FREE DVD INSIDE LATEST RELEASE - FREEBSD 7.0

BSD MAGAZINE

Vol.1 No.1 Price USD14.99 AUD14.99 Issue 1/2008(1) ISSN 1898-9144

FreeBSD Ins & Outs

Install & Configure step-by-step

EXCLUSIVELY

- SoftIntegration C++ Graphical Library
- Ch Standard 6.0
- pfSense 1.2
- Blender for FreeBSD

PLUS

FreeBSD's bsnmp
Squil 0.7.0 on FreeBSD 7.0
How-To Dual-Boot Vista with BSD
Waste Spammers' Time
Defense in Depth and FOSS
NetBSD on the NSLU2

Many useful applications and how-tos

FREEBSD-BASED OPERATING SYSTEM RE-INVENTS THE DESKTOP

PCBSD

1.5 Edison Edition

Open Source for Open Minds

A shift to open source is happening as we speak! Microsoft Vista is sure to be the last operating system of its kind. Finally there is a user-friendly version of FreeBSD that can penetrate the PC market and effectively compete with Microsoft Windows. Introduce PC-BSD, a stable and secure operating system with a smooth and seamless user experience.

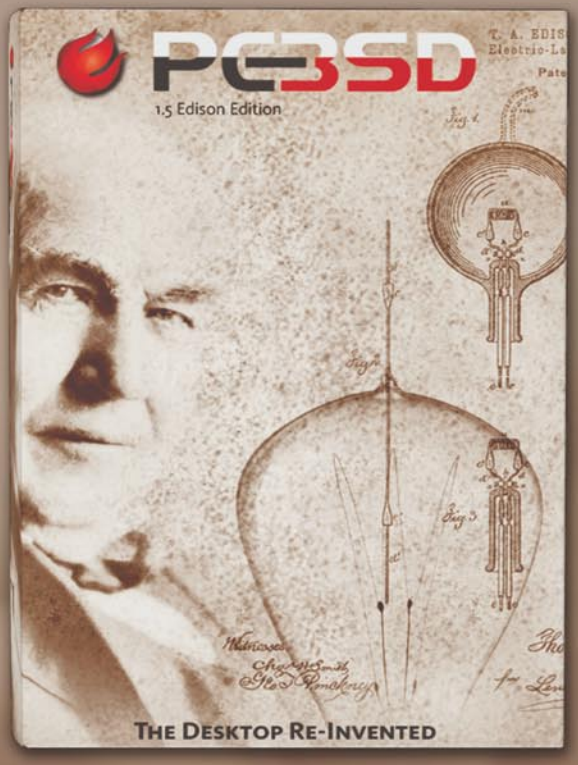
PC-BSD Version 1.5 Edison Edition features everything current users love about 1.4, including a fully functional desktop operating system running FreeBSD 6-Stable under the hood, a graphical system installer to make the system installation process effortless, safety from viruses and spyware that plague other operating systems, and self-installing software packages.

New features introduced in the Edison Edition include a 64-bit release version + Live CD, a new system-update manager, Xorg 7.3, KDE 3.5.8, and FreeBSD 6.3 running under the hood.

Join the movement and download PC-BSD Version 1.5 today from http://www.pcbsd.org. Better yet, buy the box set from http://www.freebsdmall.com. Either way you'll be installing the state-of-the-art, stable and secure open source operating system that is revolutionizing the desktop market.

iXsystems
Enterprise Servers for Open Source

POWERED BY FreeBSD



T. A. EDISON.
Electric-Lamp.

No. 223,898. Patented Jan. 27, 1880.

Inventor
Thomas A. Edison
for Lemuel W. Serrell
atty

Fig. 2.

Fig. 3.

Witnesses
Chas. H. Smith
Geo. D. Pinckney

MAGAZINE BSD

Come Play on the BSD Team!

Dear Readers,
Hi, my name is Karolina and I would like to welcome you to BSD magazine! This magazine was launched for you, so please contact us with all your comments, ideas and suggestions. Only with your help, will we become the very best information resource for BSD systems, applications, multimedia solutions, administration and security. You can even send in your complaints. I hope there will be none! But, what I really want right now is for you to join our team!

My special thanks go to Dru Lavigne for her wonderful assistance and to all those who gave us a hand when we needed it most. If I thanked everyone by name, it would take several pages, so please forgive me and know that I am thankful for your help.

In this premiere issue we cover many interesting topics. In the networking arena, Michael W. Lucas presents how SNMP provides information to a network management, how to configure bsnmpd on FreeBSD and finally, how to enable and disable different snmp functionality. For desktop fans, Jan Stedehouder focuses attention on two open source desktops, which are firmly rooted in FreeBSD: PCBSD and DesktopBSD.

On the practice court, Richard Bejtlich demonstrates installation of Sguil on FreeBSD and Jay Kruizenga shows step-by-step instructions on preparing your Vista hard drive to accept BSD. In the safety zone, Peter N.M. Hansteen shows how to easily deal with spam and have a good time doing it.

Henrik Lund Kramshoj is in center court to introduce Defense in Depth – which can help lessen the security burden while growing your infrastructure. Getting into deeper tactics, Donald T. Hayford teaches how to boot your NSLU2 into a full version of NetBSD and Girish Venkatachalam takes a closer look at OpenBSD pf.

Getting into the strategy of the game, Federico Biancuzzi interviews FreeBSD developer – Jeff Roberson and Mikel King shares his thoughts about the need for BSD Certification
I hope you will find our articles interesting and will look forward to the next issue!

Karolina Lesińska
Executive Editor

BSD Magazine is published by Software Media LLC
on Software Wydawnictwo Publishing License
(*www.software.com.pl/en*)

Editor in Chief: Ewa Dudzic *ewa.dudzic@bsdmag.org*
Executive Editor: Karolina Lesińska *karolina.lesinska@bsdmag.org*
Editor Assistant: Katarzyna Kaczor *katarzyna.kaczor@bsdmag.org*

Director: Ewa Dudzic *ewa.dudzic@bsdmag.org*

Art Director: Agnieszka Marchocka
DTP Manager: Robert Zadrożny
Prepress technician: Ireneusz Pogroszewski

Contributing: Michael W. Lucas, Jan Stedehouder, Svetoslav P.Chukov, Richard Bejtlich, Jay Kruizenga, Peter N.M.Hansteen, Henrik Lund Kramshøj, Donald T.Hayford, Girish Venkatachalam, Eric Schnoebelen, Federico Biancuzzi, Mikel King

Special Thanks to: FreeBSD Foundation

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

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

Production Director: Marta Kurpiewska
Prepress technician: Robert Zadrożny

Postal addres:
Software Media LLC
1461 A First Avenue, # 360
New York, NY 10021-2209
USA
Tel: 1 917 338 3631
www.bsdmag.org

Software-Wydawnictwo Sp zo.o. is looking for partners from all over the World. If you are interested in cooperating with us, please contact us by e-mail: *editors@bsdmag.org*
Print: 101 Studio, Printed in Poland

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

The editors use automatic DTP system AUPUS

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

Subscription orders can be sent to *subscription@software.com.pl*
Customer Service 1 917 338 3631

BIND 9.5's new features

BIND, which originally stood for Berkeley Internet Name Daemon, is a suite of DNS (domain name system) software that provides a DNS server, DNS resolver library, and various DNS-related tools.

BIND dates back to the early 1980's where it was designed to serve the needs of distributed computing communities and to be compatible with the naming service planned for the DARPA Internet. Since the mid-1990's, BIND has been maintained and developed by Internet Systems Consortium (ISC) which has become well-known for their support for many open source projects, funding and development of BIND and other open source software, and design and advocacy of many Internet standards. BIND was rewritten and version 9 was released in September 2000.

According to the Infoblox 2007 DNS Survey, 70% of the Internet's estimated 11.7 million name servers ran BIND. (Microsoft's DNS Server ran on 2.7%.) BIND is provided in the default installations of NetBSD, FreeBSD, OpenBSD, and DragonFly operating systems. Plus, it is the frequently recommended DNS server used on most Linux distributions and various Unix flavours.

For over a year, ISC has been developing and testing many new features for BIND 9.5. This article will quickly summarize some of the significant new features:

- GSS-TSIG support
- DHCID
- Statistics support for named via XML
- UDP Socket Pool
- Handling EDNS timeouts
- O(1) ACL processing

GSS-TSIG, or the Generic Security Service Algorithm for Secret Key Transaction Authentication for DNS, is documented in RFC 3645. It is an update for Secret Key Transaction Authentication.

GSS-TSIG is the authentication mechanism of choice for DNS dynamic update in Microsoft Active Directory.

It is potentially useful for other things, said Rob Austein of ISC, but the big push for BIND 9.5 was to allow named (the BIND DNS server) to act as the DNS server for an Active Directory zone.

