...) done]
---> Uninstallation of sudo-1.6.8.1 ended at: Thu, 25 Nov 2004 11:26:15 -0500 (consumed 00:00:00)
pkg_info: can't find package 'sudo-1.6.8.4.tbz' installed or in a file!
---> Installation of sudo-1.6.8.4 started at: Thu, 25 Nov 2004 11:26:15 -0500
---> Installing the new version via the package
Will not overwrite existing /usr/local/etc/sudoers file.
---> Removing temporary backup files
---> Installation of sudo-1.6.8.4 ended at: Thu, 25 Nov 2004 11:26:18 -0500 (consumed 00:00:03)
---> Cleaning out obsolete shared libraries
[Updating the pkgdb in /var/db/pkg ... - 168 packages found (-0 +1) . done]
---> Upgrade of security/sudo ended at: Thu, 25 Nov 2004 11:26:20 -0500 (consumed 00:00:10)
...truncated...
orr:/usr/ports# umount /usr/ports/packages/All
```

When done, bash, freebsd-update, and sudo were updated.

Notice that I did not have to run Portsnap, or 'make index', or portsdb(1) on the laptop prior to running portupgrade(1). Because I had mounted janney's /usr/ports over orr's /usr/ports, my upgrade process used the ports tree, INDEX-5, and INDEX.db files on the server.

---

## Addressing Security Issues in Packages

So far I've described ways to update packages without explaining why an administrator might want to take that action. The first reason is to address shortcomings in the older version of an application. A second reason is to provide additional features needed by users. A third, and potentially most important, reason is to fix security problems.

FreeBSD's Portaudit tool (found in security/portaudit) checks the security status of packages. Portaudit uses the Vulnerability and eXposure Markup Language, "an XML application for documenting security issues in a software package collection" like the FreeBSD ports system. [5] You can browse the FreeBSD or OpenBSD VuXML pages to see vulnerabilities recorded since the VuXML project began in late 2003.

Using the VuXML database is as simple as installing the Portaudit port. Be sure to begin with an up-to-date ports tree. Install Portaudit, and then run it as shown to check installed packages for problems. The -F flag tells Portaudit to fetch a new copy of the vulnerability database, while -d says show the database creation time and -a says check all installed ports/packages.

```
freebsd521# portaudit -Fda
Receiving auditfile.tbz (17217 bytes): 100%
17217 bytes transferred in 0.2 seconds (86.66 kBps)
New database installed.
Database created: Fri Dec 24 10:40:15 EST 2004
Affected package: mysql-client-3.23.58_3
Type of problem: mysql -- mysql_real_connect buffer overflow vulnerability.
Reference: <http://www.FreeBSD.org/ports/portaudit/835256b8-46ed-11d9-8ce0-00065be4b5b6.html>

Affected package: FreeBSD-502010
Type of problem: multiple vulnerabilities in the cvs server code.
Reference: <http://www.FreeBSD.org/ports/portaudit/d2102505-f03d-11d8-81b0-000347a4fa7d.html>
Note: To disable this check add the uuid to 'portaudit_fixed' in /usr/local/etc/portaudit.conf
```

1 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.

Since Portaudit found problems in mysql-client-3.23.58\_3, we should upgrade that port immediately. Since a newer version is not available, we might have to upgrade to the 4.0 or 4.1 MySQL client.

Portaudit also reported a problem with the CVS server code. However, we know this is fixed because we use FreeBSD Update to keep the OS up-to-date:

```
freebsd521# freebsd-update fetch
Fetching updates signature...
Fetching hash list signature...
Examining local system...
No updates available
```

We take the advice output by Portaudit and add this line to /usr/local/etc/portaudit.conf:

```
# this vulnerability has been fixed in your FreeBSD version
portaudit_fixed="d2102505-f03d-11d8-81b0-000347a4fa7d"
```

We re-run Portaudit after making this entry and removing the MySQL client:

```
freebsd521# portaudit -Fda
New database installed.
Database created: Fri Dec 24 10:40:15 EST 2004
0 problem(s) in your installed packages found.
```

Portaudit works with sysutils/pkg\_install-devel to warn sys admins when they try to install vulnerable software. In the following example, I

try to install Ethereum using an out-of-date ports tree. The Ethereum port wants to install version 0.10.0a, which has multiple problems.

