MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

SPEED DAEMONS

INSIDE

VOL.4 NO.11
ISSUE 11/2011(28)
1898-9144

TrueNAS™ Storage Appliance: You are the Cloud

With a rock-solid FreeBSD® base, Zettabyte File System support, and a powerful Web GUI, TrueNAS™ pairs easy-to-manage software with world-class hardware for an unbeatable storage solution.

Expansion Shelves Available

TrueNAS™ 2U System

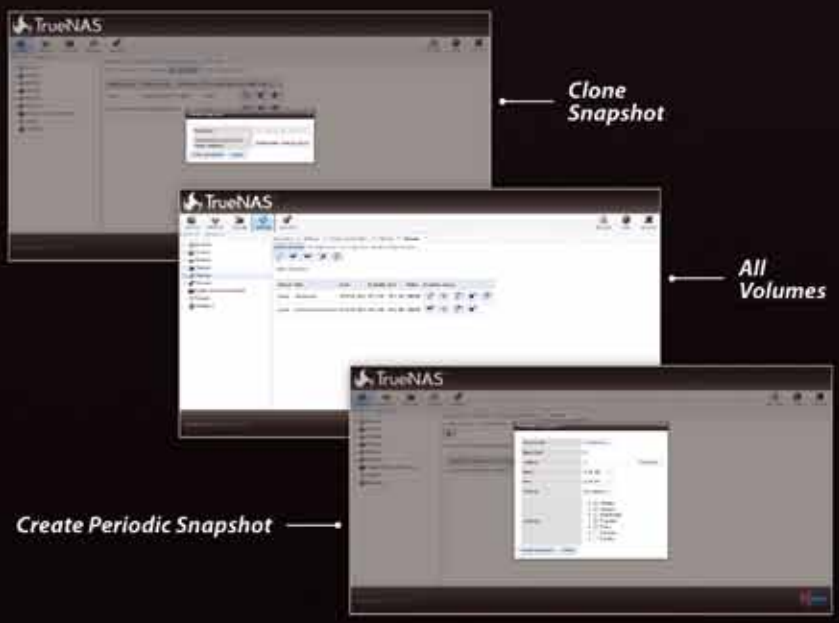
TrueNAS™ 4U System

FreeNAS 8 POWERED

Storage. Speed. Stability.

In order to achieve maximum performance, the TrueNAS™ 2U and 4U Systems, equipped with the Intel® Xeon® Processor 5600 Series, support Fusion-io's Flash Memory Cards and 10GbE Network Cards. Titan TrueNAS™ 2U and 4U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ Systems offer excellent capacity at an affordable price.

For more information on the **TrueNAS™ 2U** and **TrueNAS™ 4U**, or to request a quote, visit: **http://www.iXsystems.com/TrueNAS.**



TrueNAS™ 2U KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 12 Hot-Swap Drive Bays - Up to 36TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to triple parity
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

TrueNAS™ 4U KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 24 or 36 Hot-Swap Drive Bays - Up to 108TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to triple parity
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

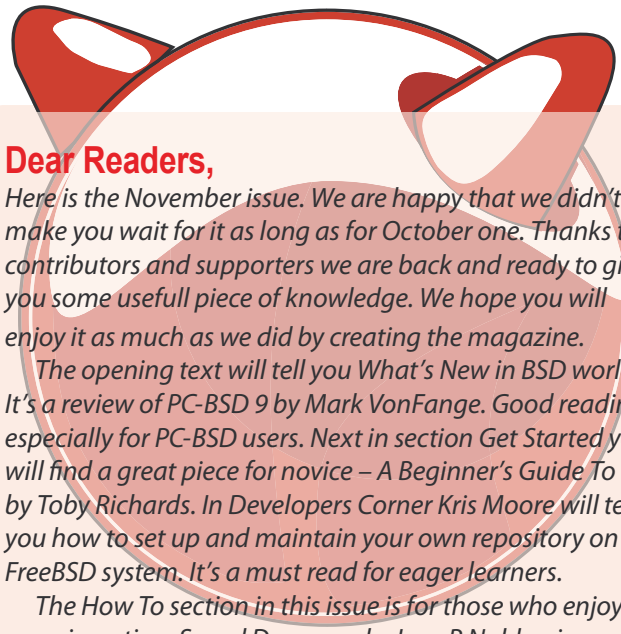
JBOD expansion is available on the 2U and 4U Systems

** 2.5" drive options available; please consult with your Account Manager*

Call iXsystems toll free or visit our website today!

1-855-GREP-4-IX | www.iXsystems.com

Powerful. Intelligent.



Dear Readers,

Here is the November issue. We are happy that we didn't make you wait for it as long as for October one. Thanks to contributors and supporters we are back and ready to give you some usefull piece of knowledge. We hope you will enjoy it as much as we did by creating the magazine.

The opening text will tell you What's New in BSD world. It's a review of PC-BSD 9 by Mark VonFange. Good reading, especially for PC-BSD users. Next in section Get Started you will find a great piece for novice – A Beginner's Guide To PF by Toby Richards. In Developers Corner Kris Moore will teach you how to set up and maintain your own repository on a FreeBSD system. It's a must read for eager learners.

The How To section in this issue is for those who enjoy experimenting. Speed Daemons by Lars R Noldan is a very good and practical text. By reading it you can learn how to build a highly available web application server with advanced networking mechanisms in FreeBSD. The following article is the final one of our GIS series. The author will explain how to successfully manage and commission a complex GIS project. Text not only for GIS series followers! We closed the section with article: Equip Your CA With a HSM For <50 Euros by Erwin Kooi. You may take a closer look at the security of Certificate Authority together with the author.

Now, some stuff for Admins: Terminals Served Up BSD Style by Toby Richards and OpenBSD Kernel Memory Pools: Monitoring Usage With Systat by Paul McMath. The first article presents two solutions for those who want to establish a BSD terminal server. The second one is addressed to more advanced users and explains how to understand memory usage statistics for kernel memory pools as they are displayed by the systat(1) command on OpenBSD.

In Let's Talk you will find a comparison of FreeBSD 8.2 and Ubuntu Server by Bill Harris. Worth reading to confront your own opinions or to gain some as well. We say farewell with EuroBSDcon Overview written from an organizers perspective by Jeroen van Nieuwenhuizen. Check what you have missed or see the event you participated in from a different perspective.

Wish you a good read and hope for a feedback!

Patrycja Przybyłowicz
& BSD Team

MAGAZINE BSD

Editor in Chief:
Patrycja Przybyłowicz
patrycja.przybylowicz@software.com.pl

Contributing:
Mark VonFange, Toby Richards, Kris Moore, Lars R. Noldan, Rob Somerville, Erwin Kooi, Paul McMath, Bill Harris, Jeroen van Nieuwenhuizen

Proofreaders:
Tristan Karstens, Barry Grumbine, Zander Hill, Christopher J. Umina

Special Thanks:
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski

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

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

Production Director:
Andrzej Kuca
andrzej.kuca@software.com.pl

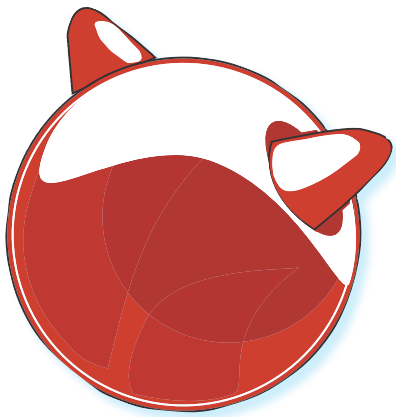
Executive Ad Consultant:
Ewa Dudzic
ewa.dudzic@software.com.pl

Advertising Sales:
Patrycja Przybyłowicz
patrycja.przybylowicz@software.com.pl

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*

Contents



PC-BSD 9 Turns a New Page

Since 2005, PC-BSD has striven to make BSD accessible to the desktop user. Offering a slew of improvements and added features, this release brings a more accessible BSD operating system than ever.

Continuing on the successful features of prior versions, like the intuitive Install *Graphical User Interface* (GUI), its PBI system and its comprehensive configuration menu, PC-BSD 9 brings added functionality and greater choices for its users.

Greater Options For The Desktop

The most obvious and significant change from prior versions is the addition of nine new desktop managers. Since it's inception, PC-BSD has utilized KDE as it's default and only fully supported desktop manager. While many consider KDE to be the most robust managers available, there are substantial numbers of users that prefer others for various reasons. This is very limiting and one of the great things about running an open source *operating system* (OS) is the flexibility it provides. Also, a lot open source users generally find their desktop manager of choice and stick with it due to familiarity. Others like to experiment and change things around. Up until recently, if you wanted to run PC-BSD, that meant you were forced to use KDE. Even then, you had to jump through hoops to get utilize any other options and you would lose some of the functionality built into PC-BSD.

That is all changing in version 9. The development team has separated the toolchain out from KDE and now everything is running in the Qt framework. The first four available desktop managers are GNOME, KDE, LXDE, and XFCE. All of these are fully supported in PC-BSD, meaning all utilities have been integrated into the desktop environment itself. The other five available desktop managers are Awesome, Enlightenment, IceWM, ScrotWM and WindowMaker. While these are not fully supported within the desktop environment, all the PC-BSD utilities are still available, though you may need to run commands via the Command Line in order to get them running. This is just a small barrier to new users, and power users should have no issues navigating through any of the available options.

For ease of use, all of these desktop managers are available for installation by simply clicking check-boxes during installation. No additional work is required for the user. The manager of choice is automatically installed and runs on the first reboot. Another great feature is that these managers are also available after initial installation. All you need to do is open up the PC-BSD Control Panel and Run the System Manager Utility. This brings up a SuperUser password prompt. From there you can uninstall and reinstall desktop managers at will from the System Packages tab and just reboot to switch to the new manager.

This gives the user a high degree of control over their system without requiring a great deal of technical knowledge. New users will need to be careful when switching to the unsupported desktop managers, as not all features are available via the UI. Even if a user does switch to a manager they aren't comfortable with, simply running the `pc-controlpanel` command in a terminal will bring up the PC-BSD Control Panel for them to switch to another. For those willing to persevere with those desktop managers, the PC-BSD Wiki page outlines the various steps needed to run unsupported utilities.

A Revamped Packaging System

One of the staples of PC-BSD is it's Push Button Installer or PBI feature. With ease of use in mind the PBI system makes installing applications a simple point-and-click operation. Similar to the .exe and .dmg file formats of Windows and OSX, PBI's are self-contained applications that take the user through the install process via a GUI system and avoiding any need for utilizing the *Command Line Interface* (CLI). This greatly helps reduce the learning curve for users to install their favorite applications.

Another issue the PBI system is designed to solve is dependency conflict and breakage. Different applications sometimes require different versions of the same dependency. This means that installing one package would may break the functionality of another. In order to avoid this, users would have to use workarounds, which can be daunting to those who don't know their way around the CLI.

One problem this caused was that each individual PBI had to have all of it's own libraries and dependencies contained within itself. This caused redundancy and substantially increased the necessary size of the programs as well as runtime memory. PC-BSD 9 has revamped the PBI system to utilize intelligent checks on the back-end of pbi's via a hash database in order to determine whether the needed libraries and dependencies already exist. This made it possible to avoid the redundancy issue and, consequently, lightened up the programs a great deal. These back-end checks go so far as to recognize when a library is no longer needed due to an uninstall. This creates a lighter PC-BSD with less bloat for the user.

Not only does PC-BSD 9 solve the problems of redundancy and breakage, it also revamped the way PBI's are downloaded and installed. This new method is a feature called AppCafe(TM). Prior to version 9, users had to go to pbidir.com in order to search for and download PBI's. AppCafe simplifies this whole process by integrating the PBI library into the desktop environment itself for an easy to use experience. This utility has added sanity checks automatically detects your system architecture and installs the proper PBI for you. Programs can be searched for by name via the search bar or you can browse through the conveniently categorized menu. Simply double click on the desired program and it will populate to the installed tab where you can view it's download and installation status. If you want to search for more programs, just switch back to the Browse tab and repeat the process. Another change to the PBI installation system under version 9 is that users can install most programs without a root password. This will be beneficial for systems with multiple users.

More Intelligence, More Versatility

Beyond improvements to the installation system, the PBI system has been revamped is in regard to updating applications as well. Before, upgrading a PBI involved downloading the entire new package. From PC-BSD 9 and beyond, when PBI's are upgraded, only changed files will be downloaded. This means saved time in downloads, which will especially help those on slower connections. In addition, the Upgrade Manager itself has been redesigned. In order to facilitate kernel updating, PC-BSD will now utilize the GENERIC FreeBSD kernel. This means that, in addition to the GUI, users will be able to upgrade via built-in `freebsd-update` command. This means that user accounts won't need root privileges to run upgrades. Furthermore, the Upgrade Manager will enable users to update the FreeBSD world and kernel, as well as packages, ports, PBI's and security updates.

As mentioned earlier, PC-BSD utilities now run independently of the desktop. This has enabled a whole new Control Panel that runs regardless of which manager you are running at the time. From it, you can access all manner of system settings, even the utilities for desktop managers you aren't currently running. Along with this new control panel comes a long overdue improvement to display configuration. Up until version 9, changing display resolution has required rebooting your system. With the new control panel, all you need to do is click on the Display icon in the hardware section of the menu. This will bring a prompt indicating that the current session will be exited and you will be brought to a display configuration menu after asking for your root password. From there, you can adjust your display resolution, color depth and select the

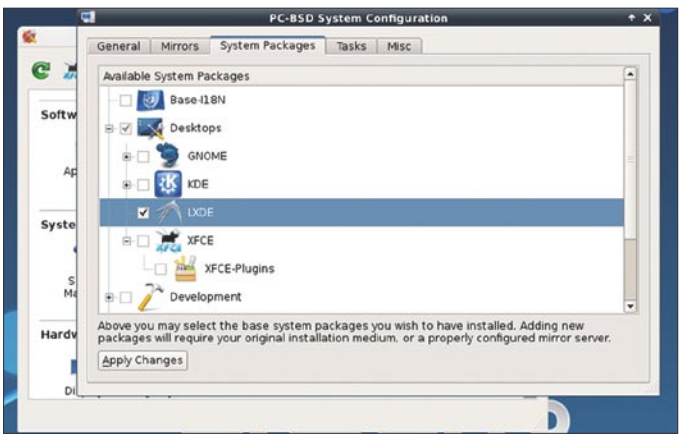


Figure 1. *PC-BSD Control Panel offers a GUI method of changing destop managers after installation*

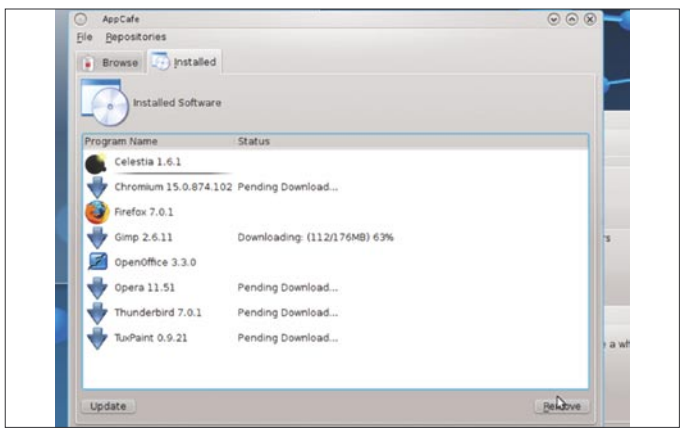


Figure 2. *AppCafe(TM) brings improved ease of use to the PBI installation system*

best driver for your video card. On the subject of display drivers, Nvidia drivers are now available for install via the install GUI as well as the control panel. The advanced tab of the display menu allows users to set up options such as refresh rates and dual-monitor configuration. In addition, once GEM/KMS support for FreeBSD is finalized, PC-BSD will offer improved Intel / ATI graphics support.

If those improvements were not already significant enough, there are even more key additions to PC-BSD 9. The first of which is the ability to install directly to a BootCamp partition. This should be a welcome addition for many mac users. Considering that OSX is largely based off of FreeBSD, this is a fitting addition to PC-BSD's repertoire. The Life Preserver feature brings additional support for system backups. While you can still simply add files to an external via the file manager or install one of the backup utilities available in AppCafe, Life Preserver provides an integrated utility for both manual and automated system backups to any external device with SSH and rsync enabled. It provides the user with a comprehensive GUI to handle things such as how often to back up, how many backups to keep, configuring which files and directories to back up, as well as the means to restore a backup.