GSS-TSIG is a composite of GSSAPI and TSIG – a wrapper layer built on top of a wrapper layer. It is insanely general, said Austein, but the common usage is DNS wrapping TSIG wrapping GSSAPI wrapping SPNEGO wrapping Kerberos 5 – thus for practical purposes it is a mechanism for using Kerberos 5 to authenticate DNS.

BIND added the new DHCID Resource Record (RR) type to keep up with standards. The DHCID RR is used for encoding DHCP information and DHCP servers and clients use it to identify DHCP clients with a DNS name with a strategy of reducing conflicts in the use of fully-qualified domain names. The data is a one-way SHA-256 hash computation. More details are in RFCs 4701 and 4703.

BIND 9.5 adds an experimental HTTP server and statistics support for the DNS server via XML. It is not a web-based configuration interface, but a statistics feed that happens to use the HTTP protocol for delivery because it is flexible and very well-supported, said Evan Hunt of ISC.

Also BIND 9.5 makes it a bit harder to play games with insecure DNS by brute force attack on the 16-bit DNS ID space, said Austein. The server provides a pool of UDP sockets for queries to be made over, for example, using eight ports instead of one in effect adds three more bits to the search space.

BIND 9.5 makes fallback to plain DNS from EDNS due to timeouts more visible. EDNS (Extension Mechanisms for DNS) have been available for around eight years and many servers (and all root servers) support it.

The problem is that some firewalls do not support EDNS by default, said Mark Andrews of ISC. Also there are some authoritative servers that fail to respond when they see a EDNS query rather than return an error code as is required, said Andrews. Timeouts may mean network problems, dead servers, broken middle boxes, and broken authoritative servers.

Falling back to plain DNS will help with the later, said Andrews, but has a negative impact on DNSSEC (which requires EDNS) especially when there are overloaded links causing packet loss.

On timeouts, named retries EDNS with a 512 octet UDP size (which usually allows EDNS to get through a firewall as it is generally not fragmented and is within the sizes allowed by plain DNS) and then tries plain DNS if still needed. The server logs this to draw attention to the issue and to get any non-RFC compliant boxes replaced or re-configured, said Andrews.

Andrews said at some point soon, BIND will not fallback from EDNS to DNS on timeout. He suggests the following for BIND administrators for EDNS:

- Firewalls and NAT boxes need to handle fragmented responses both in and out of order.
- Firewalls need to handle EDNS responses.
- Broken authoritative servers need to be replaced or upgraded which first means they need to be identified.

Also BIND 9.5 introduces a new ACL-processing engine. Instead of storing ACLs (i.e., allow-query, allow-recursion, et cetera) as linear lists that have to be searched every time a query comes in, they are now modified radix trees. There should not be any change in the way things are configured, said Evan Hunt of ISC, but sites with ACLs containing more than one or two addresses should hopefully see an uptick in queries per second.

More details about BIND 9.5 features can be found in the BIND Administrator Reference Manual and manual pages.

by *Jeremy C. Reed*

The KDE Development Platform

In January, The KDE team released the latest version of the popular open source desktop for BSD, Linux, and other operating systems. This KDE 4 release brings more to the table than a refreshed and beautiful desktop, dynamic support for portable devices, new sound system and communication framework. One of the most noticeable changes with KDE4 is under the hood, which makes it much easier to develop and use your favorite applications.

Earlier versions of KDE had one goal in mind: making a consistent, nice looking free desktop-environment usable for people in general(*www.kde.org/announcements/announcement.php*). Some will argue that the goal was reached with KDE version 2.2. The final proof was when KDE 3 got the ISO 9241 usability standard approval in May 2007 as an effective, efficient and satisfactory working environment. But the KDE project has not stopped there, reaching for an even more ambitious goal for the next major version, provide a developer platform.

As KDE 4 is based on Qt 4, a cross-platform C++ toolkit, it is able to incorporate a lot of the great functionality provided in Qt. This makes KDE even more interesting as a development platform. Earlier KDE releases had unique network capabilities. You could easily edit files remotely, and save them over the net. Developers integrated a NX client(*http://developer.kde.org/summerofcode/knx.html*) for remote desktops. Hardware detection was improving with D-bus making it easy to connect digital cameras and USB sticks. But when your network connection was lost, KMail gave annoying notification dialog. When connecting USB based headphones, you needed to do manual configuration to get it working. On the one side things seems to work. On the other, it is not really automatic. With KDE 4 it has changed. If you are using your soundcard, but are connecting a USB headset, the system switches sound output to it. The headset just works. When using your laptop with wireless, KMail automatically downloads your mail. When the network is lost or closed down, KMail recognizes that and stops the fetching of e-mails online. KDE 4 includes several new pillars, including the aforementioned Solid. The others include:

Plasma

Plasma is what makes the KDE 4 desktop customizable and useful. By combining Kicker, KDestkop and SuperKaramba, developers can make cool and beautiful applications easily, with all kinds of shapes, colors and rich functionality.

Phonon

With the Phonon multimedia interface, KDE is now media framework independent. Phonon makes it easier for developers to incorporate simple media into applications, leaving choice of the media framework to the user.

Decibel

The new (*http://en.wikipedia.org/wiki/Decibel_(KDE)*) communication framework, Decibel, integrates all communication protocols into the desktop. As of now we got all contacts in different applications: AOL, MSN, E-mail, Skype, etc. With Decibel all contacts are put on one place. If Person A wants to chat with Person B, Person A requests a connection, and the service manager choose the best available tool to communicate with Person B. Decibel will most likely be included in later KDE4 releases.

Sonnet

Sonnet is a multilingual spell check program with automatic language detection. A language can be identified with as little as 20 characters of text. Sonnet's design is more simple, as there is just one component now, compared with the 7 components in kspell2, which was used in KDE3. Sonnet now has better performance and improvements when checking languages such as Thai and Japanese.

Strigi

A pillar which provides desktop search on KDE4. Together with NEPOMUK it also introduces a semantic desktop search for KDE. NEPOMUK will allow user store add metadata, which Strigi will be able to index for more prices search.

Kross

A new scripting framework with bindings for Python and Ruby as a part of the development Platform. In addition there are efforts to also provide C# support.

Akonadi

This is a cross-desktop storage service for PIM data. It makes it easier to share PIM data across different applications as Kontact, KOffice or Evolution. It can be integrated with other groupware servers like Scalix, Kolab, GroupWise or OpenXchange. Most likely Akonadi will be introduced with KDE 4.1 or 4.2. Since the release of Qt 4, Trolltech has made Qt for Windows and Mac OS X available under a dual licensing model as well, with GPL version for free software development. This makes it easier to support platforms other than X11 based systems such as FreeBSD and Linux. Several workgroups are now working on making KDE 4 applications available on those additional operating systems. This will help the spread of free software, not only on open source system, but more proprietary ones too. It is more than 3000 applications made for KDE3. Now those applications can be ported to Mac and Windows by using Qt 4 and the KDE 4 development platform. I'll focus on universal design (*http://en.wikipedia.org/wiki/Universal_design*) most commonly known as Accessibility support. With support for accessibility, people with disabilities (such as being blind or deaf) can more easily use a computer. A small but productive group(*http://accessibility.kde.org/developer/kttsd/index.php)* are now working to include better accessibility support on KDE, making it ready for people with disabilities.

This group has done considerable work with D-bus binding to Qt 4, supporting the IAccesible2 standard. There are made some sample applications. QDasher(*http://labs.trolltech.com/page/Projects/Accessibility/QDasher*) is one, a computer accessibility tool enabling users to enter text efficiently using a pointing device rather than a keyboard. IAPoker(*http://labs.trolltech.com/page/Projects/Accessibility/IAPoke*) is an other, an accessibility information tool helping developers to verify that accessibility information is correct. In conclusion, you can get The KDE Development Platform from KDEs FreeBSD team(*http://freebsd.kde.org/*) And you can begin to explore some of the great new development functionality that has been included in this release.

by *Knut Yrvin*

FREEBSD 7.0

This is the first release from the 7-STABLE branch which introduces many new features along with many improvements to functionality present in the earlier branches. Some of the highlights:

- Dramatic improvements in performance and SMP scalability.
- The ULE scheduler is vastly improved, providing improved performance and interactive response
- Experimental support for Sun's ZFS filesystem.
- gjournal can be used to set up journaled filesystems, gvirstor can be used as a virtualized storage provider.
- Read-only support for the XFS filesystem.
- The unionfs filesystem has been fixed.
- iSCSI initiator.
- TSO and LRO support for some network drivers.
- Experimental SCTP (Stream Control Transmission Protocol) support (FreeBSD's being the reference implementation).
- Much improved wireless (802.11) support.
- JIT compilation to turn BPF into native code, improving packet capture performance.
- Much improved support for embedded system development for boards based on the ARM architecture.
- jemalloc, a new and highly scalable user-level memory allocator.
- freebsd-update(8) provides upgrades to new releases in addition to security fixes and errata patches.
- X.Org 7.3, KDE 3.5.8, GNOME 2.20.2.
- GNU C compiler 4.2.1.
- BIND 9.4.2.