```
janney:/usr/ports/net/ethereum# make
==>  ethereum-0.10.0a_2 has known vulnerabilities:
>>  multiple vulnerabilities in ethereum.
     Reference:
     cdf18ed9-7f4a-11d8-9645-0020ed76ef5a.html>
>>  Please update your ports tree and try again.
*** Error code 1
```

Stop in /usr/ports/net/ethereum.

If you are willing to accept the risks of a vulnerable application, you can disable the vulnerability checking manually. In the following example, another MySQL client has a problem:

```
neely:/usr/ports/databases/mysql40-client$ make
==>  mysql-client-4.0.18_1 has known vulnerabilities:
>>  MySQL insecure temporary file creation (mysqlbug).
     Reference:

>>  Please update your ports tree and try again.
```

This is a minor problem affecting only the 'mysqlbug' script, not core MySQL client functionality. We may not see a fix in the MySQL distribution until 4.0.19. Thanks to Michael Nottebrock, I learned how to install a port with a vulnerability:

```
neely:/usr/ports/databases/mysql40-client$ make -DDISABLE_VULNERABILITIES
==>  Vulnerability check disabled
>>  mysql-4.0.18.tar.gz doesn't seem to exist in /usr/ports/distfiles/.
...truncated...
```

The package is now installed.

Use make -DDISABLE\_VULNERABILITIES with care!

Portaudit can be used to check the status of a port before it is installed. Here we check for vulnerabilities in the Racoon port. By passing Portaudit the -C flag, we tell it to compare that specific port with the VuXML database.

```
janney:/usr/ports/security/racoon# portaudit -C
```

```
Port racoon-20040116a (security/racoon) should be marked FORBIDDEN:
- http://people.freebsd.org/~eik/portaudit/ccd698df-8e20-11d8-90d1-0020ed76ef5a.html
- http://people.freebsd.org/~eik/portaudit/40fcf20f-8891-11d8-90d1-0020ed76ef5a.html
- http://people.freebsd.org/~eik/portaudit/d8769838-8814-11d8-90d1-0020ed76ef5a.html
```

- <http://people.freebsd.org/~eik/portaudit/f8551668-de09-4d7b-9720-f1360929df07.html>

If we ran 'portaudit -A' in the /usr/ports directory, Portaudit would check for vulnerabilities in the entire ports tree.

Portaudit integrates with the scripts run on a daily basis to notify system administrators of security problems. For example, here is output from the "security run output" from one of my systems:

Checking for packages with security vulnerabilities:

Affected package: ruby-1.8.1\_2

Type of problem: ruby -- CGI DoS.

Reference: <<http://people.freebsd.org/~eik/portaudit/d656296b-33ff-11d9-a9e7-0001020eed82.html>>

Affected package: mutt-1.4.1\_4

Type of problem: Buffer overflow in Mutt 1.4.

Reference: <<http://people.freebsd.org/~eik/portaudit/67c05283-5d62-11d8-80e3-0020ed76ef5a.html>>

2 problem(s) in your installed packages found.

You are advised to update or deinstall the affected package(s) immediately.

This is an excellent feature that makes system administration much easier.

---

## Conclusion

I hope this article has helped you understand the different ways to keep FreeBSD applications up-to-date. It is by no means comprehensive, but by following it you hopefully can judge the different ways to keep your applications current.

## Acknowledgements

As readers provide feedback, I will credit them here.

## References

1. [www.freshports.org/misc/screen](http://www.freshports.org/misc/screen)
2. [www.sourceforge.net/projects/barnyard](http://www.sourceforge.net/projects/barnyard)
3. [www.onlamp.com/pub/a/bsd/2003/08/28/FreeBSD\\_Basics.html?page=1](http://www.onlamp.com/pub/a/bsd/2003/08/28/FreeBSD_Basics.html?page=1)
4. [www.daemonology.net/portsnap/](http://www.daemonology.net/portsnap/)
5. [www.vuxml.org/freebsd/index.html](http://www.vuxml.org/freebsd/index.html)

## Revision History

24 December 2004: First publication  
21 April 2005: Minor typo corrections

---

Copyright 2004 Richard Bejtlich. All rights reserved.

---

Return to [TaoSecurity.com](http://TaoSecurity.com) or [TaoSecurity.blogspot.com](http://TaoSecurity.blogspot.com)