Several other additions are also available on PC-BSD 9. First off is a wireless configuration tool that lists available networks and allows for a quick connection. Second, Myth TV and XBMC are available during or after install for those who want to run media from their PC-BSD system. Touchscreen drivers are now available and the development team intends for virtual keyboard and touchscreen capabilities from installation on at some point in the future. This means that you may see PC-BSD running on tablets some day. Version 9 also comes with Ipv6 enabled by default. Lastly, PC-BSD now includes a script to automatically detect USB devices for desktop managers that don't do so on their own. There are other minor improvements, tweaks and bug-fixes of course, but those are the most noteworthy.

Figure 3. *The Life Preserver Utility is opened by clicking the icon in the system tray, which launches a GUI based menu for adding new backups.The picture shows the utility being opened in the XFCE desktop manager*

A Look Under The Hood of PC-BSD

While offering intuitive GUI-based menus for virtually all aspects of system management, PC-BSD is still offers the full benefits of FreeBSD as well. Starting at version 7, PC-BSD releases have been tied in with FreeBSD releases. It has many added features, but PC-BSD utilizes the current FreeBSD source tree as its basis. This means that when you see a new major release of FreeBSD, you know that you are going to see many improvements and new features along with it. One of the biggest new features in FreeBSD 9 is the addition of USB3 support, which means improved performance of USB devices. Another advantage of FreeBSD is it's ZFS support. The newest release has updated ZFS to v28. This will likely be integrated into PC-BSD in version 9.1 or 9.2. ZFS is a robust open source file system with a high degree of data integrity due to capabilities such as dynamic striping, variable block sizes and cache management, as well as features such as encryption, deduplication and improved software RAID. FreeBSD 9 also offers multiple improvements for multi-processor support including NUMA support and large scale SMP support as well as 4k sector drives. These features bring enterprise capabilities to PC-BSD unseen in most desktop oriented OS distributions.

Other changes to FreeBSD include improved performance for virtual machines and laptops, better video support in linux compatibility mode, open source 3D hardware acceleration, and many others. These improvements all come on what is arguably the most stable and secure OS on the market, whether it's open source or proprietary.

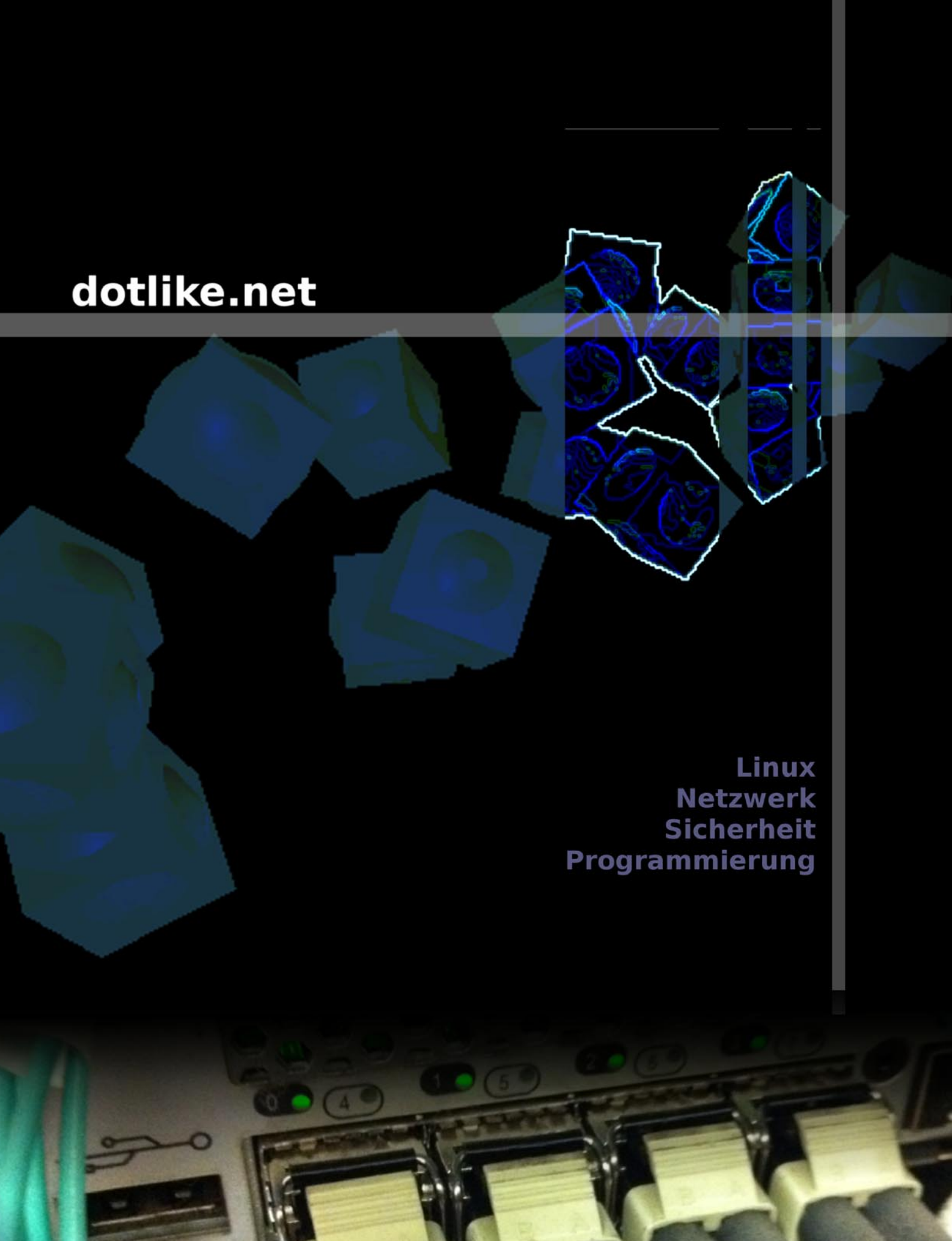
A Stable, Secure System, With Ease of Use in Mind

All in all one can say that the PC-BSD development team has made many major improvements over previous versions. From additional features to improved system management, PC-BSD 9 is an intelligent, versatile desktop operating system that will bring BSD to the desktop for many new users. With FreeBSD as a basis, one can rest assured that PC-BSD is a stable and secure operating system. With the intuitive features provided, users also gain many powerful tools for designing a desktop system that fits their needs. From new users to power users, home systems to enterprise systems PC-BSD 9 provides something for everyone.

MARK VONFANGE

Mark VonFange is the Professional Services Manager at iXsystems, providing oversight and coordination of its FreeBSD, PC-BSD, and FreeNAS support and development services. The Professional Services Team provides services ranging from mission critical support to software and firmware development to private consultation. Mark also develops internal and external documentation for division sales and marketing.

dotlike.net



Linux
Netzwerk
Sicherheit
Programmierung

A Beginner's Guide to PF

OpenBSD, FreeBSD, and PC-BSD use a built-in firewall called "Packet Filter" (PF for short). This article is intended for a PF beginner to get a beginner's understanding of how to use PF in OpenBSD. Several important intermediate and advanced concepts are intentionally left out.

What you will learn…

- A bare bones understanding of the /etc/pf.conf file for basic security.

What you should know…

- The ability to navigate your BSD's command line.
- Some knowledge about TCP & UDP port numbers.

This guide may apply to other *BSD operating systems, but I've not tested them all. I have tested PC-BSD, which means that these instructions will also work for FreeBSD. More advanced topics may be OpenBSD specific. Neither NetBSD nor Dragonfly BSD use PF.

I find that the OpenBSD documentation often throws too much information at the beginner from the get-go. The PF User's Guide is no exception. The following is a highly simplified subset of those instructions. The basic PF syntax that the beginner needs to know is this:

```
action [direction] [quick] [on interface] [af] [proto protocol] \
   [from src_addr] [to dst_port]
```

I have intentionally omitted more advanced/obscure options.

- action: This is either *block* or *pass*. Self explanatory.
- direction: This is either *in* or *out*
- quick: If a packet matches a rule specifying quick, then that rule is considered the last matching rule and the specified action is taken.
- interface: Self explanatory. It's *vic0* on VMware.
- af: *inet* for IPv4 or *inet6* for IPv6.
- proto: *tcp*, *udp*, *icmp*, or *icmp6*.

You can use the *all* directive in some places, but we're not going to talk about that other than for the final two rules in `/etc/pf.conf` file. In most cases, you probably want to specify what is allowed, and then block everything else. To do this, we make liberal use of the *quick* keyword. For incoming traffic, let's say that you only want to allow SSH & HTTP. Add these rules:

```
pass in quick on vic0 proto tcp to port 22
pass in quick on vic0 proto tcp to port 80
```

Notice that I simply exclude those directives that I don't need. In this case, I don't need to specify which version of IP or where the traffic is coming from because I want to allow this traffic on both IP versions and from anywhere.

Now, allowing pings is a little bit trickier. ICMP is a different beast than TCP or UDP. We do have to specify the *all* keyword to mean from anywhere, to any port. We also have to specify the type of ICMP packet (echoreq):

```
pass in quick inet proto icmp all icmp-type echoreq
```

Let's also only allow certain outgoing traffic. This could protect you in case a rogue user try to use an application that you don't want to be used. For this example, we'll let our server use:

- FTP (for commands such as `pkg_add`)
- SSH (remote access to other SSH servers)

- SNMP (so that sendmail can send us notifications)
- DNS (so that our server can resolve host names)
- HTTP (for using commands such as wget)
- Ping

Here are the rules:

```
pass out quick on vic0 proto tcp to port 20 # Used by FTP in PASV mode
pass out quick on vic0 proto tcp to port 21
pass out quick on vic0 proto tcp to port 22
pass out quick on vic0 proto tcp to port 25
pass out quick on vic0 proto udp to port 53
pass out quick on vic0 proto tcp to port 80
pass out quick on vic0 proto tcp to port 443
pass out quick inet proto icmp all icmp-type echoreq
```

To block all traffic that we haven't allowed above, simply add more lines to the end of the file:

```
block out all
block in all
```

Now we see the importance of the *quick* directive. With it, the packet filter immediately applies the rule for matching traffic (pass). Without it, the packet filter continues to look for matches until the end of the rule set. Since *block out* all & *block in* all match all traffic, we'd still be blocking everything without *quick*. Could we have avoided all of the business with *quick* by putting the block commands at the top? Probably. I just find that using *quick*, and then having the block commands at the bottom makes the config file easier to read and understand. Now, simply do `pfctl -f /etc/pf.conf` to reload and test your rules.

What happens when something goes wrong? If you think that your firewall rules are causing problems, then comment out your two block rules at the end of your `/etc/pf.conf` file. Issue `pfctl -f /etc/pf.conf` to reload the rules, and see if that fixes things. The telnet command is also your friend. Notice above that I have not allowed the telnet protocol (TCP port 23). The cool thing about the telnet utility is that it allows us to make a connection on any TCP port. We know that port 80 is HTTP, so we can test whether port 80 is working:

```
# telnet bsdmag.org 80
Trying 79.125.23.174...
Connected to bsdmag.org.
Escape character is '^]'.
```

[*CTRL*+*C*] will get you back to a prompt. Now let's try to connect to something that our firewall allows, but the remote host does not. We hang indefinitely at:

References

- For the official list of TCP and UDP ports, we can check with the *Internet Assigned Numbers Authority* (IANA): *http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml*
- For more documentation on PF, consult your *BSD's documentation:
- FreeBSD & PC-BSD: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls.html*
- OpenBSD: *http://www.openbsd.org/faq/pf/index.html*

```
# telnet bsdmag.org 22
Trying 79.125.23.174...
```

Again, [*CTRL*+*C*] gets you back to a prompt. Now that we know what to expect on allowed ports, let's try one that isn't allowed. Let's try TFTP:

```
# telnet bsdmag.org 69
Trying 79.125.23.174...
telnet: connect to address 79.125.23.174: No route to host
```

We know that there's a route to the host, so we must be blocking outbound TFTP connections with PF. That's key. When telnet tells you that there's no route to a host that you can otherwise connect to, then it's time to look at `/etc/pf.conf`.

What happens if you have a remote host, and you accidentally lock yourself out of SSH? Well, just like my byline says, that's one good reason to use bsdvm.com as a hosting provider. I can get to the VMware console to re-enable SSH. Otherwise, I'd be up a certain creek which shall remain nameless. As a disclaimer: I am in no way affiliated with bsdvm.com. I am just a fan. A big fan. Having console access is just so darned convenient for many, many tasks.

In the upcoming 5.0 release of OpenBSD we can look forward to using packet filter rules to prioritize certain traffic. More on that later. For now, you ought to have a grasp of the PF basics. Enjoy your new firewall!

TOBY RICHARDS

Toby Richards has been a network administrator since 1997. He considers himself to be a jack of all operating systems, but a true master of none. He feels this to be a mastery in its own right since he understands principles that are common to all operating systems. His articles are the product of teaching himself to become better with OpenBSD and PC-BSD. He simply writes about what he has learned most recently. For a hosting provider, he highly recommends bsdvm.com. They give you access to your VMware console so that you can re-install your OS at will, and with the settings of your own choosing.

Creating Your Own PBI Repository

With the major overhaul of the PBI system for the upcoming PC-BSD 9.0, it is now easier than ever to setup and run your own PBI repository, even on traditional FreeBSD systems.

This allows an individual or organization to maintain and distribute their own set of software packages, in a secure and easy manner. In this article we will take a look at the steps required for setting up and maintaining your own repository on a FreeBSD system.

One of the first steps in building a PBI repository is to first download and install the *pbi-manager* from the FreeBSD ports tree:

```
# cd /usr/ports/ports-mgmt/pbi-manager
# make install
```

After installing the pbi-manager, a variety of new command-line tools will become available for managing PBIs and repositories. Running a repository requires that all the built PBIs be digitally signed with openssl, for security and identification purposes. The next step is to generate a public / private key pair with openssl using the following command:

```
# openssl genrsa -out privkey.pem 4056
# openssl rsa -in privkey.pem -pubout > pub.key
```

After running these commands, you will be left with two files, `privkey.pem` and `pub.key`. The `privkey.pem` will be used to digitally sign your created PBI files, and the `pub.key` will be included with the repository configuration (*.rpo*) file. In order to begin creating your new repository you will need a FTP/HTTP/HTTPS location for hosting your repository's meta files, as well as a location(s) for clients to download PBI files.

These can be a public internet URL, or private LAN server, as long as it is accessible to your intended target audience.

Once you have the location URL, the next step is to create your repository *.rpo* file with the following command:

```
# pbi_makerepo --desc "My Example Repository" --key
 pub.key --mirror "ftp://ftp.example.org/pbi-files" --url
 "http://www.example.org/pbi-meta" /root
```

After running the command above, you will left with a `pbi-repo.rpo` file in the `/root` directory, or wherever you specified.

This file is all that is needed for clients to register and begin using your new repository. On the client system, this would be installed with a single command:

```
# pbi_addrepo pbi-repo.rpo
```

Once the repository is registered on the client system, the pbi daemon will automatically keep track of downloading and updating both meta-files and PBIs from the URL's you included in the repository configuration file. Meanwhile, back on the server side, we need to start creating some meta-data and PBI files so the clients have something to download and run. Luckily the *pbi-manager* includes utilities which make the process quick and easy. The first file we can create is meta-data for the application(s) we wish to distribute via this repository. This meta-data is used to give the client information about the applications available in your repository, such as categories, application names, descriptions, icons, authors, etc. First we will need to create some category for the application we want to distribute.

```
# touch pbi-meta-9
```

```
# pbi_metatool add --cat -n "Archivers" -d "File
Archivers and Utilities" -i "http://www.example.org/
                pbiicons/archivers.png" pbi-meta-9
```

The above commands will create your initial meta file, which will need to be named `pbi-meta-<Major Version Number>`. (I.E. 9, for FreeBSD 9.x, 8 for FreeBSD 8.x). Next we will need to add some information about the application we are going to distribute.

```
# pbi_metatool add --app -n "cabextract" -c "Archivers"
 -a "Stuart Caie" -d "Utility for reading and extracting
.cab files." -i "http://www.example.org/pbi-icons/cabextract.png"
 -k "cab,archive,extract" -l "LGPL" -t "Text" -u "http:
                //www.cabextract.org.uk" pbi-meta-9
```

In this example we have added the `Cabextract` meta-data, which includes the Author (`-a`), Keywords (`-k`), License (`-l`), Type (`-t`) and others. For a more complete description of all the available flags, you can run the command `pbi_metatool add`. Now that we have some initial meta-data, we can compress the file with *bzip2* and upload it to our meta location: *http://www.example.org/pbi-meta/pbi-meta-9.bz2*. On the client system, they will be able to use the `pbi_browser` command, or the AppCafe GUI (If on PC-BSD) to browse this meta-data.