For more details about FreeBSD, please see:

- *http://www.FreeBSD.org/releases/7.0R/relnotes.html*
- *http://www.FreeBSD.org/releases/7.0R/errata.html*
- *http://www.FreeBSD.org/releng/*

SoftIntegration C++ Graphical Library (SIGL)

SoftIntegration C++ Graphical Library (SIGL) is a cross platform C++ graphical library. It provides the simplest possible solution for 2D/3D graphical plotting within the framework of C/C++ for rapid application development. Plots can be generated using SIGL for display in a local monitor, through the Web, or saved in a file with a variety of different file formats. The graphical library is for applications where the convenience of use, speed and performance matter. With Graphical Library, you can design and deploy application running across different platforms. SIGL contains a plotting class with many member functions for visualization. It simplifies the graphical plotting for C++ users.

Ch Standard 6.0

Ch is an embeddable C/C++ interpreter for cross-platform scripting and shell programming. Ch Standard Edition has the following features with over 8,000 C functions:

- A complete C interpreter supporting all features in the ISO C90 Standard.
- Easy interface with C/C++ binary libraries using Ch SDK
- Wide characters in Addendum 1 for C90.
- Ch supports more features than most existing C compilers.
- Complex IEEE floating-point arithmetic with complex metanumbers ComplexInf and ComplexNaN.
- Nested functions and nested classes for modular programming.
- Interactive execution of C statements, perfect solution for classroom demos using a laptop.
- Interactive portable shell across different platforms.
- Adjustable array bounds, i.e., int a[1:10].
- Auto array bound checking.
- TCP/IP socket/WinSock for network computing.
- POSIX for portable scripting.
- Cross platform Unix utilities and commands
- A large number of Integrated Development Environment (IDE) support Ch.
- OpenGL toolkit
- GTK+ toolkit
- X/Motif toolkit on Unix/Linux
- Common Gateway Interface (CGI)
- Open Database Connectivity (ODBC)

pfSense 1.2 Open Source Firewall

pfSense includes almost all the features included in commercial firewalls, and adds its own to make a complete security solution. The firewall can filter by source and destination IP/port for TCP and UDP traffic, IP protocol, is able to limit simultaneous connections on a per-rule basis and much more. Here are the highlights of pfSense 1.2:

- Improve CARP input validation.
- Clarify text and fix typos on several screens.
- Revert DHCP client to default timeout of 60 seconds.
- Reload static routes when an interface IP address is changed by an administrator.
- Fix a few areas allowing potential cross site scripting.
- Fix a couple issues with package uninstalls.
- Shorten firewall rule, NAT and traffic shaper description fields to prevent users from entering description names too long for the pf ruleset.
- Improve efficiency of RRD graph creation by removing duplicate commands.

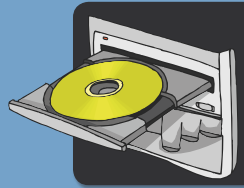
The default login information is:

- Username: admin
- Password: pfsense

Usage information can be found at *www.pfsense.org*.

Blender 2.45 for FreeBSD

Blender is the free open source 3D content creation suite, available for all major operating systems under the GNU General Public License. It can be used for modeling, UV unwrapping, texturing, rigging, skinning, animating, rendering, particle and other simulations, non-linear editing, compositing, and creating interactive 3D applications. Blender has a robust feature set similar in scope and depth to other high-end 3D software such as Softimage|XSI, Cinema 4D, 3ds Max, Lightwave and Maya. These features include advanced simulation tools such as rigid body, fluid, and softbody dynamics, modifier based modeling tools, powerful character animation tools, a node based material and compositing system and Python for scripting for tool creation and prototyping, game logic, importing and exporting from other formats such as OBJ, FBX, DXF, COLLADA and task automation.



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

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

Installing FreeBSD 7.0

Dru Lavigne

This magazine provides a DVD containing the installation program for the FreeBSD 7.0 operating system, its documentation, and over 700 applications. In this article, I will demonstrate how to install FreeBSD, configure KDE and sound, install additional software, and find resources to learn more about FreeBSD.

This article assumes that you have a test system suitable for a fresh installation. This system should meet the following minimum hardware requirements:

- 2 GB hard drive
- Fairly recent video card
- At least 128 MB of RAM (the more, the better)
- A connection to the Internet if you would like access to additional software

This article also assumes that you are interested in DIY or *do it yourself*. FreeBSD is a powerful operating system which provides the flexibility to create a minimal server installation, an *install everything including the kitchen sink* workstation, and every type of configuration in between. This means that you get to choose what to install and what to configure. Configuration is easy, it is just not pre-done for you. If you would rather have an installation program that does everything for you, consider trying instead either PC-BSD (*http://www.pcbsd.org*) or DesktopBSD (*http://www.desktopbsd.net*). These operating systems are both based on FreeBSD and provide a more attractive installation program that automagically sets up KDE, network, and sound for you.

Installation

To start the installation program, boot your test computer with the DVD placed in your CDROM drive. You should see some messages go by as the kernel loads. Then, the FreeBSD installation program known as *sysinstall* will start, providing you with a series of menus. *sysinstall* was designed to allow for any type of FreeBSD install, including minimal installs without a GUI. For this reason, the menus are not pretty, but they are functional. Your mouse will not work during the install – instead use your arrow keys to select items and your tab key to move down to the *OK* option.

The first *sysinstall* menu you will encounter is labelled *Country Selection*. Use your up or down arrow key (or page up or page down key) to select your country, then press enter to move on to the next menu screen. The next menu, seen in Figure 1, allows you to set your *System Console Keymap*. Unless you have a localized keyboard, you can press enter to select the default of USA ISO.

You will then see the sysinstall *Main Menu* screen. Use your arrow key to highlight *Standard* and press enter. You will see a message introducing the fdisk menu; press *enter* to continue.

The *FDISK Partition Editor* menu will show the current layout of the disk with each partition on one line. It will also indicate if you have any free space. If you're installing FreeBSD as the only operating system, use your arrow key to highlight a partition, then press `d` to delete it. Repeat until there is only one line left showing the amount of free space. You can then press `a` to select all of the free space for FreeBSD. Note that FreeBSD will leave 63 sectors as free space to hold the boot code. When finished, press `q` to quit this screen.

The *Install Boot Manager* menu offers three choices for the boot manager. If you are booting multiple operating systems and do not have a boot manager such as LILO, press *enter* to select the BootMgr option. If FreeBSD is the only operating system on this computer, select *Standard*. If you are using another boot manager such as LILO, select *None*. After your selection, press *enter* to leave this screen. You will see another message; press *enter* to move on to the FreeBSD Disklabel Editor screen. Press *a* and FreeBSD will automatically create the partitions it needs. Press `q` to leave this screen.

The *Choose Distributions* menu allows you to select which components to install. Use your arrow key to highlight All, then press tab and *enter* to exit this screen. When asked if you would like to install the FreeBSD ports collection, press *enter* to select *Yes*. When you are

returned to the *Choose Distributions* screen, press *enter* to select *OK* and move on to the next screen.

In the *Choose Installation Media* screen, press *enter* to select the default CD/DVD installation media. The message will remind you that this is your last chance before your hard drive is reformatted and FreeBSD is installed. When ready, press *enter* to start the installation. The installation itself will take some time as FreeBSD, its documentation, the ports tree, and the packages needed for the X window system are installed.

Post Installation Configuration

Once the install is finished, you will receive a congratulatory message. If you press *enter*, you will go through a series of post-installation configuration options. The first asks if you would like to configure any Ethernet or SLIP/PPP devices. You should press *enter* for *Yes* if you do not like to access the Internet from your FreeBSD system. If you are using a network interface card attached to a router or switch or cable modem, the first entry should indicate the FreeBSD name of that interface along with a description of manufacturer of the card. If you instead use a modem card or external dial up modem, select the PPP entry that matches your COM port. See this section (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/userppp.html* of the FreeBSD Handbook for more information on setting up your dialup connection. After highlighting your selection, press *enter*. Press *enter* again to choose *No* to IPv6. If your ISP automatically assignes your IP address, select *Yes* for DHCP configuration. If successful, the network configuration screen will contain your IP settings, as seen in Figure 2. Manually type in a name under host, then tab over to *OK* and press *enter* to leave this menu. If you are prompted to bring up the interface, press *enter* to do so.