With the initial meta-data in place, we can now begin the process of building PBI files. In this example we will be building the cabextract package from the FreeBSD ports tree. This command will expect that you have the FreeBSD ports tree already extracted and ready to be used.

More Information on the Web

- PBI-Manager Homepage: *http://wiki.pcbsd.org/index.php/PBI_Manager*
- PBI Module Builder Guide: *http://wiki.pcbsd.org/index.php/PBI_Module_Builder_Guide*
- PBI Developers Mailing List: *http://lists.pcbsd.org/mailman/listinfo/pbi-dev*

```
# pbi_makeport --sign privfile.key archivers/cabextract
```

Once the build is finished, you will be left with a resulting `cabextract-<ver>-<arch>.pbi` file which can now be uploaded to your PBI mirror server. Lastly we need to add this PBI to a pbi-index-9 file, which clients will download along with the meta-data. Adding this file can be done with the following commands:

```
# touch pbi-index-9
# pbi_indextool -f cabextract-1.2-amd64.pbi -u
 "archivers/cabextract-1.2-amd64.pbi" pbi-index-9
```

The `-u` option in this example will be the location of where you have uploaded the PBI file to your Mirror URL. In addition, we can also *bzip2* the `pbi-index-9`, and upload the `pbi-index-9.bz2` file to the meta URL, alongside the `pbi-meta-9.bz2` file we previously uploaded. Now your clients will be able to start downloading and installing PBIs from your repository, using the `pbi_add` command or the AppCafe GUI.

The *pbi-manager* includes a number of other tools and utilities for creating and managing PBI files. Repository maintainers may also find the `pbi_autobuild` command of interest, which monitors the ports tree, automatically rebuilding PBI's when the respective port version is updated. This command can be used with optional module configurations, allowing additional make options, desktop icons, helper scripts and other data to be used in the building of your PBI files. Using this and other commands, managing a PBI repository becomes easy to initialize and maintain. All the supplied commands include man pages with additional tips and options for power users and additional information may be obtained via the links below.

KRIS MOORE

Kris Moore is the founder and lead developer of PC-BSD. He lives with his wife and four children in East Tennessee (USA), and enjoys building custom PC's and gaming in his (limited) spare time. kris@pcbsd.org

Speed Daemons

Using advanced networking mechanisms in FreeBSD to build a high performance, highly available web application server.

What you will learn…

- What CARP is and how to use it
- Installing and configuring ifstated
- Brief introduction to the ZFS filesystem

What you should know…

- How to use the ports system
- Basic understanding of the command line
- Basic understanding of rc.conf
- Understanding of TCP/IP networking

It's no secret that web applications have recently been returning us full circle to a network computing model where your computer is only a mechanism used to connect you to your applications. Gone are the days of a single i486 running hundreds of websites. Web application servers have grown more powerful and infinitely more complex. Reliability of these web applications is at least as important, if not more so, than raw performance. The following details a web application server cluster Six Feet Up built for a Fortune 500 company specifically to address the issues of providing high performance, highly available web applications in a modern business setting.

Requirements

First, we need to review the requirements the application needed to meet.

- 99.9% application uptime
- Pages load in under 10 seconds to users on 1.5Mb/s internet connections
- User registration processing in under 20 seconds on 1.5Mb/s internet connections
- System must support up to 50,000 registered users
- System must integrate with other client-provided systems

System Tools: carp(4), zfs(1M), nfsd(8), sftp(1), […]
Major Ports: zope, solr, postgresql, nginx, varnish, haproxy, ifstated, […]
External Packages: plone, […]

Design

The application used in this project is Plone, an enterprise content management system capable of meeting the use requirements of the client. In addition to Plone, an Apache

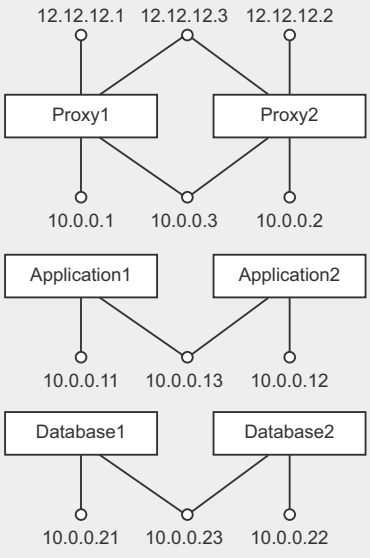


Figure 1. *Logical Layout of Hosting Stack*

Solr full text search was integrated to implement excellent search capabilities throughout the application. Going into the specifics of setting up the Plone application itself is beyond the scope of this article.

Due to how Plone runs, we have a relatively non-traditional web proxy chain installed using the nginx web server, varnish reverse proxy cache server, and haproxy load balancer. These services are in the ports tree. The specific installation and configuration of these services are also beyond the scope of this article.

There are six physical servers dedicated to the Plone application. (*Figure 1: Logical Layout of Hosting Stack*) In addition, there is a lower capacity server setup at another datacenter for disaster recovery/business continuity. Finally, there are a total of three ZFS based storage servers providing data storage and shipping capabilities to the application. The 6 physical servers used to host this Plone application are paired up as follows:

- Two Proxy Servers – These provide web caching and proxy services
  - Nginx web server – listening on ports: 80/443
  - Varnish caching reverse proxy server – listening on port: 3180
  - Haproxy load balancing with session affinity – listening on port: 3380

Listing 1. *An example carp setup in rc.conf*

```
proxy1 /etc/rc.conf
hostname="proxy1.mysite.com"
cloned_interfaces="carp0 carp1"
ifconfig_em0="inet 10.0.0.1 netmask 255.255.255.0"
ifconfig_em1="inet 12.12.12.1 netmask 255.255.255.0"
ifconfig_carp0="vhid 4 pass vhid4 advbase 3 advskew 50
                    10.0.0.3/24"
ifconfig_carp1="vhid 10 pass vhid10 advbase 3 advskew
                    50 12.12.12.3/24"
ifconfig_lo0="inet 127.0.0.1"

proxy2 /etc/rc.conf
hostname="proxy2.mysite.com"
cloned_interfaces="carp0 carp1"
ifconfig_em0="inet 10.0.0.2 netmask 255.255.255.0"
ifconfig_em1="inet 12.12.12.2 netmask 255.255.255.0"
ifconfig_carp0="vhid 4 pass vhid4 advbase 3 advskew 100
                    10.0.0.3/24"
ifconfig_carp1="vhid 10 pass vhid10 advbase 3 advskew
                    100 12.12.12.3/24"
ifconfig_lo0="inet 127.0.0.1"
```

- Two Application Servers – This is where the Plone CMS and Apache Solr full text search applications are handled
- Two PostgreSQL Database Servers – Plone operates with a PostgreSQL database back end, and maintains a reports database for data processed by the application.

Proxy services are configured identically across our front end proxy servers. The proxy servers serve as a gateway to the application servers and don't have very high loads even during peak traffic hours. The CARP protocol is being used to provide redundancy in the event of a server failure or in case of a service failure. CARP essentially allows you to assign a single IP address across multiple servers and provides an internal mechanism for failing over from one server to the next seamlessly. Setting up CARP requires recompiling the kernel. This section of the handbook: *http://www.freebsd.org/doc/handbook/kernelconfig.html* covers how to recompile the kernel. The option to enable CARP is: *device carp*. After restarting with the custom kernel, edit `/etc/rc.conf` similar to these examples: Listing 1.

The interface `em0` on each server is a unique private IP address. The `em1` interface has a unique public IP address. The `carp0` interface is a shared private IP address between the two servers. (*Figure 2: Proxy Server Logical Layout*) Finally the `carp1` interface is a shared public IP address. The vhid needs to be unique per shared address. It is possible to share an address across two or more servers. The server with the lowest advskew determines which is the MASTER and is serving the address. See the carp manpage for more information on the specific values.

In the event the MASTER proxy server fails, the BACKUP with the lowest advskew will automatically promote itself to the MASTER state and start serving traffic. This change over usually drops three or fewer packets making it nearly seamless to any end user, especially on a stateless protocol like http.

By itself, CARP is only concerned with the network state of it's peers, and does not detect the state of

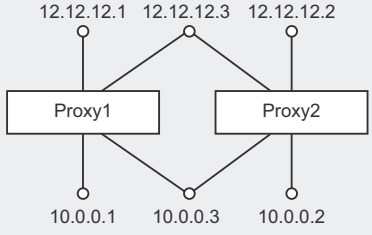


Figure 2. *Proxy Server Logical Layout*

Listing 2a. *An ifstated configuration on proxy servers*

```
# Set the default state to proxy_up:
init-state proxy_up

# carp1 is the main site:
proxy_carp_up = "carp1.link.up"
proxy_carp_init = "carp1.link.unknown"

# Ping our own proxy chain as well as the CARP master
                    every 15 seconds:

proxy_ping = '("/usr/local/bin/checkproxy.sh" every 15)'
proxy_remote_ping = '("/usr/local/bin/check_remote_
                    proxy.sh" every 15)'

# Define a state in which there is a problem with the
                    proxy chain
# on this machine. This state is set if the proxy_ping
                    command returns
# unsuccsessful (non-zero) status:

state proxy_down {
    init {
   run "sleep 10"

# Log the problem with the proxy chain, and attempt to
                    bring
# the interface down if we're the CARP master:

   run "echo 'date': Proxy chain down >> /tmp/
                    ifstate.log"
   if $proxy_carp_up {
        run "echo 'date': Proxy chain is down but I\'m
                    CARP master. \
      Attempting to become CARP slave by increasing
                    advskew \
      >> /tmp/ifstate.log"
       run "ifconfig carp0 advskew 250"
       run "ifconfig carp1 advskew 250"

    }

# Monitor the proxy chain on this machine, and set the
                    state to up
# if the ping script has a successful (0) exit status:

    if $proxy_ping {
   set-state proxy_up
    }

# If we can't get through to the site using this server,
                    or the
# CARP master go to panic state:

    if ! $proxy_remote_ping {
   set-state panic
    }

}
# End of proxy_down state


# Define the default state, with the proxy chain
                    responding normally.

state proxy_up {
    init {
        run "ifconfig carp0 advskew 50"
   run "ifconfig carp1 advskew 50"
   run "echo 'date': Proxy chain up >> /tmp/ifstate.log"
    }

# Set the state to proxy_down if the ping script returns
                    an unsuccessful
# (non-zero) status:

    if ! $proxy_ping {
   set-state proxy_down
    }

# If the proxy chain is up on this machine, but it is
                    not the CARP master
# and the site is unresponsive attempt to take over by
                    decreasing advskew:

    if ! $proxy_carp_up {
   if ! $proxy_remote_ping {
        run "echo 'date': My proxy chain is up but I\'m
                    CARP slave and \
      site isn\'t responding. Attempting to become CARP
                    master
      by decreasing advskew. >> /tmp/ifstate.log"
       run "ifconfig carp0 advskew 5"
       run "ifconfig carp1 advskew 5"
   }
    }
}
```

services hosted on the individual proxy servers. There are a number of different tools that can be used to test the proxy chain and take actions based on event failures. We chose to deploy ifstated (`/usr/ports/net/ifstated`) The description of this tool in the man page reads as follows: Quotation.

Listing 2b. *An ifstated configuration on proxy servers*

```
# End of proxy_up state


# Define the state in which the site is unreachable
                    because there is either
# a problem with the proxy chain on all proxy servers,
                    or there is a
# problem on the application servers.  We've avoided
                    restarting services
# for better debugging up until this point.  In this
                    case since it is a total
# outage we will restart the proxy chain and set
                    advskew low as a last attempt
# to automatically correct an outage situation.  In
                    order to keep from
# constantly changing CARP state causing erratic
                    behavior the only way to
# get out of this state is manual intervention.

state panic {
    init {
        run "echo 'date': ALL proxy servers are unable
                    to reach the site. \
      Entering panic state and attempting to restart
                    the entire
      proxy chain. MANUAL INTERVENTION IS REQUIRED AT
                    THIS POINT
      TO RESTORE AUTOMATIC FAILOVER >> /tmp/
                    ifstate.log"

   run "/usr/local/etc/rc.d/nginx restart"
   run "/usr/local/etc/rc.d/varnishd restart"
   run "/usr/local/etc/rc.d/haproxy restart"
        run "ifconfig carp0 advskew 5"
        run "ifconfig carp1 advskew 5"
    }
}
```

Quotation: A snip from man ifstated

The ifstated daemon runs commands in response to network state changes, which it determines by monitoring interface link state or running external tests. For example, it can be used with carp(4) to change running services or to ensure that carp(4) interfaces stay in sync, or with pf(4) to test server or link availability and modify translation or routing rules.

Ifstated was built with CARP in mind, and as a result is a lightweight tool that provides a lot of flexibility. The configuration on proxy1 and proxy2 is as follows: Listing 2.

There are two scripts called by ifstated every 15 seconds. These are checkproxy.py and *check_remote_proxy.py*. *Checkproxy.py* connects to the local unique public IP address and makes an http connect to the application. If the application does not respond the script exits with a non zero exit code. Ifstated will detect this and, depending on the current state, will take appropriate steps based on the configuration.

In simple tems, if proxy1 is the master, and *checkproxy.py* determines there is a problem with the local proxy chain, but the *check_remote_proxy.py* determines that the remote proxy is functioning properly ifstated will raise the local advskew of the carp interfaces to 250. This will usually result in proxy1 being demoted to BACKUP status. At the same time, proxy2 will have checked the local and remote proxy chain. If proxy2 is in state BACKUP, and gets a non-zero exit code when *check_remote_proxy.py* runs ifstated will lower the advskew on the local CARP interfaces to 5. This will under most circumstances promote proxy2 to MASTER status. The panic state occurs when neither proxy server has a functioning proxy chain. In this state ifstated attempts to restart each component in the proxy chain, and then lower the advskew to 5.

An external monitoring system is used to detect outages and send alerts. This configuration of ifstated is not generating any notifications to our systems administration staff. Ifstated is used only for attempting to keep the site alive in the event of service or server outages. This puts notification tasks in a place that is independent of both proxy servers and provides notifications in the case of site-wide outages as well.

The application servers are using a single internal CARP address for the Solr search process.

(*Figure 3: Database Server Logical Layout*) This provides a common address for each of the Zope application servers to connect to for internal search functions. Replication of the Solr data is managed by Solr itself via internal replication mechanisms. Ifstated is not running on the application servers as the advanced state

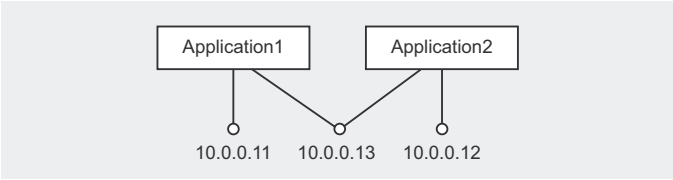


Figure 3. *Database Server Logical Layout*

Figure 4. *Database Server Logical Layout*

monitoring features aren't necessary here. The Zope application servers are load-balanced via haproxy on the proxy servers further up the stack.

Moving further into the stack, we have a pair of database servers running PostgreSQL 9.0. Each of these servers also has a unique private IP address, as well as a shared CARP address. (*Figure 4: Database Server Logical Layout*) PostgreSQL is handling replication internally using Database2 as a read-only hotspare.

Ifstated is running on these servers with the following configuration: Listing 3.

Similar to the proxy servers, ifstated calls an external script every 10 seconds to check the state of the PostgreSQL service. On these servers, if the database fails over we do not want to automatically attempt to recover like we do with the proxy chain. In the case of a database failure a systems administrator's attention is desired to prevent potential data corruption or loss.