Note: if DHCP configuration fails, this section *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/install-post.html* of the FreeBSD Handbook may assist in configuring networking after your first reboot.

You will then be asked a series of yes or no questions. Unless you have a technical reason for selecting a different answer, you can keep the default answers as follows:

- Do you want this machine to function as a network gateway? – No
- Do you want to configure inetd and the network services that it provides? No
- Would you like to enable SSH login? – No
- Do you want to have anonymous FTP access to this machine? – No
- Do you want to configure this machine as an NFS server? – No
- Do you want to configure this machine as an NFS client? – No
- Would you like to customize your system console settings? – No
- Would you like to set this machine's time zone now? – Yes
- Is this machine's CMOS clock set to UTC? – No

You will then be able to choose your continent from a menu, then your country, then your time zone with a confirmation message regarding the selected time zone. Use your arrow keys to make the appropriate selections.

Would you like to enable Linux binary compatability? – Yes

Some applications require this, so accept the default and wait a moment as it installs.

Does this system have a PS/2, serial, or bus mouse? – Yes

Select *yes* unless you plan on only using the command line on your system. In the mouse configuration menu, arrow down to Enable and press *enter* to see if your mouse is automatically detected. When finished, arrow up to *Exit* and press *enter* to leave this screen.

If you do not like to install software at this time, press *enter* to browse the FreeBSD package collection. Otherwise, if you would prefer to install software later, arrow over to *No* and press *enter.* If you select *YES*, you will be presented with a menu of software categories. As an example, we will install the KDE window manager and the firefox web browser. Use your arrow keys to select the x11 category, then select kde-3.5.8. You will see an X as you select KDE. Tab down to *OK* and press *enter*. Select the *www* category, then firefox-2.0.0.12,1. Tab down to *OK* and press *enter*, then tab over to Install. KDE and firefox should show up in the *Package Targets* menu. Press enter and the installation will occur. You will receive a message when it is finished. Again, this will take some time due to the amount of software that is installed with KDE. You will then be prompted to create initial user accounts. You should create at least one account for yourself so press Enter. In the next menu, select *User* and fill in the user details as follows, using tab to move between sections:

- Login ID: your username; this is case sensitive
- UID: leave as-is
- Group: type in the word `wheel` which will allow you to become the superuser
- Password: this is case sensitive
- Full name: you can either leave this as is or type in your full name
- Member groups: leave empty

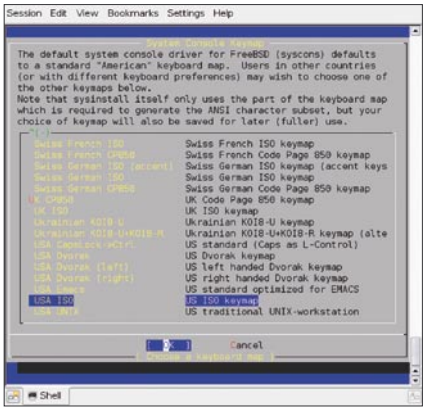


Figure 1. System Console Keymap

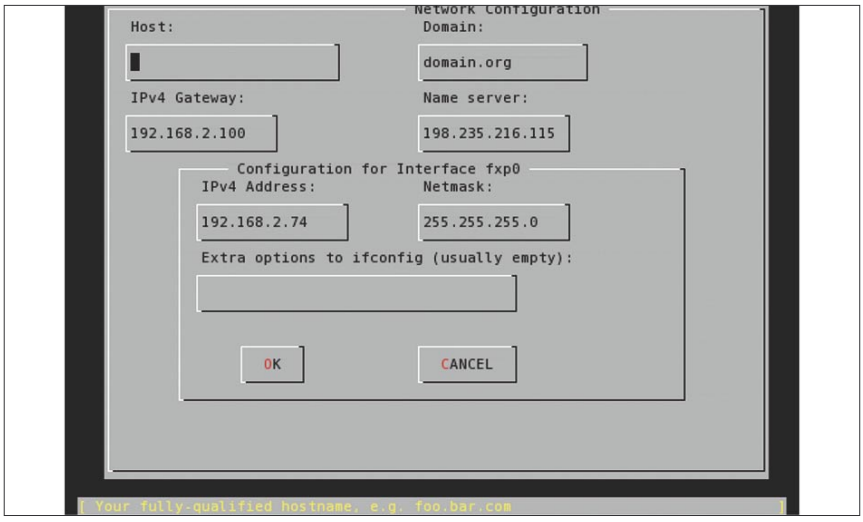


Figure 2. Network Configuration Containing DHCP Settings

- Home directory: leave empty
- Login shell: change to `/bin/tcsh`

After you press enter at *OK*, you will return to the previous menu. You can either select *User* to create another account or Exit to leave this screen. You will next be prompted to set the system manager's (also known as the superuser) password.

The post-configuration stage is now finished. Press *enter* to choose *No* when asked if you would like to visit the general configuration menu. Then select *Exit Install*, then *Yes* to leave the installation utility. Once you see your POST messages, eject the DVD so the system will continue to reboot into the new operating system. (If you forget to remove the DVD, it will reboot from DVD and re-enter the installation program).

Configuring KDE

Once the system reboots, you will receive a login screen. To configure X and KDE. login as the user you created, then type `su` – and input the superuser password. You now have superuser privileges, meaning you can configure the system. Start by running the X configuration script which will probe your video hardware and create a configuration file for you:

```
Xorg -configure
```

Your screen may go blank for a second as your hardware is probed. As you receive your command prompt back, you should see this message:

```
Your xorg.conf file is /root/
xorg.conf.new
To test the server, run 'X -config
/root/xorg.conf.new'
```

I prefer to run this command:

```
startx
```

This should open up a screen with three white windows containing green title bars labelled login or xterm, as well as a clock in the upper right hand corner. This is not KDE, but it does confirm that X is configured properly. If you press [*CTRL*]+[*ALT*]+[*Backspace*], you will exit this screen and return to your command prompt.

Before running KDE for the first time, copy the X configuration file to its permanent location, then leave the superuser account:

```
cp /root/xorg.conf.new /etc/X11/
xorg.conf
exit
```

You have a choice of how to start KDE. If you would like to start it manually when you boot your computer, that is, you want to receive a login prompt first and start KDE later when you feel like it, create a file called *.xinitrc* like so:

```
echo "exec startkde" > .xinitrc
```

Now, whenever you would like to start KDE, type: `startx`. The first time you run this command, the *Kpersonalizer* wizard (see Figure 3) will present you with some screens where you can choose your country and language, system behavior, eyecandy, and theme. Your mouse should work, allowing you to select your desired settings. Should you wish to end your KDE session, click on the *K* button in the bottom left corner and select *Log Out* from the menu. If you click the *End Current Session* button, you will return to your original command prompt. If you want to shut down your computer, type su – to gain superuser privileges, then type halt to safely shutdown your system. If you prefer to have KDE start automagically whenever you turn on your computer, you will instead want to configure KDM, the KDE Display Manager. Instead of creating that *.xinitrc* file as your user account, you'll need to modify a system configuration file as the superuser. You can use the vi editor to do this like so:

```
vi /etc/ttys
```

Use your arrow key to bring your cursor down to this line:

```
ttyv8   "/usr/local/bin/xdm -nodaemon"
xterm   off    secure
```

Type `dd` to erase this line, then i to insert new text. Carefully type this new line:

```
ttyv8   "/usr/local/bin/kdm"
xterm   on   secure
```

then press *enter* so the new line is separated from the existing comment:

```
# Serial terminals
```

When finished, double-check for typos, press the ESC key, and type `:wq` to write and quit this file. You can test that your change has worked by rebooting the system. Type `reboot` and KDE should load after your computer reboots.