Listing 3. *An ifstated configuration on database servers*

```
init-state psql_down
db_carp_up = "carp0.link.up"
db_carp_init = "carp0.link.unknown"
STATUS = '("/usr/local/bin/psql-status.sh > /dev/null
                    2>&1" every 10)'


state psql_down {
    if  ! $STATUS {
        run "echo 'date': psql not running >> /tmp/
                    ifstate.log"
    }
   if  $STATUS {
    if ! $db_carp_up {
        run "echo 'date': psql up and CARP is down,
                    changing state to slave >> /tmp/
                    ifstate.log"
        set-state db_slave
    }
    if $db_carp_up {
        run "echo 'date': psql up and CARP is up,
                    changing state to master >> /tmp/
                    ifstate.log"
        set-state db_master
    }
    }
}

state db_master {
    init {
        run "echo 'date': CARP link UP >> /tmp/
                    ifstate.log"

        run "echo 'date': promoting Postgres to master
                    >> /tmp/ifstate.log"
        run "/usr/bin/touch /var/db/pgsql/data/FAILOVER"
    }
    if ! $db_carp_up {
        set-state db_slave
    }
    if  ! $STATUS {
       run "echo 'date': psql changed to non-running
                    state >> /tmp/ifstate.log"
        set-state db_slave
    }

}

state db_slave {
    init {
        run "echo 'date': CARP link DOWN >> /tmp/
                    ifstate.log"
        run "/sbin/ifconfig carp0 advskew 250 >> /tmp/
                    ifstate.log"
    }
    if $db_carp_up
   set-state db_master

}
```

The distributed cluster of servers as well as the Plone application and Apache Solr full text search require file system data be shared across multiple machines. One of the customer's requirements was integrating the Plone application with some of their internal systems. To accomplish this, an sFTP server is setup on the proxy servers, allowing the customer to place a data file which will in turn to be processed by the application servers daily. In order to make the data available to the application servers, a ZFS filesystem, *ftpdata*, was configured on our ZFS-based storage server. The *ftpdata* filesystem is shared via NFS and is mounted on both proxy servers as well as both application servers. The Zope application also has data that needs to be available to all of the running application servers across both physical machines. This data is also stored on the ZFS storage servers using NFS mount, *zopedata*, which is mounted on both application servers. Finally we have a ZFS filesystem on the storage servers, *solrdata*, that is mounted on both application servers. Six Feet Up wrote a ZFS snapshot management package called *The Fortville Snapshot Distribution*, or FSD for short. FSD takes snapshots of the important filesystems and replicates them from our primary storage server onto our secondary storage server, as well as an off-site tertiary storage server. This provides redundancy of important data, as well as a way to roll back time to any five minute interval within the previous 24 hours, or any hour within the previous month without going to the backup system. As a side note, some of the work that went into FSD has been used in recent versions of FreeNAS 8.

The redundancy in the hosting stack has allowed multiple FreeBSD system and software upgrades while the site is live by upgrading half of the stack, triggering a failover, and then upgrading the other half of the stack.

Overall, the networking configuration and tools deployed have provided 99.995% uptime to date; which is significantly above the customer requirement we are required to maintain. The combination has allowed us to provide our client with the reliability and speed needed for a modern, high performance web application with a level of automation which keeps management overhead within reason.

LARS R. NOLDAN

Lars has over 14 years of technical experience, including open source server administration. For the last three years Lars has worked as a Systems Administrator for Six Feet Up – managing approximately 100 servers and another 200 jails running FreeBSD. He has extensive experience with web hosting technologies, focusing primarily on complex deployments in higher education and Fortune 500 companies.

A GIS Strategy

For Web-Enabled Business

In this final article in our GIS series, we will look at how to successfully manage and commission a complex GIS project

What you will learn…

- How to plan and commission transformational technology

What you should know…

- Previous FreeBSD GIS tutorials in this series

In the previous articles, we have looked at the core technological *nuts and bolts* that comprise an enterprise scale GIS. Rather than focusing on the technology, in this final article we will look at GIS from a different angle – culturally – and how to roll-out a successful system that could potentially revolutionize a business. While a substantial percentage of this article will be GIS specific, a lot of the wisdom contained here will also be applicable to other major projects so I hope there will be something here for everyone.

Do You Have Buy-in?

In these times of austerity the IT department is a critical source of innovation and creativity especially in the area of improving both the business model and generating efficiencies. As a core business group, IT are often the first to sense the change of mood at ground floor, in the marketplace and at management level. Yet, in some organizations layers of management and bureaucracy prevent effective communication. Unfortunately, it has been the authors experience in many businesses that there is a fundamental disconnect between board level (e.g. CEO) and IT unless the technologists have a strong representation via a Chief Technical Officer or a very senior manager who can be their advocate and argue their case. This ultimately ends in frustration for both groups, as IT feel nobody is listening and the executives feel that IT is not delivering. Ironically, both groups ultimately want to achieve the same goals, but are looking at the problem from radically different perspectives (albeit with entirely different motives sometimes) and this can end in deadlock or a breakdown of trust. From senior managements perspective, if the IT department cannot be trusted to handle the small matters (e.g. reliable servers, PC's etc.) there will be a lack of confidence in IT advising in the very sensitive area of corporate strategy, especially where technology will mean a major change in business practice. Likewise, if management undermine IT (We want system X from vendor Y regardless of the technological issues or long term cost) IT can get disillusioned from fire-fighting and vote with their goodwill or feet. Sometimes perception is more valued than reality, but the first step to a successful enterprise system is confidence and trust from all sides. The same applies in the other direction, if senior management do not have buy-in from the users, the change in business practice that technology brings will be resisted and potentially in some cases even sabotaged. Again, this is where IT can offer an invaluable service to the CEO in helping to identify how technology can make life easier and better for both the customer and the employee alike. Once trust is developed across the organization, the door is then open for everybody to contribute to innovation and moving the system forward.

Realistically, technology is relatively straightforward, the biggest hurdle is navigating through any cultural resistance so that we ultimately end up at the position of continuous improvement inside the organization and meeting our customers needs externally.

Integration (Both Human and Technological) is The Key

Legacy systems, vendor lock in, proprietary API's – all of these scenarios give IT departments a major headache. Thankfully, the Open Systems model is gaining ground, not only with Open Source, but vendors are realizing now that the era of the closed business model – where the customer is the milch cow – is rapidly coming to an end. Where there will be some vendors that cling to the proprietary model and fight their corner through the courts demanding justice for intellectual property rights, those with vision will be getting on with cutting code, peer reviews, and realizing that the sum is always greater than its parts. The struggle from the technological perspective is when a vendor refuses to take part in organizational change, and either makes this process financially prohibitive, or worse still refuses point blank for commercial reasons citing a catalog of reasons. Without naming any names, the author was once asked by his manager to get the views of a well known IT vendor on a relatively minor systems integration project which was of strategic importance. At great cost to the company (they were blue-chip so they could afford it!), 2 senior consultants were dispatched from the vendor for an afternoon meeting to discuss how this could be achieved. After much coffee and discussion, both consultants shook their head and their joint opinion was that a suitable product would not be available for 18 months at least. Needless to say, the author rolled out his own solution in 3 countries in 2 months, saving the organization on paper at least the cost of his annual salary per year. It was fortunate that the author had a good relationship with his line manager, as it was understood that the invitation to the vendor was very much a "box ticking" exercise, and the supplier had no intention of providing us as the customer with a workable solution. It would have been easy to feel undermined but once again it was a good illustration of corporate politics getting in the way of progress. If the level of trust had been greater, my employer would have saved the consultancy fee as well, and the potential pitfall of alerting the vendor that we were looking at other solutions.

Ultimately, there are always solutions available to the integration nightmare, the question is one of resources – money, time and innovation. Better still, don't fall for eloquent sales pitches, *smoke and mirrors* accounting and read the contract and licence agreement, preferably with a cynical lawyer who is on your side. That way you will not bear the pain further down the road when your organization wants or needs to change.

The other issue that always raises it's head is what happens when a key technological player *falls under a bus*? Rather than looking at this from the organizations perspective (redundancy, risk assessment, having a very propriety system etc.) I tend to see this as a cultural barrier. I may be wrong, but in my mind the organization group-think is more aligned in this scenario to *What if he leaves*? This to me is rather sad, as good IT staff are hard to come by, and really what the organization should be thinking is how can we use, retain, reward and expand our talent? Google manages to do this innovatively, along with a lot of blue-chips, and the concept of staff being your greatest asset should be embedded at the very core of corporate culture. Inevitably, there will be those who wish to progress their careers, sometimes at much cost to the organization, but in the authors experience the easiest employers to work for are those that naturally and successfully manage to *embed* their employees into not only the corporate culture, but their creative vision. Béla Hatvany (of Silver Platter fame) ran a small entrepreneurial start up in the '80's which I was privileged to be associated with, and it was evident that a major contribution to the success of the venture would be the employees, their synergy and personal commitment. When the business venture was heading towards hard times financially, all of the team offered without being asked what sacrifices they could make to turn the operation around. Some were even willing to work for a subsistence salary. How many corporates today could instill this level of loyalty and drive? The difference here is that the synergy between team members and Béla was so strong that his *problem* was their *problem*. Human chemistry, dynamics, personality and corporate culture all play their part here, but the energy that is found around a cohesive team that is allowed to *flex their wings* and get on with providing an excellent solution is truly something to be experienced. Too often, micro-management, not empowering individuals, and an unwillingness to allow them to make their own mistakes prohibits creativity, and the net result is the delivery of the average or mediocre rather than the superior.

I am not advocating here that management needs to take a totally hands off approach, rather that boundaries should be set and the right individuals chosen for the

team and they are then allowed to get on with their specialties. My best GIS functional specification to date has been a few sketches on the back of some scrap paper, this means more to me than all the Prince 2 project management software in the universe. Why? It came from the customer, we both know what we want to achieve and fortunately we have good synergy. Sometimes a more formalized structure is necessary, but I have yet to see an inspired team and hungry for success fail to rise to the challenge of solving the problem when confronted with a firm but respectful *You are not giving up until we have a solution*. While this meeting may be over coffee (or in the evening down at the pub), it is the *esprit de corps* that delivers, not the paperwork or the ultimatum.

What Does The Organization Want to Achieve?

People love maps. Google Earth and Google Maps are a prime example of *killer applications* that are very hard to put down. Tie this in with a good user interface, and users want to explore. This in turn raises the *What if*? scenario. Creativity begins to flow, and inevitably there will be a clash of civilizations, between the flexible analogue of spatial data and the previous method of measurement. Customers and requirement will be discovered in new areas, different perspectives will be developed and soon the previous ways of the organization will be found lacking. This is a dangerous time, as alongside the new technology with all its successes and revelations will be the temptation to totally replace the current orthodoxy with the newcomer and in the process throw the baby out with the bath water. Rinse, wash, repeat. This is the mantra the vendor doesn't want you to hear – built in commercial obsolescence.

The world will change, but provided core systems are able to both speak and listen (send and receive information to each other), the solution is simple – snap more technology on the end. Mr Vendor will not want his system to be this extrovert, so he will prefer to provide you with his introverted version of GIS with his *Customer Relationship Management* (CRM) system or whatever the latest understanding is on enterprise technology. If you couple best of breed with best of breed, you will end up with a thoroughbred, not a mongrel. So, the first essential requirement is that your GIS is truly open as indeed all your systems should be. The universe needs to expand – not shrink. If you want your GIS department (or any others for that matter) to live in a dark corner and not be a core part of your organization, a closed source solution is therefore ideal – for the vendor at least.

Once persuaded that a new channel is important, then two fundamental questions arise. What do you want to achieve and why do you want to achieve it? I am a firm believer that *killer app 2* in the enterprise will be the combination of telephony, CRM, GIS and the Internet. Mr Jones telephones, CTI picks up his caller ID and pulls his service record and location from the CRM. Meanwhile, GIS locates the nearest service representative, and at a click of a button the new service request is passed to the closest man in the field. Alternatively, CTI passes to GIS and the call goes directly to the nearest representative. The possibilities are endless. Mobile applications on the Iphone etc. can fire directly into a CRM / GIS combination for the reporting of issues in the field, allowing the company not only to compose a geographic picture of where the hot spots lie, but also develop a relationship with the customer. The reverse scenario is also true, if you have multiple mobile devices located in your area, activate them automatically according to region. Region A likes product B. Region B likes product A. You are a marketing agency, company C wants to penetrate both markets, so you beacon their sales representatives with tailored point of sale and marketing information as they enter their different territories.

What is more, all these platforms are available on *BSD. SugarCRM, Geoserver and Asterisk with some glue code and additional hardware would not be a bad start. Coupled with the true reliability, security and stability we all know, this would be a dream development platform.

The key to this is to get together and ask *What if*? Blue sky thinking and GIS make great team players.

Data Quality, Gathering and Data Logging

Paradoxically, while an organization is greater than the sum of its parts, no organization is greater than the quality of its' data. Poor quality customer data is the bedrock of disasters, poor customer PR, financial loss, and ultimately a *blame the system* mentality. As we all know, garbage in = garbage out, so as IT professionals we must continuously be on our guard to validate any input to our systems. How can this be dealt with in legacy systems however? Data matching and cleansing is very processor and time intensive, even with an army of validators the scale and complexity can be daunting. Often, the solution is just flush the database and start again. This goes against the grain for me, as there is bound to be something important in there to the business. If your organization is serious about your data, I would recommend the former route – cleanse – but use this as an opportunity for some positive PR and create a feedback loop between yourself and the customer. Using PR, and / or financial incentives, ask the

References

- Béla Hatvany 2000 Miles Conrad Memorial Lecture – *http://www.nfais.org/page/46-bela-hatvany-2000*
- NFAIS Annual Conference – How to prosper in the era of the Internet
- Robin Gawlik – Inspire Case Study – *http://www.lga.gov.uk/lga/aio/20246680*
- Barrow Borough Council – GIS system – *http://webgis1.barrowbc.gov.uk/webgis/bingis.html*

customer to register with you for a discount or a free offer etc. Again, you have a win-win scenario, you know what customers are enthusiastic about dealing with you, and you will get accurate data (you hope!). Better still, get customer services to phone the customer back just to make sure the details are correct.

Processes need to be in place not only to ensure Data Protection standards are met, but the quality of the data is excellent. Edgar Codd, the father of RDMS, states in rule 1 – *All information in the database is to be represented in only one way*. An organization should be aiming for disparate systems with a common key (e.g. a unique property ID or unique customer ID) across boundaries, rather than replicating the lower level data or functionality across systems. The problem is not the data that matches, it is the data that should match but doesn't.

GIS is Not Cheap

While Open Source is free, sadly infrastructure isn't. The one major lesson I have learned about GIS is that to do it properly you need decent hardware. Our largest tile-cache holds over 19 million files on one virtual guest, and ~ 10 million on the other, so JFS was the only practical option. I thought that 1TB of storage would be more than enough for my Virtual Machine images, but it would have been more prudent to have 5TB or a SAN. Geospatial data takes a lot of space and processing power, so specify your hardware with more than plenty of redundancy. Ironically, our Postgres database is actually very lean, the most CPU load is on the Geoserver processor (8 cores) and the physical size of the tile-cache.

Interfaces to GIS can be accurate up to 5 centimeters for professional kit, but in certain environments (e.g. wooded areas) getting a good lock on a satellite can be a hit and miss affair. A ground station to provide a reference signal may be required if pinpoint accuracy is essential. That aside, GIS location software is used for plotting Formula 1 cars during the race. Decent GPS kit is not cheap either, expect to pay > L10K for a highly accurate ground station in a ruggedised case excluding hand held units. Consumer grade units vary substantially, and your accuracy will depend on which country you are in and what satellites you have access to.

Thanks and Credits

The inspiration for this series of articles would not have come about if it were not for Copeland Borough Council moving from a proprietary GIS to an Open Source platform. The catalyst for this transformation rests with two individuals, Robin Gawlik of Barrow Borough Council and Julia Jackson at Copeland, both GIS Information Officers. The ability to manipulate complex equations, write great code, think spatially and maintain a good sense of humor are a rare combination of talents, especially as I am bereft of any sense of direction whatsoever. Without the developers, data trustees and the many others in the GIS community Open Source GIS would not be a reality. Our collective thanks goes to you all – you know who you are.