You will note that using theis method, you will receive a GUI login menu. If you are the only person using your computer and would like to automatically login without typing in your username and password, click on the K button in the lower left corner

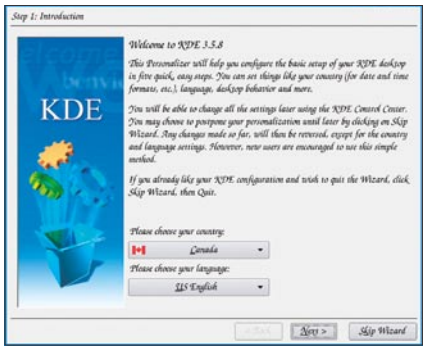


Figure 3. Kpersonalizer Wizard

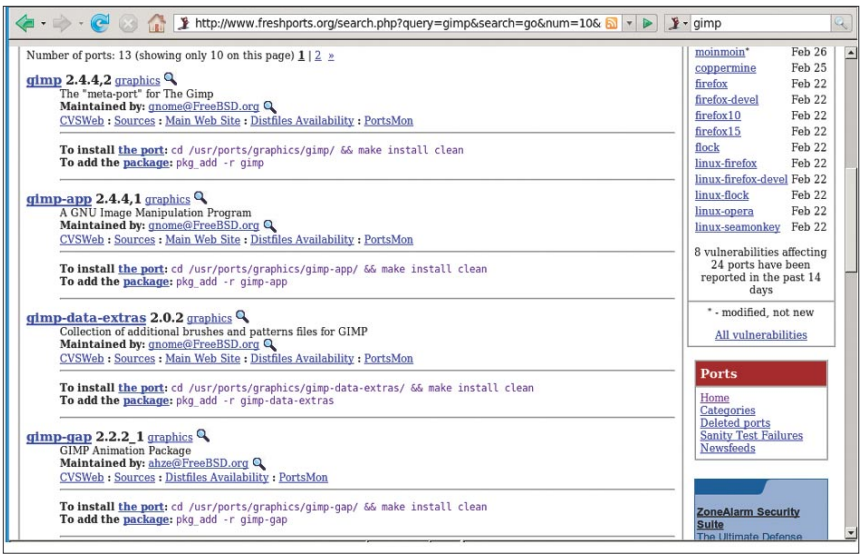


Figure 4. Freshports Search Plugin

and select Control Center. Click on System Administration, then Login Manager, then Convenience. Click the *Administrator Mode* button in the right pane which will prompt you for the superuser password. You can then check the Enable Auto-Login box. Another feature that has been added can be found when you choose to Log Out. In addition to ending your current session, you will now have the option to turn off (halt) your computer or restart (reboot) your computer.

Configuring Sound

You may have noticed an error message regarding sound when you start KDE. FreeBSD comes with everything you need to use sound, but it is up to you to provide the configuration. You will need to be the superuser in order to do so. I prefer to load sound at the console as I can watch for messages as sound is loaded to get more information as the sound card hardware is probed. You can access the console by typing [*ALT*]+[*F1*] – note this will leave KDE and drop you down to a login prompt. (Note, when you are finished, type [*ALT*]+[*F9*] to return to your KDE session.) Login as the superuser, then type:

```
kldload snd_driver
```

After a few seconds pause, you should see some bright white text indicating what type of sound hardware is installed in your computer. Mine looked like this:

```
pcm0: <Creative CT5880-C> port 0xb000-
0xb83f irq 5 at device 8.0 on pci0
pcm0: <SigmaTel STAC9708/11 AC97
Codec>
pcm0: [ITHREAD}
pcm0: <Playback: DAC1,DAC2 / Record:
ADC>
```

You can determine the name of the FreeBSD sound driver associated with your card by using this command:

```
grep kld /dev/sndstat
pcm0: <Creative CT5880-C> at io 0xb800
irq 5 kld snd_es137x [MPSAFE] (2p:
1v/1r:1v channels duplex default)
```

I have highlighted the name of my driver in this output. In your output, look for the word after kld. In my case, the name of the driver is `snd_es137x`. To tell FreeBSD to load this driver every time I boot, I would carefully add this line to /boot/loader.conf:

```
snd_es137x_load="YES"
```

As you may have guessed from the name, this is the configuration file for loading drivers at boot time. Make sure you do not have any typos when you save this file, and replace `snd_es137x` with the name of the driver that showed in your output. Note, you do not have to reboot to start using sound as you already loaded the sound driver when you typed the `kldload` command. However, the next time you do reboot or start your computer, FreeBSD will read this file and load the sound driver for you.

Installing Additional Software

As of this writing, there are over 18,000 applications which are available for installation on your FreeBSD system. The easiest way to find FreeBSD software is to use the search function at the Freshports *http://freshports.org* website. If you use the firefox web browser, you can install a Freshports search plugin from: *http://www.searchplugins.net/pluginlist.aspx?q=freshports&mode=title*. Figure 4 shows an example search for the gimp imaging application using this plugin. This page was accessed by typing *gimp* into the search bar in the upper right hand corner of firefox.

Freshports will show you which applications are available that match your search criteria. For each application, it will also give the instructions for installation of either the port or the package. As a new user, using the `pkg_add` command to install the package is the quickest and easiest way to install new software. `pkg_add` will automagically install the application for you, as well as any dependent software needed for that application to run properly.

If you would like to see what software is already installed on your system, use this command:

```
pkg_info | more
```

You can also use `pkg_info` to determine which files were installed with an application. For example, to *list* what was installed with firefox, type:

```
pkg_info -Lx firefox
```

The ease and flexibility of dealing with software is one of the primary strengths of FreeBSD. Chapter 4 of the FreeBSD Handbook (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports.html*) can get you started. The books mentioned at the end of this article also contain entire chapters on installing and managing FreeBSD software.

Learning More about FreeBSD

This article was meant to get you started with FreeBSD by walking you through the installation and configuration of a FreeBSD 7.0 system running KDE. There are several excellent books available to help you learn more about FreeBSD. These include:

- Best of FreeBSD Basics *http://reedmedia.net/books/freebsd-basics/*
- Absolute FreeBSD *http://www.absolutefreebsd.com/*
- Complete FreeBSD *http://www.lemis.com/grog/Documentation/CFBSD/*

The first two books are available for purchase and the third book has been open sourced, meaning you can download it and read it for free.

The FreeBSD Handbook, *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/*, and FAQ, *http://www.freebsd.org/doc/en_US.ISO8859-1/books/faq/*, are also excellent resoures. A local copy of these were installed with the operating system and can be found in */usr/share/doc*. To read the local copy of the handbook, point your browser to */usr/share/doc/handbook/index.html*, and for the FAQ, point to */usr/share/doc/faq/index.html*.

About the Author

Dru Lavigne is a network and systems administrator, IT instructor, author and international speaker. She has over a decade of experience administering and teaching Netware, Microsoft, Cisco, Checkpoint, SCO, Solaris, Linux and BSD systems. A prolific author, she pens the popular FreeBSD Basics column for O'Reilly and is author of BSD Hacks and The Best of FreeBSD Basics. She is currently the Editor-in-Chief of the Open Source Business Resource, a free monthly publication covering open source and the commercialization of open source assets. She is founder and current Chair of the BSD Certification Group Inc., a non-profit organization with a mission to create the standard for certifying BSD system administrators.

FreeBSD's bsnmp

Michael W. Lucas

Any large datacenter or network uses the Simple Network Management Protocol (SNMP). Many Unix-like operating systems use the Net-SNMP toolkit from http://www.net-snmp.org. While Net-SNMP is a powerful and flexible SNMP implementation available for many platforms, it does not support BSD-specific functions.

The amount of data Net-SNMP provides from the BSD kernel is limited, and Net-SNMP does not support BSD features such as the pf(4) packet filter or bridges. FreeBSD 6.0 and later includes the Begemot SNMP daemon, a lightweight SNMP agent that reports on these BSD-specific functions as well as all of the things you would expect from a SNMP agent.

The Begemot SNMP daemon, or bsnmpd(8), is specifically designed to be very small but extensible. The main daemon provides only basic functions for speaking through the network, but all functionality that reads data from the host is provided via external libraries.

This design makes bsnmpd very useful for small and embedded systems, as well as an excellent choice for a datacenter client. If you are a programmer, you can add additional libraries to make it work with your own programs.

If you are required to run net-snmp, bsnmpd can run as a net-snmp sub-agent.

We will look at SNMP in general, and see how SNMP provides information to a network manager. Then we will see how to use SNMP clients on your workstation. We will look at how to configure bsnmpd on FreeBSD, and how to enable and disable different snmp functionalities. Finally, we will look at some of the FreeBSD-specific information that you can read with bsnmpd.

SNMP Basics

SNMP works on a classic client-server model. The snmp agent is daemon that runs on a server and provides information to SNMP clients upon request.

SNMP agents can provide either read-only or read-write access to the system. Most UNIX-like systems provide read-only SNMP services, as UNIX expects to be configured via the command line. A great deal of network equipment is configured via read-write SNMP, but that is rare in the UNIX world. No system administrator I know is comfortable managing their system via SNMP, and read-write access is disabled by default on FreeBSD. With this in mind, we will focus specifically on read-only SNMP.

In addition to having an SNMP agent answering requests from an SNMP client, servers can also transmit SNMP traps to a trap receiver elsewhere on the network. SNMP agents generate these traps in response to particular events on the server, much like syslogd(8) messages. Again, most UNIX-like systems do not send SNMP traps.

The main purpose of SNMP in the BSD world is to provide system information to external monitoring software. Monitoring and network management systems such as Nagios, HP OpenView, MRTG, and CiscoWorks use SNMP queries to evaluate the status of different system services and facilities. Use SNMP when you want your management system to graph how full your disks are, or alarm if the DHCP server fails, or trigger a notification when your packet filter starts receiving large numbers of malformed packets.