The Barrow GIS system can be found at the link below, and the Copeland version will be live in the very near future, as it is currently being tested in-house.

All opinions in this article are purely those of the author and do not necessarily represent either Copeland or Barrow Borough Council.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Equip Your CA With HSM For <50 Euros

With the recent breaches of Certificate Authorities (Comodo, Diginotar) I wanted to take a closer look at the security of my own Certificate Authority (CA). This CA is used for identification and authentication of servers, clients and users in my home network.

What you will learn…

- How to use smart cards to store private keys
- The use of OpenSSL with a different crypto provider

What you should know…

- Your way around a FreeBSD system
- The basics of a PKI
- Why you need a Certificate Authority

The keys for signing certificates are typically stored in text files on the file system. Access to the disk will reveal the keys. An attacker can make a copy and start issuing certificates outside my control. The primary design requirement is therefore that private keys will never be accessible in plain text.

Hardware Security Module

Enter HSM, hardware designed to keep the private key private. They come in various forms, from an appliance / PCI card to a USB token or smart card. PCI cards are typically encased in metal with features like automatic key deletion upon physical tampering. They are basically a dedicated computer running HSM software and using the PCI bus for interfacing with the host computer. Appliances are typically these PCI cards in an enclosure providing networking and rudimentary user control. HSMs can also have the form of a single chip. These do not have their own power supply (for deleting keys), but are still a small computer running HSM software. They can be embedded on smart cards, in USB tokens or integrated in other systems like the TPM chip on your motherboard.

Design Choices

For my HSM I decided to use smart cards, because they are cheap, readily available and easy to experiment with. The computer hardware is generic FreeBSD supported platform (i386 in my case, as I am working on putting it on an ALIX / NanoBSD installation). Attached is a Feitian SCR310 smart card reader (ftsafe-r310) and a Feitian PKI card (FTCOS / PK-01C). This reader is cheap, but any supported reader is fine (see *http://pcsclite.alioth.debian.org/ccid/section.html* for the complete list). This card was primarily chosen for its price and availability. It has lots of storage memory available (64k), but a limitation of 2048 bits for RSA keysize. Please do your own research on the types of readers and cards to fit your requirements.

For my production system, I will move to a Gemalto USB TR reader. It has an adapter so it can be mounted in a 3.5" floppy bay.

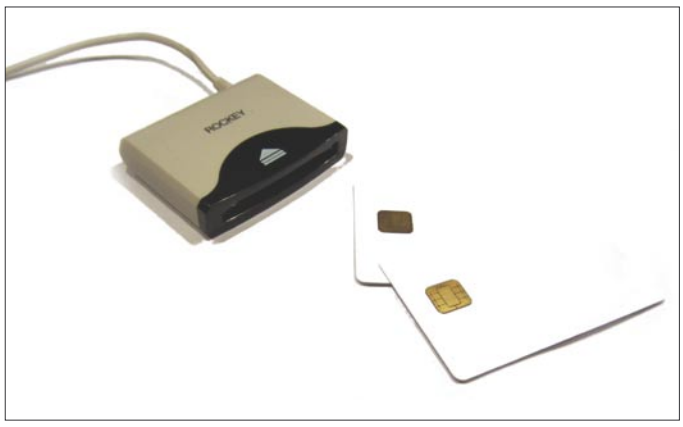


Figure 1. *Cards*

The software is a clean FreeBSD 8.2 with the following ports:

- `/usr/ports/devel/libccid` – interface for USB and serial smart card readers
- `/usr/ports/security/opensc` – tools for smart card managment (in PC/SC mode)
- `/usr/ports/security/engine-pkcs11` – engine for PKCS11 support in OpenSSL
- `/usr/ports/security/openssl` – tools for certificate management (in this case)

Due to the many, many dependencies, it will take some time to install. All default settings for the ports are fine.

There is a number of different CA designs possible. From very flat (the root CA issues all certificates directly) to very hierarchical (various sub and sub-sub CA's issue certificates on behalf of the root CA). My design has one root CA with a sub CA per functional area. This is done for security reasons. First of all, the root CA only has to sign the sub CA's. The use and exposure of the root CA's private key is therefore very limited. Next to that, the server sub CA never issues client certificates, so the VPN concentrator has to trust only the clients sub CA when validating certificates. Client certificates issued by the (possibly hacked) server- or user sub CA are not accepted.

Testing The Setup

After the installation of the software, it is time to plugin the reader and test the setup. Insert the card in the reader and run:

```
# opensc-tool --list-readers

 # Detected readers (pcsc)
Nr.  Card  Features  Name
0    Yes             Feitian SCR310 00 00
```

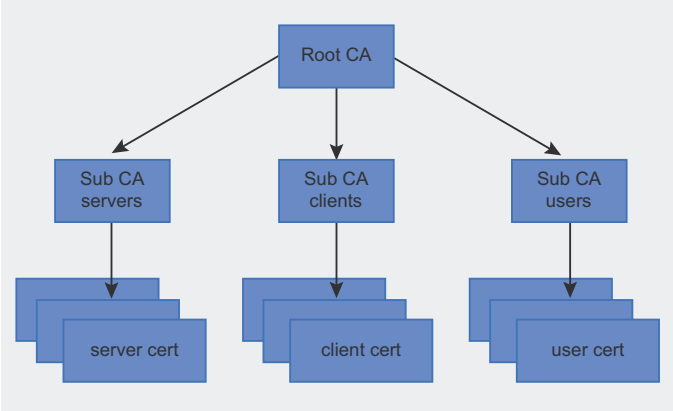


Figure 2. *CA-design*

The reader is recognized by the driver. Let's initialize the card by formatting it with a PKCS15 structure. PKCS15 is a cryptographic token information format standard originally designed by RSA. ISO 7816-15 now manages the card-related parts of this standard.

```
# pkcs15-init --erase-card
# pkcs15-init --create-pkcs15 \
  --profile pkcs15+onepin \
  --auth-id 01 \
  --pin 0000 \
  --puk 123456 \
  --label „ewak.net PKI"
```

This specific card supports only one user-PIN, so the pkcs15+onepin profile is used. More advanced cards can offer more users and role separation.

The PIN for the user (auth-id 01) is set to 0000. If you want to reset the PIN, you will need the PUK, which is set to 123456. The label is just a name to identify the card.

The card now has a PKCS15 structure and is able to store private keys and their certificates.

Listing 1. *Formatting the card*

```
# pkcs15-init --verbose \
  --store-private-key ewak.net_Sub_CA_servers.p12 \
  --format pkcs12 \
  --auth-id 01 \
  --cert-label "ewak.net Sub CA servers"

  Using reader with a card: Feitian SCR310 00 00
  Connecting to card in reader Feitian SCR310 00 00...
  Using card driver entersafe.
  Found ewak.net PKI
  About to store private key.
  Importing 1 certificates:
    0: /C=NL
       /ST=GLD
       /L=T******e
       /O=ewak.net
       /OU=Certificate Services
       /CN=ewak.net Sub CA servers
       /emailAddress=certificate.services@ewak.net
  User PIN [User PIN] required.
  Please enter User PIN [User PIN]:
```

Listing 2. *Contents of the card*

```
# pkcs15-tool --dump

  Using reader with a card: Feitian SCR310 00 00
  PKCS#15 Card [ewak.net PKI]:
     Version        : 0
     Serial number  : 3021303609260511
     Manufacturer ID: EnterSafe
     Last update    : 20110925214004Z
     Flags          : EID compliant

  PIN [User PIN]
     Object Flags   : [0x3], private, modifiable
     ID             : 01
     Flags          : [0x32], local, initialized,
                      needs-padding
     Length         : min_len:4, max_len:16, stored_
                      len:16
     Pad char       : 0x00
     Reference      : 1
     Type           : ascii-numeric
     Path           : 3f005015

  Private RSA Key [Private Key]
     Object Flags   : [0x3], private, modifiable
     Usage          : [0xC], sign, signRecover
     Access Flags   : [0x0]
     ModLength      : 2048
     Key ref        : 1 (0x1)
Native         : yes
     Path           : 3f005015
     Auth ID        : 01
     ID             : 53f70c3ea5be9aef27d959c134d8ebe
                      f77322786
     GUID           : {53f70c3e-a5be-9aef-27d9-
                      59c134d8ebef}

  X.509 Certificate [ewak.net Sub CA servers]
     Object Flags   : [0x2], modifiable
     Authority      : no
     Path           : 3f0050153100
     ID             : 53f70c3ea5be9aef27d959c134d8ebe
                      f77322786
     GUID           : {53f70c3e-a5be-9aef-27d9-
                      59c134d8ebef}
     Encoded serial : 02 01 02
```

Importing Keys

It is possible to have the card generate the private and public keys, so the private key will never be available in plain text. This is the most secure option, but losing the card will also make the keys unrecoverable.

Because I already have a CA in place, I started with importing the existing keys. Generating the keys on the card will be discussed at the end of this article. Importing the key material (in PEM or PKCS12 format) is straightforward (Listing 1).

The user PIN for auth-id 01 is defined during the initialization of the card, `0000` in this article. It can also be

Listing 3. *Finding the card slot*

```
# pkcs11-tool --module /usr/local/lib/opensc-pkcs11.so
                    --list-slots

  Available slots:
  Slot 0 (0xffffffff): Virtual hotplug slot
    (empty)
  Slot 1 (0x1): Feitian SCR310 00 00
    token label:   ewak.net PKI (User PIN)
    token manuf:   EnterSafe
    token model:   PKCS#15
    token flags:   rng, login required, PIN
                    initialized, token initialized
    serial num  :  3021303609260511
```

Listing 4. *Loading the engine*

```
# openssl
OpenSSL> engine dynamic \
  -pre SO_PATH:/usr/local/lib/engines/engine_pkcs11.so
                   \
  -pre ID:pkcs11 \
  -pre LIST_ADD:1 \
  -pre LOAD \
  -pre MODULE_PATH:/usr/local/lib/opensc-pkcs11.so

  (dynamic) Dynamic engine loading support
  [Success]: SO_PATH:/usr/lib/engines/engine_pkcs11.so
  [Success]: ID:pkcs11
  [Success]: LIST_ADD:1
  [Success]: LOAD
  [Success]: MODULE_PATH:/usr/local/lib/opensc-
                   pkcs11.so
  Loaded: (pkcs11) pkcs11 engine
```

supplied to the pkcs15-init tool by adding the `--pin 0000` option. Now let's see what is on the card (Listing 2).

The import was successful. The info for the PIN, the private key and the certificate are displayed. Record the ID of the private key, as we will need it later to tell OpenSSL which signing key to use. A shorter command for retrieving this ID is `pkcs15-tool --list-keys`. More keys can be added to the card. My card has three keys for signing server, client and user certificates.

OpenSSL and PKCS11

OpenSSL can be instructed to use an external crypto provider for generating and storing key material using the PKCS11 API, also known as Cryptoki. When using OpenSSL, we can specify keys on the smart card instead of a keyfile on the disk. To do this, we need the card's slot and the key's ID. The ID we found using the `pkcs15-tool -list-keys` command. The slot can be found with the pkcs11-tool command (Listing 3).

Slot 1 is our card reader with the correct smart card (`ewak.net PKI`) inserted.

Now start an OpenSSL prompt and load the PKCS11 engine: Listing 4.

The engine is loaded and can be used by specifying it in OpenSSL commands with the following parameters:

```
-engine pkcs11 \
-keyform engine \
-key slot_1-key_53f70c3ea5be9aef27d959c134d8ebef77322786
```

For example:

```
OpenSSL> req -new -engine pkcs11 -keyform engine \
  -key slot_1-key_d893afddc82b28fb539e975b2a3e18efc2f3c474 \
  -out cert.csr -text
```

(This will create a certificate signing request from the public key with the id shown.) The use of PKCS11 can be set in the OpenSSL configuration file by adding the following lines to your `/path/to/openssl.cnf` configuration file (Listing 5). All OpenSSL commands can now be run using the PKCS11 engine by specifying:

Listing 5. *openssl.cnf*

```
openssl_conf = openssl_init
[openssl_init]
engines = engine_section
[engine_section]

[pkcs11_section]
engine_id = pkcs11
dynamic_path = /usr/local/lib/engines/engine_pkcs11.so
MODULE_PATH = /usr/local/lib/opensc-pkcs11.so
init = 1
```

Listing 6. *Signing the certificate*

```
# openssl ca -config /etc/ssl/openssl.cnf \
  -engine pkcs11 \
  -keyform engine \
  -keyfile 1:53f70c3ea5be9aef27d959c134d8ebef77322786 \
  -cert ewak.net_Sub_CA_servers.crt \
  -in cert.csr -out cert.pem

  Using configuration from /etc/ssl/openssl.cnf
  engine "pkcs11" set.
  PKCS#11 token PIN:
  Check that the request matches the signature
  Signature ok
  Certificate Details:
  -snip- OpenSSL output omitted -snip-
  Sign the certificate? [y/n]:y

  1 out of 1 certificate requests certified, commit?
                  [y/n]y
  Write out database with 1 new entries
  Data Base Updated
```

Listing 7. *Contents of the card*

```
# pkcs15-tool --list-keys
Using reader with a card: Feitian SCR310 00 00
Private RSA Key [Private Key]
     Object Flags   : [0x3], private, modifiable
     Usage          : [0x4], sign
     Access Flags   : [0x1D], sensitive,
                    alwaysSensitive, neverExtract,
                    local
     ModLength      : 1024
     Key ref        : 1 (0x1)
     Native         : yes
     Path           : 3f005015
     Auth ID        : 01
     ID             : d893afddc82b28fb539e975b2a3e18e
                    fc2f3c474
     GUID           : {d893afdd-c82b-28fb-539e-
                    975b2a3e18ef}
```

```
-config /path/to/openssl.cnf -engine pkcs11 \
-keyform engine -key slot_<slot>-id_<id>
```

at the command line.

Signing Keys

The imported keys will be used for signing certificates. The OpenSSL command is not that much different from all available howto's on setting up a self-signed CA (Listing 6).

The signing request *cert.csr* is signed with the specified private key on the smart card. The signed certificate is stored in cert.pem. The certificate of the signing private key is stored in a local file called `ewak.net_Sub_CA_servers.crt`. It is also stored on the smart card, but since it is the public key and meant to be public, storing it on the local file system is more convenient and poses no security risk.

Note the different notation of the parameter for the key selection `-keyfile <slot>:<key id>`. OpenSSL will otherwise look for the keyfile specified in your openssl.cnf and fail to load the key.

Listing 8. *Creating the csr*

```
# openssl req -new \
  -engine pkcs11 \
  -keyform engine \
  -key slot_1-key_d893afddc82b28fb539e975b2a3e18efc2f
                   3c474 \
  -out cert.csr -text

  PKCS#11 token PIN:
  You are about to be asked to enter information that
                   will be
  incorporated into your certificate request.
  -snip- OpenSSL output omitted -snip-
```

Listing 9. *Storing the certificate*

```
# pkcs15-init --store-certificate cert.pem --auth-id
                   01 \
  --id d893afddc82b28fb539e975b2a3e18efc2f3c474

  Using reader with a card: Feitian SCR310 00 00
  User PIN [User PIN] required.
  Please enter User PIN [User PIN]:
```

Generating Keys

If you do not have an existing PKI or are willing to change its private keys, you can have the keys generated by and stored on the smart card. The keys will be generated with the true random number generator on the card and will never have touched you computer's memory or file system. This is done with the pkcs15-init tool.

```
# pkcs15-init --generate-key rsa/1024 --auth-id 01

  Using reader with a card: Feitian SCR310 00 00
  User PIN [User PIN] required.
  Please enter User PIN [User PIN]:
```

Both private and public keys are stored on the smart card. The keys are RSA keys of 1024 bits length. This public key can now be used for generating a certificate signing request. First we need the key ID (Listing 7). Create a signing request from the public key (Listing 8).

The *cert.csr* file can now be signed as shown before. The signed certificate can then be stored on the card using the pkcs15-init tool (Listing 9).

Conclusion

Smart cards can provide a low-cost and relative secure storage for private key material. This article focused on the use of smart cards in a CA environment. It did not focus on storing and using a user certificate for email signing, storing and using a client certificate for OpenVPN or storing and using SSH keys. These are all interesting options I will research further.