Each SNMP agent can extract certain information from the local system. The agent provides this information in a hierarchical tree called a Management Information Base, or MIB, in ASN.1 format. Each branch of the MIB tree has very general categories: network, physical, software, and so on, with subdivisions in each.

Think of the MIB tree as a well-organized filing cabinet, where individual drawers hold specific categories of information and files within drawers hold individual facts. Similarly, the uppermost MIB con-

tains MIBs beneath it. For example, here is a MIB from my laptop.

```
Interfaces.ifTable.ifEntry.ifDescr.3=
STRING: "fxp0"
```

This is from the Interfaces MIB tree. If this machine had no interfaces, this branch of the tree would not exist. The *ifTable* is the interface table within the interface information, and *ifEntry* means that this describes one particular interface within the table.

IfDescr means that we are looking at the description of one particular interface. This entry can be summarized as *Interface number 3 on this machine is called fxp0*. Another set of entries in the interfaces table might provide how many packets have been processed by each interface. Interface 3 in that other table would correspond to interface 3 in this table.

MIBs can be expressed as numbers – for example, the preceding MIB can also be expressed as `.1.3.6.1.2.1.2.2.1.2.3`. Some of you have probably noticed that the words have 5 terms separated by periods, while the number has 11 parts, but they are supposed to be the same thing.

The numerical MIB is longer because it includes the default `.1.3.6.1.2.1`, which means `.iso.org.dod.internet.mgmt.mib-2`. This is the standard subset of MIBs used on IP networks such as the Internet.

The vast majority (but not all) SNMP MIBs have this leading string in front of them, so nobody bothers writing it anymore when the MIB is being printed for human use. Since the machine needs the specific number, numerical MIBs include everything.

MIBs are defined according to a very strict syntax and are documented in MIB files.

Each SNMP agent has its own MIB files. Bsnmpds' MIB files are in `/usr/share/snmp`.

While you can read and interpret them by eye, I highly recommend copying them to a workstation and installing a MIB browser so that you can comprehend them more easily. MIB browsers are GUI clients that present MIB files in a complete and easy-to-read format, with definitions of each part of the tree and descriptions of each particular MIB.

SNMP Security

Many security experts state that SNMP really stands for *Security: Not My Problem!* This is unkind, but true. Only use SNMP behind firewalls on trusted networks. If you must use SNMP on the naked Internet, use packet filtering to keep the public from poking at your SNMP agent. SNMP agents run on UDP port 161.

SNMP provides basic security through community names. The SNMP purists offer all sorts of explanations on why a community name is not the same thing as a password, but to the IT practitioner a community name is just a password. Most SNMP agents set two communities by default: public (read-only) and private (read-write). Yes, there is a well-known read-write default. Whenever you encounter a SNMP-managed device, be sure to change all the defaults community names!

SNMP comes in two versions. Version 1 was the first attempt, with version 2c being the modern standard. You will see references to SNMP version 3, which uses advanced encryption to protect data on the wire.

However, very few vendors actually implement SNMPv3. Bsnmpd uses SNMPv2c.

This means that anyone with a packet sniffer on your network can capture your SNMP community name, so be absolutely certain you are only using SNMP on a trusted network.

Control client access to bsnmpd through TCP wrappers in `/etc/hosts.allow`.

Using FreeBSD as a SNMP Client

FreeBSD does not include a SNMP client, only the agent. If you want to make SNMP queries from a FreeBSD workstation, install Net-SNMP (`/usr/ports/net-mgmt/net-snmp`). If you install from the *Ports Collection*, the build process will ask several questions. Just accept the defaults for each, as they are only relevant if you are using Net-SNMP as a server.

While Net-SNMP installs several programs, we will only use `snmpwalk(1)` and `snmptranslate(1)`. Read the manual pages for the other programs if you are really interested.

Wait a minute, I hear some of you saying. *I thought you said that bsnmpd meant you did not have to install Net-SNMP, and now you are telling us to install Net-SNMP! What gives?* Bsnmpd is the agent, or server. Net-SNMP includes SNMP clients as well as a server. You need a SNMP client program to access a SNMP agent, much as the Apache Web server is useless without Firefox or Safari.

You only have to install Net-SNMP on your client workstation, but your hundreds of FreeBSD servers only need `bsnmpd`.

To get information from a SNMP agent use `snmpwalk(1)`. This program lets you query a host with a community name and view the information provided by the agent. The basic syntax for snmpwalk is:

```
# snmpwalk -v version -c community
hostname MIB
```

If you do not get an answer, either you are using the wrong community name or traffic to the SNMP port on that host is filtered in some way.

For example, here I use SNMPv2c to query my webserver with a community name of *public*.

I do not include a MIB, so I am asking snmpwalk to start at the beginning of the MIB tree and print out every MIB the agent offers.

```
$ snmpwalk -v 2c -c public webserver
```

If this host is running a SNMP agent, and you have used the correct community name, you will start to see information sprawl across your screen. Most SNMP agents offer at least a few hundreds MIBs, but some offer tens or hundreds of thousands of MIBs. Perhaps you know a more specific MIB. Hey, we had one earlier!

```
$ snmpwalk -v 2c -c public webserver
.1.3.6.1.2.1.2.2.1.2.3
RFC1213-MIB::ifDescr.3 = STRING:
"fxp0"
```

This is not terribly useful – odds are, the names of the interfaces on this machine do not change very often. I recommend capturing the output of a complete snmpwalk to a text file so you can browse it at your leisure and see what your SNMP agents on different systems offer. For ex-

ample, here is a thin slice of snmpwalk output:

```
...
IP-MIB::ipInReceives.0 = Counter32:
331608
IP-MIB::ipInHdrErrors.0 = Counter32: 4
IP-MIB::ipInAddrErrors.0 = Counter32:
0
...
```

Some of the MIBs you will see appear self-explanatory. I can make an educated guess what all of these means. Even ones with obvious names can be misleading, however.

When you see a MIB that looks like it might be interesting, be sure to check its definition in the MIB file. You can do this with `snmptranslate(1)`, using the `-Td` flag. Listing 1.

This provides a full description of the MIB from the MIB file, as well as a name-to-number translation of the MIB. This is especially useful when investigating proprietary devices.

Now that you can talk to SNMP agents, let's configure bsnmpd.

Configuring Bsnmpd

To enable bsnmpd, set `bsnmpd_enable=YES` in `/etc/rc.conf`. Start, stop, and restart bsnmpd with `/etc/rc.d/bsnmpd`. Before you start bsnmpd, however, configure the daemon in `/etc/snmpd.config`.

In addition to including the default community name of public, the default configuration does not enable any of the FreeBSD-specific features that make bsnmpd so desirable.

Bsnmpd uses variables to assign values to configuration statements. Most high-visibility variables are set at the top of the configuration file, as you see here:

```
location := "Basement Datacenter"
contact := "mwlucas@blackhelicopters
.org"
system := 1     # FreeBSD
traphost := localhost
trapport := 162
```

These first variables define values for MIBs that should be set on every SNMP agent. The *location* describes the physical location of the machine, and every system needs legitimate contact information. Leave the default system type of FreeBSD unchanged. If you have a SNMP trap receiver on your network, you can also set its hostname and the UDP port used.

```
# Change this!
read := "mySecretString"
# Uncomment begemotSnmpdCommunityStri
ng.0.2 below that sets the community
# string to enable write access.
write := "geheim"
trap := "mytrap"
```

The read string defines the read-only community name of this SNMP agent. The default configuration file advises you to change it, which I have done already. The write string is the read-write community name, which is disabled by default further down in the configuration file. You can also set the community name for traps sent by the agent.

Liating 1. Using `-Td` flag

```
$ snmptranslate -Td IP-MIB::ipInHdrErrors.0
IP-MIB::ipInHdrErrors.0
ipInHdrErrors OBJECT-TYPE
  -- FROM       IP-MIB, RFC1213-MIB
  SYNTAX        Counter32
  MAX-ACCESS    read-only
  STATUS        mandatory

  DESCRIPTION   "The number of input datagrams discarded due to errors in
       their IP headers, including bad checksums, version number
       mismatch, other format errors, time-to-live exceeded, errors
       discovered in processing their IP options, etc."

::= { iso(1) org(3) dod(6) internet(1) mgmt(2) mib-2(1) ip(4)
ipInHdrErrors(4) 0 }
```

This provides enough information for bsnmpd to start and run. You will not get any special FreeBSD functionality, however.

Bsnmpd Variables

bsnmpd uses the variables you set at the top of the configuration file to assign values to different MIBs later in the configuration.