ERWIN KOOI

Erwin Kooi is an Information Security Manager for a large grid operator. He started with FreeBSD 4.5 and is an avid fan ever since.

Terminals Served Up BSD Style

I believe there is a splash screen on the PC-BSD installer that says, "PC-BSD: The Desktop Served Up BSD Style." It seems that the desktop BSD would be a good choice for a terminal server. My personal goal for this project is to provide a safe browsing environment to VIP's at my organization.

They refuse to let me filter out risky web sites. I figure if they do their browsing from a remote *nix session, then I don't have to worry about them getting malware on their Windows PC's. You may have your own reason for wanting a BSD terminal server.

There are two solutions to this goal: FreeNX (a compression solution for the X protocol) or XRDP (a product that wraps the VNC protocol inside Microsoft's Remote Desktop protocol). This article will show you how to use both solutions.

I am admittedly better with OpenBSD than with PC-BSD or FreeBSD. My understanding is that unlike OpenBSD, ports are preferred over packages under PC and Free. This article will therefore use the ports system to obtain the necessary software.

Prerequisites

- You should understand that PC-BSD is simply FreeBSD with KDE and an easy to use package manager installed by default. We can think of PC-BSD as the *desktop* version of FreeBSD.
- You need to know how to use the command line interface.
- You need to have a basic understanding of TCP and TCP ports.
- You need to have the PC-BSD ports collection installed. There is an option for this during the PC-BSD install process.
- It is assumed in this article that you are at the command line as root (indicated by the `#` prompt in the example commands).
- Understand that this is not a solution for screen sharing or simple remote administration. Our server will serve multiple networked X sessions to multiple users at the same time.
- You should have used – at least once – some window manager/desktop environment other than KDE or GNOME (eg. Fluxbox, FVWM, IceWM, XFCE, xterm, etc).
- You need to know the *name* of your network interface. Mine is em0. Yours may vary, especially if your PC-BSD is running in VMware.
- You need to know how to add users. I assume that you will add users as necessary to take advantage of your terminal server. Adding users will not be addressed in this article.

FreeNX

In most cases, FreeNX is going to be the superior method for turning any *nix computer into a terminal server. It's a faster protocol than VNC, and easier to get running on PC-BSD. First, let's install FreeNX:

```
# cd /usr/ports/net/freenx
# make install
```

There will be some option screens. Simply accept all of the default answers. Next we need to have the NX server start at boot time.

```
# /usr/local/NX/bin/nxsetup
```

Answer "N" to both of the questions.

Finally, let's open up TCP port 22 so that we can connect from a remote host. Edit your `/etc/pf.conf` file. Assuming you haven't tweaked the default firewall rules with *block* directives, then you can simply add this line to

the bottom of the file (remember, my interface is `em0`; yours may be different):

```
pass in on em0 proto tcp from any to (em0) port 22 keep state
```

Reload the firewall rules:

```
# pfctl -f /etc/pf.conf
```

Your FreeNX server is now ready. You can connect with the NX client for whatever operating system your clients use. Download the client of your choice at *http://www.nomachine.com/download.php*. Unfortunately, you will find that there are no BSD NX clients. Your ports collection has one based on PC-BSD's Linux compatibility feature, but it doesn't include all of the necessary Linux libraries. There are many threads to read about getting the net/linux-nx-client port to work, but I'm not going to address that in this article.

XRDP

As I said before, XRDP is inferior to FreeNX in most ways, but there are reasons that you may want to use it anyway. Here are some that I've encountered:

- FreeNX uses lossy JPG compression. You will sometimes notice blurriness in graphics.
- FreeNX can be slower than XRDP in certain environments. I don't know why, but I've speculated that one or all of these factors may play a role:
  - FreeNX's encryption.
  - The Cygwin layer of the Windows FreeNX client.
  - QoS or other scheduling features on your routers.
- RDP-specific features, such as being able to select the color depth that suits you.
- Lack of a FreeNX client for any BSD.

XRDP supposedly has a feature to serve actual RDP instead of VNC wrapped in RDP; however, you need something along the lines of *xorg-x11-server-rdp*.

This doesn't exist except as an RPM package for a very old version of OpenSUSE.

It doesn't work on current versions of OpenSUSE, and it certainly won't work under PC-BSD's Linux compatibility mode.

Let's get started. XRDP does not work well with KDE. You'll need to install an alternate environment (unless you intend to use TWM or xterm, which are included by default with PC-BSD).

If you know how to use ports, then install the environment of your choice. If you don't know how to use ports, or if you want the easiest environment for your luddite customers, then install GNOME from the software manager. Here is a list of environments that I tried: Table 1.

Once you have the environment of your choice installed, we'll install XRDP and its dependency, TightVNC:

```
# cd /usr/ports/net/xrdp
# make install
# cd /usr/ports/net/tightvnc
# make install
```

Let's open up the RDP port on your firewall. Edit `/etc/pf.conf`, and add the following line to the end of the file (remember to double-check the name of that network interface):

```
pass in on em0 proto tcp from any to (em0) port 3389 keep state
```

Now we need to force XRDP to use the environment of our choice. Edit `/usr/local/etc/xrdp/startwm.sh`. You will find a line near the top that reads:

Table 1. *Desktop Environments*

Environment	Result	Command to launch
KDE	Very messy & slow. Many artifacts.	startkde
GNOME	Good	gnome-session
Enlightenment	Fail	enlightenment
Fluxbox	Good	fluxbox
IceWM	Good, but you'll have to create menus from scratch.	icewm
Afterstep	Fail	afterstep
Windowmaker	Good	wmaker
TWM	Good, but difficult for novice	twm
XFCE	Good, but difficult for novice	xfce
Openbox	Good	openbox
Xterm	Good, but difficult for novice	xterm

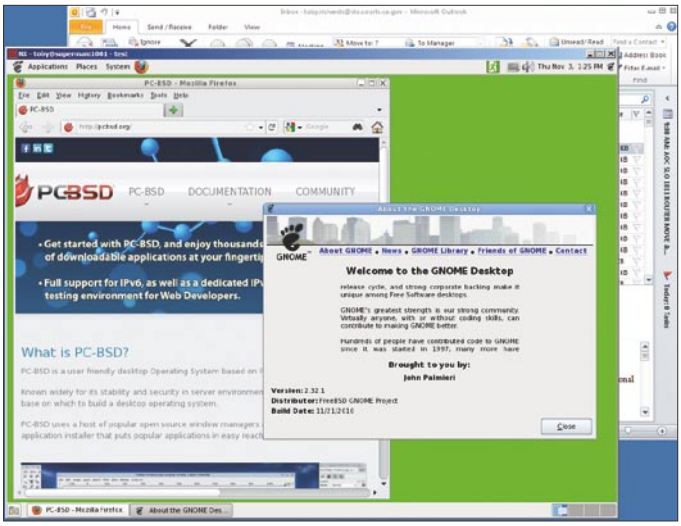


Figure 1. *PC-BSD on a terminal*

```
SESSIONS="gnome-session blackbox startxfce4 startkde xterm"
```

Edit that line to have ONLY the command for the environment of your choice. Use the table above to find the commands for some common environments. I'm using GNOME, so my line looks like this:

```
SESSIONS="gnome-session"
```

Next we're going to edit `/usr/local/etc/xrdp/xrdp.ini` to remove options that we don't need (and might confuse our luddite customers). Delete the following line, and delete every line below it:

```
[xrdp2]
```

By default, XRDP wants us to create groups where users will be added to if we want them to have access to XRDP. The easiest thing to do is just give access to everybody. Edit `/usr/local/etc/xrdp/sesman.ini`, and find these lines:

```
TerminalServerUsers=tsusers
TerminalServerAdmins=tsadmins
```

Just change *tsusers* to *users*, and change *tsadmins* to *wheel*. The last thing to do is have XRDP start at boot time. Add the following to the bottom of your `/etc/rc.conf` file:

```
/usr/local/sbin/xrdp
/usr/local/sbin/xrdp-sesman
```

To start XRDP without rebooting so that `/etc/rc.conf` can do its job:

```
# /usr/local/sbin/xrdp && /usr/local/sbin/xrdp-sesman
```

Now this is important: Newer versions of Microsoft's RDP client (mstsc) will cause XRDP to use 100% of your CPU until you reboot the server or kill the process like this:

```
# kill -KILL xrdp
```

The trick is to get a copy of rdesktop that has been precompiled for Windows. I found one here: *http://www.atomice.com/blog/?page_id=9*.

You can create batch files to launch rdesktop for your customers. The following would work fine (assuming that rdesktop is somewhere in your Windows PATH):

```
rdesktop -g 1024x768 xrdpserver.mydomain.com
```

You're Done!

Whether you've used FreeNX or XRDP, you're done. We now hope that our customers enjoy using their new terminal server.

TOBY RICHARDS

Toby Richards has been a network administrator since 1997. He considers himself to be a jack of all operating systems, but a true master of none. He feels this to be a mastery in its own right since he understands principles that are common to all operating systems. His articles are the product of teaching himself to become better with OpenBSD and PC-BSD. He simply writes about what he has learned most recently. For a hosting provider, he highly recommends bsdvm.com. They give you access to your VMware console so that you can re-install your OS at will, and with the settings of your own choosing.


Looking for help, tip or advice?
Want to share your knowledge with others?
BSD MAGAZINE
Give us your opinion about the magazine's content
and help us create the most useful source for you!

OpenBSD Kernel Memory Pools:

Monitoring Usage With Systat

This article explains how to understand memory usage statistics for kernel memory pools as they are displayed by the systat(1) command on OpenBSD.

What you will learn…

- Basic understanding of kernel memory pools
- Understand statistics regarding usage of kernel memory pools

What you should know…

- A page of memory – a fixed sized piece of memory (typically 4k in size) used by memory management system

A memory pool, in its simplest form, is a pre-allocated list of fixed sized pieces of memory. Memory pools can ensure availability of memory resources, allow for more efficient memory allocation and reduce memory fragmentation. Both NetBSD and OpenBSD implement memory pools in their kernels. FreeBSD uses a 'slab allocator' which is an extension of the functionality provided by memory pools. NetBSD also implements the extended functionality of slab allocation on top of its memory pool implementation.

The first section briefly describes the systat(1) utility. The second section explains how memory pools are more efficient for certain types of dynamic memory allocations and reduce fragmentation. The final section looks at the output of the systat(1) utility and explains the aspects of the OpenBSD pool implementation which are relevant for understanding the statistics as reported by systat.

Systat(1) Utility

The *sysat(1)* utility displays a range of system statistics in a format similar to the familiar `top(1)` command, taking over the entire terminal and refreshing the values at specific intervals. Related statistics are grouped together and displayed on a single screen called a *view*. Different views display different statistics; for example, the iostat view displays statistics related to I/O activity, and the malloc view displays statistics on kernel memory allocations which use `malloc(9)`. The vmstat view, which is the default, is unique in that it displays an overall picture of system activity including statistics on interrupts, memory usage, process activity and filesystem activity. On OpenBSD, one can switch between views using the right and left arrow keys.

The systat utility first appeared in 4.3BSD and is available on FreeBSD, NetBSD and OpenBSD. The implementations are only slightly different – the set of views available are not exactly the same, and navigation from one to the other is invoked differently. It is necessary to read the manpage on the particular system to learn the specific details of the implementation. Many of the statistics presented are available via other standard utilities from which the views take their names, e.g., netstat, iostat, vmstat. The particular systat view discussed in this article is the *pool* view and is only available on the OpenBSD implementation of systat. However, the same statistics can be seen using `vmstat(8)` with the `-m` option on NetBSD. On FreeBSD, `vmstat -z` will show statistics for *zones*, which are the equivalent of pools in the slab allocator implementation. This article does not discuss aspects of slab allocation.

Memory Allocations

Memory allocations in the kernel occur frequently. It is therefore important that these allocations be fast and

efficient. Since the kernel manages the entire system, if kernel performance is slow, it impacts the performance of the system as a whole. Many requests for kernel memory occur as user level programs make system calls. If satisfying these memory requests is inefficient, performance of user programs will degrade.

Typically, the kernel's general purpose memory allocator, `malloc(9)`, takes a request for an arbitrary piece of memory, rounds it up to a power of 2, and returns that amount of memory to the caller. For some types of allocations, this isn't inefficient. However, for memory requests which are large or of an odd size (i.e., not a power of 2), this can waste memory through fragmentation.

The kernel often requests pieces of memory of an identical size and which are used for identical purposes. For instance, whenever a new process is created on the system, the kernel must allocate memory for a structure to hold information about the process such as the PID, information about signals and CPU scheduling, etc. Similarly, a vnode structure must be allocated for vnode operations, socket structures for opening network connections, and filedesc structure for open files. Due to the various characteristics of these structures (in particular, their size and the frequency in which they are allocated), using the general purpose memory allocator to satisfy these requests is sub-optimal.

A memory pool is an optimization whereby multiple fixed sized pieces of memory are set aside for exclusive use by a specific part of the kernel. These fixed size pieces of memory are called 'pool items' and the part of the kernel which initializes and uses the pool is the 'pool owner'. A pool can guarantee that a minimum amount of memory will be available to satisfy requests from the pool owner independent of the available memory allocated by the general purpose allocator. Each pool is dedicated to serving requests for memory for one type of data structure (vnode structure, process structure, etc). The size of each pool item is the size fo the data structure to be stored in the pool (e.g., on OpenBSD, each item in the socket pool is 424 bytes, which is the size of the socket data structure). When the kernel needs one of these structures, it calls a routine which retrieves an item from the appropriate pool. When the structure is no longer needed (e.g., when the process exists, the socket or file closes), the item is returned to the pool where it can be used again the next time memory for that particular type of structure is needed.

An important optimization (whose implementation isn't limited to memory pools) is the pre-allocation of memory. This a achieved by making a single request for a large chunk of memory and carving up that chunk into smaller pieces which are ready for use. Taking the pool implementation in OpenBSD as an example, memory for a given pool is allocated in 4kb pages. If a single page can hold 20 pool items, then a single memory allocation for a pool creates 20 of these items which become immediately available for use by the pool owner. The same benefits exist for freeing memory. A single free operation releases multiple items with one function call. Also, because the items in a pool are of identical size, memory fragmentation is reduced when the page is divided up.

Output of the Systat(1) Utility

The remainder of this article looks at the interface and implementation of memory pools in OpenBSD and explains the pool usage statistics as they are displayed by the systat utility.

The `pool(9)` manpage describes the interface through which different parts of the kernel can initialize and destroy pools, get and return pool items, and set other parameters for pool management such as upper and lower bounds on the pool's resources (pool items or pages of memory). These interface routines, as well as the backend functions which implement them are in `sys/kern/subr_pool.c`. The various pool related structures and other declarations are in `sys/sys/pool.h`.

A kernel memory pool is initialized with the `pool_init()` function. `pool_init()` does not immediately allocate any resources. This step is normally deferred until the first request for an item in the pool. The `pool_prime()` function can be called to preallocate a specified number of items to the pool, but this function is not used in most cases. Although the GENERIC kernel creates many pools at system boot time, it is not unusual for several of them to never receive a request for items during the uptime of the system.

A pool dynamically manages its own resources based upon increased or decreased demand. Each pool tracks the number of items it has available for allocation. This number is decremented with each allocation and incremented as items are returned to the pool. If a pool is depleted (i.e., the number of available items is 0), additional memory will not be allocated until there is another request for a pool item. If, on the other hand, a lower bound has been set and fulfilling a request will bring the number of available items below the lower bound, then additional memory will be allocated to the pool before the request is completed.

Figure 1 is an example of output from the `systat pool` command. The top line shows the number of users on the system, the load averages and system date/time. In the middle of this line, in parentheses, is the number of lines visible in the terminal window and the total lines output in

this view. On this machine, there are 123 pools, but only the first 46 are visible. The up/down arrow keys or the pageup/down keys will either scroll or page up/down to display the additional lines of pool statistics. The default is to refresh the screen every 5 seconds with new values. The `p` command will pause the refresh operation.

The output displays values pertaining to pool items and pool pages. The values in the first four columns after NAME apply to pool items; the last 5 columns of output pertain to pages of memory. The following briefly describes the values in the columns:

- NAME – the name of the pool
- SIZE – the size of the pool item in bytes
- REQUESTS – total number of requests for this pool item since the pool was initialized
- FAIL – total number of failed requests
- INUSE – total number of pool items currently allocated and in use
- PGREQ – total number of pages allocated to this pool since initialization
- PGREL – total number of pages released from the pool since initialization
- NPAGE – total number of pages currently allocated to the pool
- HIWAT – the most pages concurrently allocated to the pool since initialization
- MINPG – minimum number of pages to retain in the pool
- MAXPG – maximum number of idle pages to keep in the pool (default value is 8)
- IDLE – number of idle pages, i.e., pages from which no items have been allocated

There are four ordering options which are invoked by single key commands:

- `N` – order alphabetically by NAME
- `Q` – order by value in the REQUEST column
- `Z` – order by total amount of memory allocated from a given pool (SIZE * INUSE)
- `P` – order by value in the NPAGES column

The `r` command will reverse the given sort order. The `o` command will cycle through the orderings as listed above.

The name of the pool and the size of the items are set when the pool is initialized with the `pool_init()` function. *REQUESTS* is a running total of requests for items since the pool's initialization.

```
   14 users    Load 0.12 0.16 0.12  (1-46 of 123)      Wed Oct 26 19:05:16 2011

NAME           SIZE REQUESTS     FAIL    INUSE    PGREQ    PGREL    NPAGE    HIWAT    MINPG    MAXPG     IDLE
allocdirect     128        0        0        0        0        0        0        0        0        8        0
allocindirp     104        0        0        0        0        0        0        0        0        8        0
amappl           72  3808159        0    29400      662        0      662      662        0       75        7
anonpl           24  4776210        0    57826      505        0      505      505        0      282       70
aobjpl           72    99777        0      843       26        0       26       26        0        8        2
bmsafemappl      64        0        0        0        0        0        0        0        0        8        0
bufpl           304    44539        0    12389     1156      202      954     1042        0        8        1
ccdbufpl        336        0        0        0        2        0        2        2        2       86        2
cryptodesc       72        0        0        0        0        0        0        0        0        8        0
cryptop         112        0        0        0        0        0        0        0        0        8        0
dino1pl         128     5468        0     4844      157        0      157      157        0        8        0
dino2pl         256        0        0        0        0        0        0        0        0        8        0
diraddpl         56        0        0        0        0        0        0        0        0        8        0
dirhash        1024      375        0      373       94        0       94       94        0      128        0
dirrempl         64        0        0        0        0        0        0        0        0        8        0
dma1024        1024        0        0        0        0        0        0        0        0        8        0
dma128          128        0        0        0        0        0        0        0        0        8        0
dma16            24        0        0        0        0        0        0        0        0        8        0
dma2048        2048        0        0        0        0        0        0        0        0        8        0
dma256          256        0        0        0        0        0        0        0        0        8        0
dma32            32        0        0        0        0        0        0        0        0        8        0
dma4096        4096        0        0        0        0        0        0        0        0        8        0
dma512          512        0        0        0        0        0        0        0        0        8        0
dma64            64        0        0        0        0        0        0        0        0        8        0
dma8192        8192        0        0        0        0        0        0        0        0        8        0
drmobjpl        200    99760        0      837       72        7       65       70        0        8        8
ext2dinopl      128        0        0        0        0        0        0        0        0        8        0
ext2inopl       232        0        0        0        0        0        0        0        0        8        0
extentpl         40    97224        0      894       15        0       15       15        0        8        1
fdescpl         440      571        0       64        8        0        8        8        0        8        0
ffsino          232     5468        0     4844      286        0      286      286        0        8        1
filepl          120    39245        0      260        9        0        9        9        0        8        0
freeblkspl      192        0        0        0        0        0        0        0        0        8        0
freefilepl       48        0        0        0        0        0        0        0        0        8        0
freefragpl       64        0        0        0        0        0        0        0        0        8        0
ifaddritemp      64       11        0       11        1        0        1        1        0        8        0
indirdeppl       56        0        0        0        0        0        0        0        0        8        0
inodedeppl      152        0        0        0        0        0        0        0        0        8        0
inpcbpl         352     3622        0       17       25       15       10       22        0        8        8
ipqepl           40        0        0        0        0        0        0        0        0        8        0
ipqpl            40        0        0        0        0        0        0        0        0        8        0
knotepl         104        2        0        2        1        0        1        1        0        8        0
kqueuepl        256        1        0        1        1        0        1        1        0        8        0
l2cap_pdu        72        0        0        0        0        0        0        0        0        8        0
l2cap_req        80        0        0        0        0        0        0        0        0        8        0
lockfpl          88     2808        0        9        1        0        1        1        0        8        0
```

Figure 1. */usr/bin/systat displaying output for the 'pool' view*

INUSE is the total number of items which have been allocated from the pool and are actively being used in the system. This value will decrease as items are returned to the pool. Also, this value, when multiplied by the size of the pool item yields the amount of memory (in bytes) which has been allocated from this pool and is actively being used. The third ordering option mentioned above, invoked using the `Z` command, orders the pools by this derived value (INUSE * SIZE). There is potential confusion on this point, as the `systat(1)` man page refers to this as an ordering by size, which could lead one to expect an ordering by values in the SIZE column.

Each pool item is associated with exactly one page of memory in the pool, and each page is kept on exactly one of three lists depending upon how many of its individual pool items have been allocated:

one list for *full* pages – pages which have allocated all their items and cannot satisfy any more requests; another list for *empty* pages, which have none of their items allocated; and the third list for *partially full* pages – pages which have some items allocated but still have more available to satisfy additional requests. As pool items are allocated or returned to the pool, these pages will be moved from one list to the other as their status changes from being either full, partially-full or empty. Pages which are 'empty' are also called *idle* pages and their count is displayed in the IDLE column of Figure 1.

There are three functions available to explicitly set upper and lower bounds on the resources (pool items or pages of memory) in a given pool. `pool_sethardlimit()` sets the maximum number of pool items which can be allocated at one time. This value (which is not displayed by systat) defaults to the maximum of an unsigned 32-bit integer.

The `pool_setlowat()` function sets the lower bound on the number of items to keep in the pool for further allocations. If an allocation will cause the number of remaining items in the pool to decrement below this low watermark, then the backend routine, `pool_catchup()` will be called before completing the allocation. `pool_catchup()` will request more pages of memory from the system; these pages will be used to replenish the number of available pool items.

Setting upper bounds establishes thresholds which define when memory should be returned to the system. If a spike in demand requires committing additional memory to a given pool, then this memory should be returned the system once demand has subsided and the memory is no longer needed. The `set_hiwat()` function sets the maximum number of idle pages (i.e., empty pages) to keep in the pool; this is the value in the MAXPG column of Figure 1. If, as items are returned to the pool, the number of idle pages exceeds the value set by this function, the idle pages will be taken out of the pool and returned to the system. By default, this value is set to 8. The value in the IDLE column should never exceed the value in the MAXPG column.

Both `pool_setlowat()` and `pool_prime()` will set the value in the column *MINPG*. This is the minimum number of pages to keep in the pool. The default value is zero.

There are two scenarios in which a request for a pool item can fail and increment the value in the FAIL column. If a request is made for an item and the hard limit has already been reached, the allocation routine checks whether the requestor set the `PR_WAITOK` flag, in which case the routine will wait until the number of pool items in use drops below the hard limit (i.e., an item is returned to the pool) before fulfilling the request. If the `PR_WAITOK` flag is not set, then the routine will log a message that the hard limit has been hit, increment a counter that tracks the number of failed requests and return immediately. The other case occurs if a pool is depleted and has no more items to give out when a request is received. If the `PR_WAITOK` flag is not set as part of the request and the attempt to allocate more pages of memory from the system fails, then the request will be unsuccessful and the number a failed requests will be incremented.

The values in the columns PGREQ and PGREL are simply running totals of the number of pages either requested or released back to the system by the pool backend routines. The HIWAT value is the highest number of pages ever allocated to the pool at one time since its initialization (N.B. – not the value set by `set_hiwat()`).

Some of the pools created by the kernel are used for internal operations. Others are for handling operations initiated from programs in the user space. Watching the statistics as they are refreshed, one can easily see the backend memory management activity at work, requesting more pages to meet the demand for more pool items. To demonstrate quite obviously, one can execute the `find(1)` command over a directory with many files (the `/usr` directory, for example). This causes a dramatic increase in demand for vnode structures as can be seen by the spike in the number of requests for memory from the vnode pool. As the items are removed from the pool, the backend pool allocator requests more pages of memory to keep up with the demand. This accordingly increments the number of pages requested (PGREQ) and pages in use (NPAGE).

Conclusion

Monitoring kernel pool memory usage provides a glimpse into the activities of the running kernel. Pools provide an efficient method for dynamic memory allocation and are used for memory structures which are often allocated and are of odd sizes. Using pools can reduce memory fragmentation which might otherwise occur if memory were allocated using the general purpose allocator. If required, pools can be used to guarantee that parts of the kernel will continue to have memory available for their operation regardless of other memory constraints on the system. The systat utility in OpenBSD (and the vmstat utilityin NetBSD) return statistics about activity in the pool subsystem.

PAUL MCMATH

Paul McMath has worked as a Unix admin for 10+ years in Europe and the United States. He has been using one BSD variant or another as his OS of choice since 2002.

FreeBSD 8.2 Against Ubuntu Server 11.10

An Objective Comparison of Two Power House Open Source Server Platforms, BSD Unix and Linux.

FreeBSD is the dominate title holder in the BSD based OS market, but how does it compare to an up and coming challenger if that perspective of Open Source server platforms is broadened? We'll take a wide and ranging look at the strengths and weaknesses of each platform and let the reader decide. We'll look specifically at Canonical's latest release of Ubuntu Server, Oneric Ocelot, 11.10, and the latest release of FreeBSD, 8.2.

Installation Media

FreeBSD is primarily distributed on CD iso's either as a full size distribution weighing in close to 700 megabytes, or also as a lightweight net-install if you have a good bandwidth to the Internet or local ftp source. Options are also available for USB thumb-drive installs as well. Note, internet connectivity is not generally needed for standard CD distributions, either the 32 bit or 64 bit versions.

Ubuntu Server is distributed as well in CD iso format, as well as a distribution that can be used on a thumb-drive. Both 32 bit and 64 bit versions are available as well. Internet connectivity is highly recommended as the installer downloads the latest packages from the repository during the install.

The Install Process

FreeBSD's installer hasn't changed much in the 15 years that I've used it, since early 4.6 days... You have the quick, and expert modes, of course, and the most flexibility of disk layouts. The 8.x releases give more choices of filesystem types, including ZFS. One thing I have found annoying is the default sizes of /var are in most instances much too small for most configurations.

Although the FreeBSD appears to be a bit archaic, it is worth pointing out that FreeBSD's simple CUI interface allows the OS to be installed on devices that don't have a graphical display output, and only have a serial console. Examples would be the Alix SBC, and other server platforms such as DEC Alpha's and DecStations.

The typical steps of creating the partition table, allocating slices, filesystems, follow and are pretty straight forward. Once the initial install is through, a myriad of other steps are necessary, setting the root password, setting up network interfaces, timezone, enabling ssh, etc, all of which are post-installation.

Ubuntu Server tends to focus primarily on the PC enterprise server market though they have recently been some interest in some of the embedded devices markets. As such, Ubuntu tends to explore the graphics modes more freely, switching between various graphics and text modes throughout the install process. It is possible to install Ubuntu with a serial console, but it is by no means as straight forward as FreeBSD is, involving changing multiple system files, boot-loaders, etc.

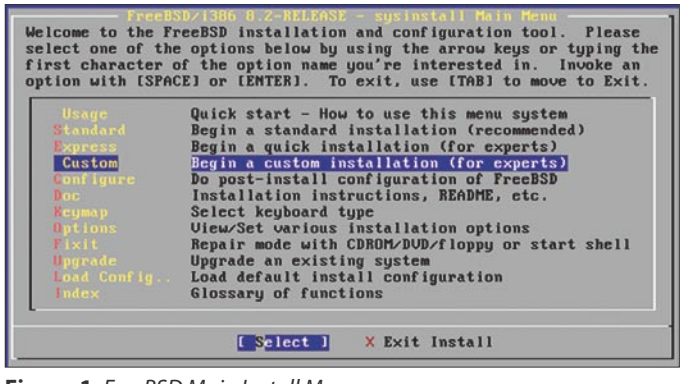


Figure 1. *FreeBSD Main Install Menu*

Ubuntu's boot screen launches the user into an menu that allow a prompted install, and gives the user the chance to choose a guided disk layout or manual if preferred.

Ubuntu doesn't by default enable the *root* account, and in fact, requires the creation of a user account during install that will be in the *sudo* group. That information is collected after the disk layout is chosen, and the user account is created during the install. Networking, timezone setup are done during the initial installation process.

One handy feature of the Ubuntu Installer is the Software Selection Box, or *tasksel* as it is called from the command line. It lets you select several of the most common packages or configurations from one selection box.

Typically, you might select to install SSH, or a full LAMP (Linux, Apache, Mysql, and PHP) stack. This is quite handy, as the installer not only install and configures Mysql, but let's you set the mysql *root* password from dialog boxes. No extra steps are necessary, and your up and running with a LAMP server. One note, Ubuntu is still shipping Mysql 5.1 from it's repositories, rather than the more commonly available 5.5. Some differences each of the Installers:

FreeBSD doesn't

- require or expect Internet access
- initially configure network interface (post-install)
- include a bash shell by default (available as a package)
- require a video display adapter

Figure 2. *Ubuntu Server Main Install Menu*

Ubuntu Server doesn't

- support kernel ZFS (Sun Zettabyte File System), but does offer BTRFS
- limit RAM to 4 gig on 32bit hardware (default PAE kernel included)
- use the Linux 2.6 kernel anymore (as of 11.10)
- install software development tools by default (build-essential)

Neither include a GUI by default. These are server operating systems after all.

Package Management

Both Operating Systems support remote package management and installation. They differ greatly and functionality.

FreeBSD has two primary methods for installing applications, Ports and packages

Ports is an organized directory structure of packages, comprised of pointers to remote locations of each package's remote location and dependencies. Executing a *make* in a given package directory will retrieve not only that packages source, but all it's dependencies and compile them as necessary to complete the applications build from source.

eg:

```
$ cd /usr/ports/shells
# [/usr/ports/shells]$ cd bash
# [/usr/ports/shells/bash]$ make; make install
                //compiles and installs
```

Packages are compiled and complete compressed installation binary bundles of a given application and can be installed without compiling from source. An example would be to install the Bash shell after an initial install.

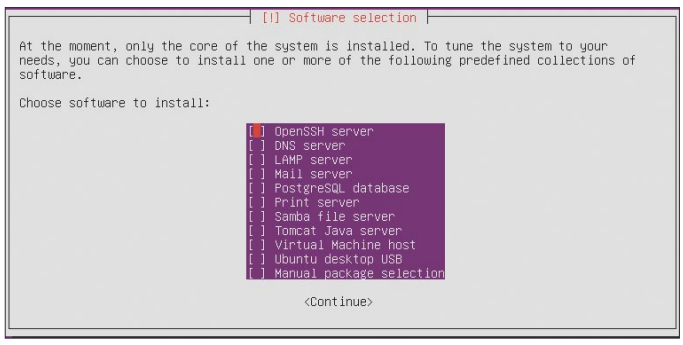


Figure 3. *Software Selectio*

This would be easily done in Packages by typing:

```
# pkg_add -r bash              // This will retrieve and
                  install the binaries.
```

The FreeBSD package management system is both wonderful and a cause of frustration. It's design is limited it's handling of dependencies of local packages. This can be a real irritation if you are installing a local package that has many dependencies. The dependencies have to be installed manually via `pkg_add -r some_pkg`, before the local package will install. This is inherent in Pkg's original design for the packages to be hosted by FreeBSD's main package sites.

Ubuntu Server Uses The Package System That It's Parent Os, Debian Uses, Called Advance Packaging Tool, or APT

It can also handle Redhat packages, using RPM, but APT is the primary method. APT uses a local cache of remote repositories for tracking and updating both system and other package databases.

Once the server is up and on the network, executing the following commands will update the database caches locally, and allow searching for packages.

```
$ sudo apt-get update      // Update caches from
                  repositories
$ sudo apt-get upgrade     // Upgrade installed packages
$ sudo apt-cache search python   // search for
                  available package called python
```

Additional packages can be added to the servers list of sites, simply by the repository to the list of apt sources. A very flexible system and works well.