For example, at the top of the file I have set the variable read to mySecretString. Later in the configuration file you will find these statements:

```
begemotSnmpdCommunityString.0.1 =
$(read)
####################################
# begemotSnmpdCommunityString.0.2
# = $(write)
####################################
```

This sets the MIB `begemotSnmpdCommunityString.0.1` equal to the value of the read variable. The line to set the write string is disabled.

Why not just set these values directly? Bsnmpd(8) is specifically designed to be extensible and configurable. Setting a few variables at the top of the file is much easier and less error-prone than directly editing the rules further down the file.

Let's go back to this begemotSnmpdCommunityString MIB set here. Why set this?

Go into your MIB browser and you will see that this is the MIB that defines a SNMP community name. Yes, you probably could have guessed that, but system administration is not about guessing.

Throughout the configuration file, you will find similar MIBs being set with these variables.

Bsnmpd Modules

Most of bsnmpd's interesting features are contained in modules. Enable modules in the configuration by giving the begemotSnmpdModulePath MIB a class that the module handles and the full path to the shared library that implements that feature.

For example, in the default configuration you will find a commented-out entry for the PF bsnmpd module.

```
begemotSnmpdModulePath."pf"     = "/
usr/lib/snmp_pf.so"
```

The *pf* subtree of begemotSnmpdModulePath is defined in the shared library `/usr/lib/snmp_pf.so`.

If you are running PF, uncommenting this and restarting bsnmpd will make the agent read information from your PF packet filter.

FreeBSD 7.0 ships with the following FreeBSD-specific modules.

All of these modules are disabled by default, but can be enabled just by uncommenting their configuration file entries.

- `netgraph` – This provides visibility into the Netgraph system, letting administrators and developers track how different Netgraph modules pass data amongst themselves.
- `pf` – This lets you track packet filter information, including the number of packets matched on different rules, the size of various tables and how often they are matched, the number of packets blocked, and bandwidth queue information.
- `hostres` – This implements the Host Resources SNMP MIB, letting SNMP provide information about storage and hardware utilization and errors, as well as what programs are installed or running on the machine.
- `bridge` – This gives visibility into bridging functions, such as bridging types, errors, and RSTP.

Restart bsnmpd after enabling any of these. If the program will not run, check `/var/log/messages` for errors.

Bsnmpd provides easy access to FreeBSD-specific information for your network management system without installing any additional software on your server.

Using bsnmpd eliminates the need to install any other SNMP agent on your system.

About the Author

Michael W. Lucas is a network manager, author, and FreeBSD committer. His most recent book is *Absolute FreeBSD*, from No Starch Press.

Pushing BSD as an open source desktop

Jan Stedehouder

It may come as a surprise for the majority of computer users, but open source operating systems have a long and solid track record. They form the backbone or crucial building blocks of the ICT infrastructure of companies and organizations. In a world where desktop computing has equaled Windows for such a long time, the ascent of the open source desktop may appear as something new but those desktops are firmly rooted in decades of development.

On the other hand, promoting open source operating systems like Linux and BSD to the desktop bringst a whole new set of challenges. The large majority of end users are accustomed to functional graphical interfaces, to plug and play for most if not all of their peripheral devices and where running a program begins with double-clicking an icon. To create an open source desktop that closely resembles this experience is not an easy task.

Linux has stolen the limelight of attention for the open source desktop with Ubuntu taking the lion's share of that attention. Since entering the open source desktop market in 2005 Ubuntu has been growing and actually been able to draw new groups of users to Linux. However, Linux is not the only open source operating system. In this article, we will focus your attention to two open source desktops, which are firmly routed in FreeBSD: PC-BSD and DesktopBSD.

BSD is not Linux

Before we take a closer look at PC-BSD and DesktopBSD it is good to realize that BSD is not Linux nor a fork of Linux. There are a lot of similarities and more experienced Linux users should encounter only a few problems when dipping their toes in BSD. Both BSD and Linux are Unix-like systems with BSD having the firmest roots in Unix history. The development of Linux is described as *release quick, release often* where the push forward sometimes comes at the price of loosing stability. BSD development has a much stronger focus on stability which sometimes results of not providing the *bleeding edge* to the most eager of users. Another big difference can be found in the two key licenses, the GPL v2 for Linux and the BSD license. Both are open source licenses, but the BSD license allows a developer to take BSD-licensed code and incorporate it in a closed source work. The GPL license prohibits this.

This does not mean that a Linux distribution is completely GPL. Open source software is released under a plethora of open source licenses, but that software can peacefully co-exist in the same distribution. For the end user, both under BSD or Linux, this is a good thing, because it gives the maximum access to open source software possible.

One major difference between BSD and Linux is that BSD is developed as a complete operating system, where with Linux all components (kernel, applications etc.) are developed separately and are packed by the distributors.

What are the goals of PC-BSD and DesktopBSD?

The main ideas behind both PC-BSD and DesktopBSD were developed in 2004. One might say they started with the question: *What does it take to create a good open source desktop*? Both teams took FreeBSD as their starting point. From here different routes were taken. DesktopBSD set out to develop a set of graphical tools to make it easier to accomplish various tasks under FreeBSD, like software management, mounting and unmounting of devices and configuring the network. For PC-BSD it meant to develop a completely new way to install and manage software for FreeBSD-based systems, which led to the PBI-system. Both BSD's are similar in that they added a graphical installer to aid the end user in installing a FreeBSD-based system and a preconfigured KDE desktop.

Before we continue, let us look at some technical data. DesktopBSD is now -at the time of writing- at release 1.6 RC3, based on FreeBSD 6.2-STABLE. It is delivered as two CD iso files, the second disk contain-

ing the language files. The final release will be a live dvd with a wider collection of software than currently available. PC-BSD is also delivered as two CD iso files. The second disk contains various PBI's. At the time of writing you can download 1.4.1, but an upgrade to 1.4.1.1. is already available. This version is based on FreeBSD 6.3-PRE-RELEASE.

Installing PC-BSD

We will first look into the PC-BSD installer. It is divided into six steps, the first of which allows you to set the system language, the keyboard layout, your time zone and whether you wish to report back to bsdstats. The last option is enabled by default. The next two steps are simple: accepting the license and selecting the installation option (fresh or update/repair). In the fourth step you set up up the administrator and user accounts. You enter the password for the administrator account and enter the user name and password for the first user. There is a possibility to alter the shell the user will use. The default is csh, but bash is available as well.

The fifth step is the one that normally gives the most headaches for novice users: preparing and setting up your hard drive. The partition editor shows the detected hard drives in the top panel. You have the choice to customize the partition layout. If you opt for that you can edit a suggested layout and increase or decrease the size of the partitions, either by entering the correct size manually or use a slider. The same window allows you to set a mount point.

The sixth step is called System Components and allows you to select the PBI's you want to have added to your system during install. These include OpenOffice.org, Firefox, K3b, Opera, Koffice, the Ports Collection and the system source. Having the ports collection on the disk and installed from there is definitely a time saver.

During the installation process you are provided with sufficient feedback on each of the steps. The developers took time to write concise paragraphs to explain what each step is about and what you should do. The layout of each screen really breaths a business-like atmosphere and a presentation as a mature product.

After reboot you are taken to the final step via the Display Setup Wizard. You can select your resolution, your color depth and the video driver you are using. This wizard remains at your disposal since it is part of the boot loader menu (Figure 1).

Installing DesktopBSD

The last step in the PC-BSD installer is actually the first step in the DesktopBSD installer. The first step is to initialize the graphical environment and for this DesktopBSD probes your system. It then suggests a screen resolution followed by a window to select your keyboard layout.

From here you have the choice either to go to the live desktop or to start the installation routine. It is definitely good to have a choice since it always takes some time for a live desktop to be up and running. Now you can take a short cut and install DesktopBSD right away.

The installation routine asks for the language you wish to use and then provides you with an overview of the hardware that has been detected. The next three steps gather necessary information like whether it is a new installation or an upgrade/recovery, which hard disk you wish to use and whether you need a boot loader or not.