Configuration Files

FreeBSD uses a single system config file, `/etc/rc.conf` which is read by individual startup files in `/etc/rc.d` and it's child directories... Network interface configuration, enabling the startup and options to various options are accomplished in `/etc/rc.conf`.

Ubuntu Server uses `/etc/network/interface` for the primary interface configurations and `/etc/init.d` for startup of various system and other applications. Both OS's use `/var/run` and `/var/log` for obvious reasons.

Performance, How Do They Stack Up?

This is one subjective issue, that has as many questions as answers... Ubuntu Server is the most similar to FreeBSD, as far as the variants of Linux's go. Each performs well, in a variety of high load areas. I use both systems in similar load conditions, and each has it's own distinct advantages.

FreeBSD's ZFS filesystem, has proven to outperform the Linux EXT4 filesystem on the same hardware, under high i/o load. Note, ZFS requires 64bit hardware and a minimum of 4 gigs of memory for successful operation.

FreeBSD also seems to have an edge in network i/o under similar high load udp network load. If ZFS filesystems are not used, then Ubuntu Server performs much better with EXT4 against FreeBSD's aging UFS2 filesystem. Ubuntu seems to better utilize the cache buffer than UFS does.

Ubuntu's has a pretty nice implementation of Logical Volume manager, which is tightly integrated in the the installer, and can be enabled during the partitioning phase of installation. It's well designed, if software raid is desired.

The FreeBSD kernel has other performance advantages over the typical Linux kernel, enough so that Debian released it's distro with a FreeBSD kernel, called KFreeBSD.

Ubuntu Server does virtualization very well, if you have the hardware for it. The latest release 11.10 offers Xen, and KVM. Other layer 2 hypervisors work just as well, like VirtualBox, which is my favorite. The VM Host in my office is running Ubuntu server with 2 FreeBSD VM's and another Ubuntu Server VM's all happy in 4 gigs of ram.

FreeBSD and Ubuntu Linux, though from completely different lineages, share many common traits, and can work well together.

On the 'Net

- Ubuntu Server – *http://www.ubuntu.com/business/server/overview*
- Debian KFreeBSD – *http://wiki.debian.org/Debian_GNU/kFreeBSD*

BILL HARRIS

Bill Harris has 27 years experience in programming and Unix/Linux Administration. His work experience includes data center environments as well as education IT. He hold a General Class Amateur Radio license, and enjoys working with the processing of realtime weather data.

EuroBSDcon 2011 From Organizers Perspective

Although the idea of the EuroBSDcon 2011 was born during the EuroBSDcon 2009 in Cambridge, the preparations for the event actually started in 2010 in Karlsruhe, where some Dutch people offered to organize it.

The announcement was made that the 10th *EuroBSDcon* would take place in the Netherlands from the 6th to the 9th of October 2011. Announcing exact dates without having done proper research was not a smart move as we learned along the way...

Organizing a conference is simple. You just need a venue, a website, some hotel arrangements, speakers, a registration system, a lot of visitors, some money and you're done. Right?

As it turned out it was not that easy. Finding the right location was the first challenge. Some of the places we looked into were not available during the conference dates. Like Eindhoven for example, which was going to be closed on the 9th because of a marathon. This limited our choices. Apparently we should not have announced a date up front before doing our research. Fortunately due to RedHat (yes, Linux) we stumbled upon Meeting Plaza in Maarssen. One of their big advantages is their *pay per visit* model, which makes conferences more easily scalable. Instead of paying per room, a bit extra for using beamers and AV, some more extra for Wifi internet, and even more extra for coffee, tea, lunch, etc. Meeting Plaza just charges a fixed price per person which includes everything what is needed.

Meeting Plaza's scalability was exactly what we were looking for. Despite having a lot of sponsors from the beginning, our calculations showed that it would be nearly impossible to invite all the speakers that we wanted. By the end of August we were determined to make every penny count. Then, unexpectedly two other companies contacted us and offered their support. We gained two additional platinum sponsors this way, what gave us the financial room we needed to make the conference a real 10th anniversary and to invite more speakers.

Our last challenge was to attract enough visitors. For that we needed a registration system online by the middle of June. Unfortunately, it went live in the mid of August. How did that happen? Well, registration systems are more complex than we expected. They typically consist of a website and a payment module. The payment module communicates with a payment provider which talks with a bank. We faced difficulties with procedures and had contract issues. We were even rejected by one bank, because we were in the Netherlands...

Hotel arrangements were also a challenge. We learned that there was an electronics fair in Utrecht that same weekend, making it difficult to find affordable accommodations. As I mentioned before, we should have done some research before announcing the date of the 10th EuroBSDcon.

Finally, after months of work and planning, Thursday the 6th of October 2011 came. At 8:00 we started to get everything ready in order to open the registration desk at 9:00. Around 8:30 the first enthusiastic visitors showed up. We quickly improvised a sign indicating that registration would open at 9:00 and that breakfast was served on the first floor. At 9 o'clock we opened up the registration and everything went quite well, until a few minutes passed 10 o'clock.

A few minutes after the tutorials started, the first complaints arrived. There were not enough IP addresses in the DHCP range. The internet came with the venue, so we contacted them and they started to call their network provider, to get the problem solved. Unfortunately it could not be fixed until the next day.

On Thursday we had two tutorials, one by Kirk McKusick which is always interesting if you want to know more about the design behind FreeBSD. The other tutorial was the most visited tutorial of the conference: *pfSense 2.0*

by Chris Buechler and Ermal Luçi. This tutorial covered many of the features and changes in the 2.0 release from the perspective of new and existing users. Unfortunately with so much information to cover, the tutorial went well beyond the allocated time and we had to cut it short because the room was booked for another event in the evening.

On Friday Kirk continued with the second part of his *An Introduction to the FreeBSD Open-Source Operating System* tutorial. The next tutorial track featured Peter Hansteen, who gave his tutorial on PF, which provided updates on the new PF syntax and features introduced since OpenBSD 4.7. If you work or want to work with PF this is a tutorial you should have attended. The third tutorial track on Friday consisted of two half day tutorials. The first being a Dtrace tutorial by Tod McQuillin. Some of the participants of the conference told me that they saw great benefit from having Dtrace available for them. In the afternoon Tod's tutorial was followed by a NETGRAPH tutorial by Adrian Steinmann, which gave great insights in what NETGRAPH can do. On Saturday, the conference really started.

Hans van der Looy had the honor of giving the first keynote of the conference or we had the honor of having Hans as a keynote speaker, depending on how you look at it. Hans took the audience on a tour through the DigiNotar

incident and explored some alternatives for the trust issue online. This resulted in a discussion between Hans and Henning Brauer about viable alternative protocols. The takeaway message was that it is hard to establish real trust online when everybody lies (sometimes).

The keynote was followed by talks about IPv6 on FreeBSD, Open vSwitch, Testing NetBSD automagically and the 10th anniversary of PF.

During the lunch break one of our sponsors treated us to a dutch specialty (*poffertjes* – kind of small pancakes) which were served on the ground floor, close to the entrance. Having all previous lunches on the first floor, a lot of people stayed there. So it was my job to walk to the first floor to point out the *poffertjes* to the visitors. On my way to the lunch room I ran into some colleagues, who asked if we had everything under control. Optimistically, I responded with *yes*, which proved to be a bad idea. Pascale, one of the crew members, touched me on the back and uttered: *We do have a problem*. At the same time the smell of burned plastic reached my nostrils. Time to turn around and take a look. As it turned out a power bar had burned through due to the difference in voltage in Europe and the USA. Anyway, this smell of molten plastic somehow solved the problem by driving our visitors towards the *poffertjes* on the ground.

After the lunch break we had to put one of our contingency plans in place when we had a room overflow for the NPF talk. Redirecting everyone to the plenary session room solved this problem easily. The talk itself presented some really great ideas on handling traffic and processing the firewall rule set. Besides the NPF talk the other talks after the lunch break where about Webcamd, nginx on FreeBSD and OpenBSD's new suspend and resume framework. After the tea break another talk about PBI for FreeBSD and PC-BSD 9. For the people interested in virtualization the *Virtualization under *BSD: the case of Xen* talk provided insight into the Xen hypervisor.

The first conference day finished with the always entertaining *History of BSD* by Kirk McKusick, after which everyone was directed to the buses to get to the social event at the Dutch Railway Museum in Utrecht. Everyone got a tour around the Museum, in Dutch or English. In hindsight we should also have added a tour in German, but all the ingredients for a social event were present: interesting location, good food and the ability to socialize. We got a lot of positive feedback about part of conference. However it should be noted that it was organized by one of our sponsors so we can't claim the accolades. Without Anja's and Mona's experience we would not have been able to make the social event this good. So we are very happy they organized it for us.

On Sunday we started with our second keynote by Herbert Bos called: *The eight-fold path to reliable operating systems*. A very entertaining and interesting keynote about the design choices made in Minix 3. The most important one being that everything should keep working despite parts of the system crashing and restarting. Hopefully this has provoked some thoughts and ideas to make the BSD's better.

The first slot after the break featured a capsicum talk by Robert Watson. Capsicum being a practical approach to fine grained access control. Another track in this slot was ideal for who wanted to learn more about Minix after the keynote. By following the *Beastie Meets Raccoon: MINIX 3 as a BSD* talk we could learn more about it very quickly.

I think the Sunday program created a lot of hard choices for the visitors. Multiplicity, FreeNAS or learn more out how NetBSD started to use the fossil version management system? Learning more about HAST and run the risk of missing out the latest developments in OpenSSH? Get introduced to ZFS from a system management point of view or get new insights of how to protect your data nowadays? And if you think these are difficult choices, remember that by following these you would have missed the history of sendmail, the obsolescence of the OS, insights into BSDcertficiation and the porting of OpenBSD to Sun ultrasparc processors.

We are looking back at the EuroBSDcon 2011 as a successful, entertaining and interesting conference, which would not have been possible without the sponsors and of course all speakers and visitors. It is very rewarding to see the enthusiasm and hear the positive feedback of all who visited Maarssen. With visitors from 27 different countries it is safe to call the EuroBSDcon an international conference.

At the moment of writing it is still unclear where and when the EuroBSDcon will be held next year. Hopefully it will be known very soon. In the meantime we are busy trying to get the EuroBSDcon Foundation up and running. The target of the EuroBSDcon Foundation will be to provide support to the organizers of, share knowledge about and transfer resources to future EuroBSDcon conferences. For example in helping with or providing the registration system.

JEROEN VAN NIEUWENHUIZEN

Jeroen van Nieuwenhuizen was the chair of the EuroBSDcon 2011 organizing committee. In his daily life Jeroen works as a Unix Consultant for Snow B.V. He came in contact with Unix in 1997 and started to work with the BSD's in 2002. His free time activities include reading, recumbent cycling, speed skating and herding his cats.

The Passing of Steve Jobs: Not Just Apple's Loss

I think it's only appropriate that this magazine comment on the passing of Apple's most charismatic co-founder. When I read Apple's simple but tasteful words of memorial, I knew what I wanted to say. From http://apple.com/stevejobs:

> *Apple has lost a visionary and creative genius, and the world has lost an amazing human being. Those of us who have been fortunate enough to know and work with Steve have lost a dear friend and an inspiring mentor. Steve leaves behind a company that only he could have built, and his spirit will forever be the foundation of Apple.*

Contrary to his company's tribute to his life, the passing of Jobs is a loss to more than just Apple, his friends, and his family. Whether you love or hate Apple and Jobs, nobody can deny that his contributions to technology reach far beyond his own company's products. Here is an excerpt from a February 26, 2004 post on the blog of outspoken FreeBSD community member, Grant Hayes (aka "Trollaxor"):

> *... only after Apple started modifying FreeBSD 4.x and submitting their modifications did FreeBSD progress to the 5.x branch. The advanced VM and SMP code that allows Mac OS X to run so efficiently is the very same code that finally put FreeBSD on the level with Linux. I run FreeBSD 5.2 on a four-way Xeon box at work and thank Apple every day. If it weren't for the Mach micokernel from Apple we wouldn't be able to do these nice things with FreeBSD now or probably ever.*

The work and genius that Jobs put into Apple (especially after returning from NeXT) has had impacts on several major players. Android wouldn't be where it is today if it weren't for iOS. Even Microsoft had to follow Apple's lead in order to create a Windows Mobile device that can compete. Mark Shuttleworth openly admits that OS X is the inspiration for his Ubuntu designs.

Jobs has affected so much more than just what Apple does. His vision sent ripples of positive innovation throughout the computing industry at large. The BSD community ought to be thankful for the life and work of Jobs every day.

Rest in peace, Steve.

By Toby Richards

In the next issue:

- BSD Kernel Compiling
- nsswitch, nscd and caching
- OSPFv6
- and Other !

Next issue is coming in December!

If you wish to contribute to BSD magazine, share your knowledge and skills with other BSD users – do not hesitate – read the guidelines on our website and email us your idea for an article.

Join our team!

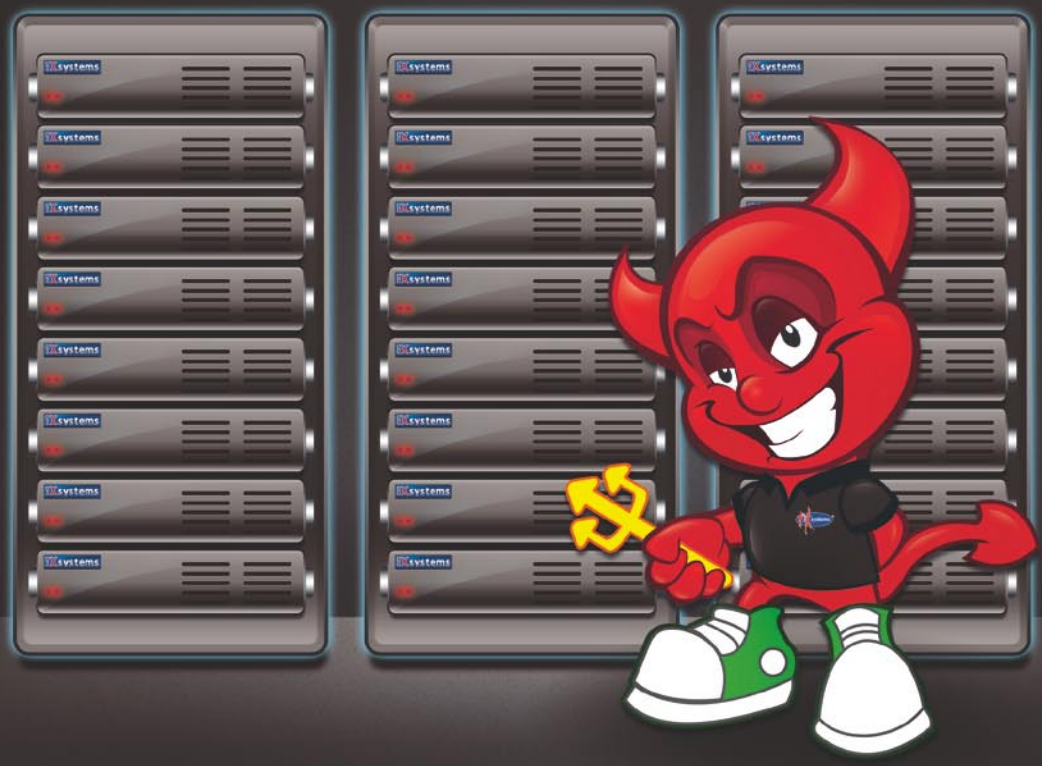
Become BSD magazine Author or Betatester

As a betatester you can decide on the contents and the form of our quarterly.

It can be you who read the articles before everybody else and suggest the changes to the author.

Contact us:
editors@bsdmag.org
www.bsdmag.org

What has your server vendor done for BSD lately? Probably, not much.



Work with a vendor that supports the operating system you love!

iX is the corporate sponsor of the PC-BSD® Project, a major corporate donor to the FreeBSD Foundation, and leads the FreeNAS™ development team -- all while employing some of the most brilliant minds in the FreeBSD® community. For BSD hardware and software expertise, look no further.

1-855-GREP-4-IX
http://www.iXsystems.com/community

iXsystems®