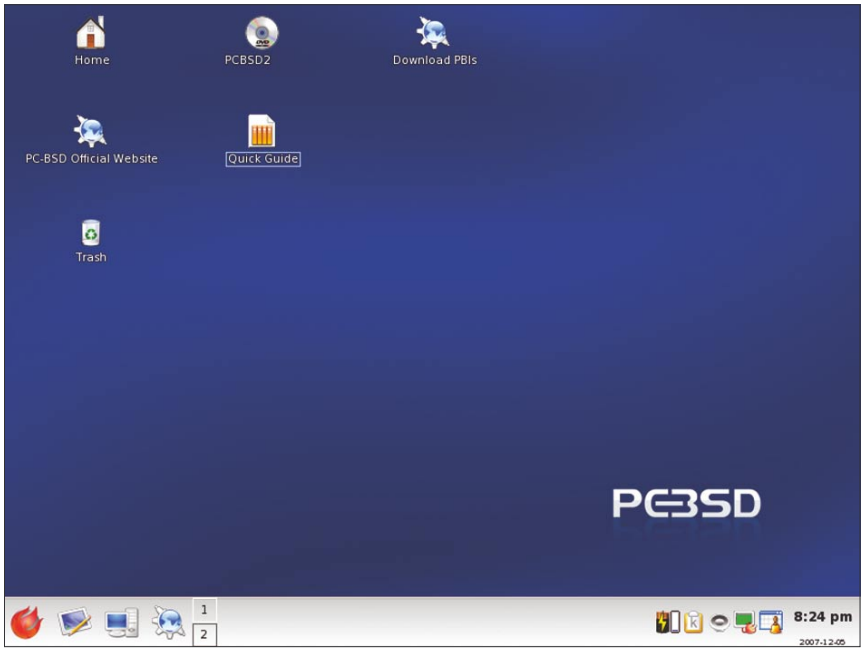


Figure 1. PC–BSD install

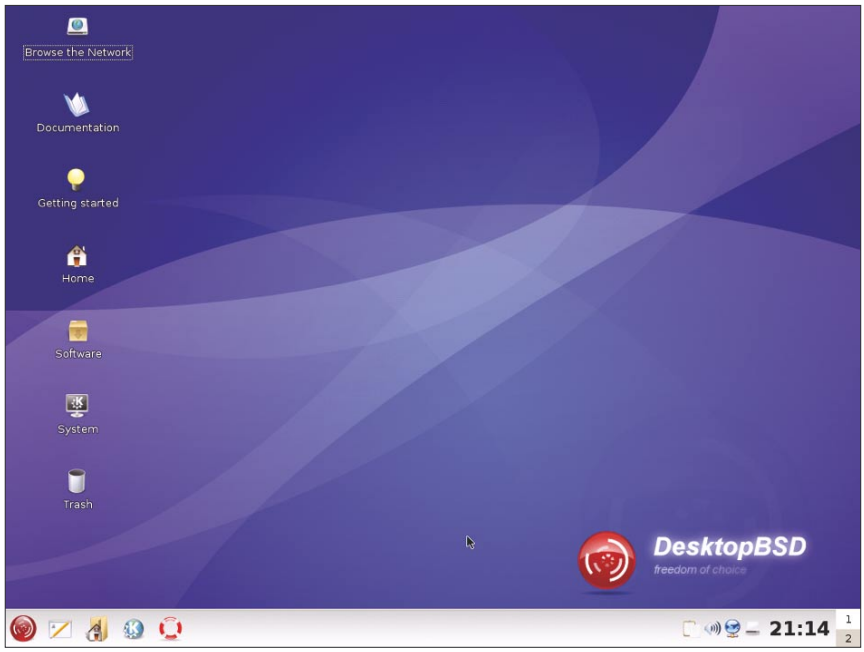


Figure 2. DesktopBSD install 28

Disk partitioning is again taken care off by selecting the disk and adding partitions and slices. The option *Use entire disk* is the easiest to use of course. There is no option to customize a suggested layout and changing the layout is less friendly than under PC-BSD.

DesktopBSD now commences by installing itself on your hard drive. After reboot you can add additional language support, the files of which are on the second ISO, as part of the Initial Setup Wizard. You can provide a new name for your system. The next step requires you to set a system password and add a user. This step is broken down in smaller steps, each taken care of in pop up windows.

With this the installation routine is almost finished. There is just the matter of allowing your system to report back to bsdstats and to read the introduction to DesktopBSD. It is only a few screens and for new users highly recommended (Figure 2).

To make a full comparison to PC-BSD we should also take another step into account. PC-BSD has the ports collection as part of the installation routine. Under DesktopBSD the ports collection still has to be downloaded. The Package Manager takes care of that.

Exploring the default desktops

Both PC-BSD and DesktopBSD use the KDE desktop with a very clean and business-like look and feel. PC-BSD uses a custom PC-BSD 2 theme, where DesktopBSD makes use of a custom icon theme based on NuoveXT. Both of them have shortcuts on the desktop to provide easy access to software management and information resources that are useful for novice users.

First impressions also count when it comes to the default software collection. Through packages and ports new users do have access to over 17.000 packages, but they will first look at what is available right from the beginning. The DesktopBSD team decided to include Firefox, Thunderbird and Pidgin, where the PC-BSD team is more K-oriented with Konquerer, Kontact and Kopete. The only other distinction between the two is the inclusion of Compiz Fusion by PC-BSD.

The two BSD's are multimedia enabled, which means you can play your music and video files out of the box. Amarok is there on both desktops with VLC the video player for DesktopBSD and Kmplayer for PC-BSD (Table 1).

Updating the system

Both Windows and Linux users no doubt will next look to an update option. This is part and parcel of their everyday experience. PC-BSD has this option under *Settings->Software & Updates-> Online Update Manager*. The *Online Update Manager* (OUM) is not enabled by default, but the user can simply click on the *Check Now* button. Since I had downloaded the 1.4.1 version, OUM greeted me with the message than a patch was available to upgrade the system to 1.4.1.1. A 295 Mb patch that is.

This is not the only update function for PC-BSD as there is also a *PBI Update Manager* (PUM). PUM checks whether newer versions of installed PBI's are available. If there are updates and you click on *Get Update* this launches the PBI website for you to download and install the package manually.

DesktopBSD users can click on the icon *Software* which launches the Package Manager. The Package Manager is a graphical fronted that takes care of various tasks. When you run it for the first time it explains that it needs the ports collection and requests your permission to download it (portsnap). This takes quite some time. Once this is done, it then tells you it needs to do a check for possible vulnerabilities on the installed packages (portaudit). It found 18.

The Package Manager window appears, giving you access to the upgrade function. Just go to the tab *Installed packages* and click on *Upgrade all*. This will move the packages that have available updates to the *Pending Operations* cue. Clicking on *Start* should be enough now. However, before you do this I would recommend changing at least one setting. Just go to *Packages -> Settings -> Tab Advanced* and tick the box with *Force processing of further packages even if prerequisite packages have failed to upgrade*. Without this setting the upgrade process halts quite quickly. With 223 packages needing an upgrade (of 448 installed) this process is best left to the nightly hours.

Focus: DesktopBSD tools

The DesktopBSD tools are a collection of programs that aim to make various tasks as easy as possible. Sometimes they are so unobtrusive and so matter of fact you can easily overlook them. For instance, it is very useful for a laptop user to see what the status of your battery life is. You would not want your laptop to go down without a warning.

Same thing for providing information about the network you are attached to. Some people just want to see a network icon in their panel. Double-clicking the network icon in the panel launches the network manager. Through the network manager you can set up your local network, your wireless connections and your DSL connection.

Two other tools are meant to make life easy for managing external drives and sticks: Hardware Notifications and Tray Mounter/ Mount Control. The first tool simply reports whether new hardware is attached or detached. The second tool makes it possible to mount or unmount partitions and devices. With Mount Control it becomes a matter of clicking on the correct partition entry. When it is not mounted it will get mounted and vice versa. It is so simple you almost forget to appreciate it, but it is much simpler and easier to use than similar tools under Windows and Linux.

Table 1. Comparison of DesktopBSD and PC-BSD

	DesktopBSD	PC-BSD
Internet	Firefox	Konquerer
	Thunderbird	Kontact
	Pidgin	Kopete
Multimedia	Amarok	Amarok
	K3b	K3b (via PBI)
	VLC	KMplayer
	Noatun	Kaffeine
Eye candy		Compiz Fusion
		Superkaramba

Adding, removing and locking users is not something most of us do everyday. Still, it is nice to have a graphical tool at hand that does the job. *With Settings (2) -> Security & Privacy -> User Management* you can launch the DesktopBSD specific tool. With a few clicks new users are added. Permissions have been kept simple with only three groups: system administrator, extended device access and user.

The Package Manager takes care of software management and is a graphical fronted to both the packages and the ports collection. Again, like on so many other places, you are provided with quite a lot of explanation and feedback in the windows that you are using. It is such a simple concept that you wonder why it has not been done more often. Instead of referring to manuals, man pages, how to's and generate non-descriptive messages, the developers explain in a few clear lines what the function is about.

The default settings of Package Manager only allow packages to be installed and not ports. In some cases this results in failed installations. The more experienced user will check Freshports and find out why it could not be installed. Package Manager could use a bit of polish here and provide somewhat more feedback. The settings could be changed in order to select the port when no package is available or when the port is more recent, but that usually results in even lengthier installations.

There is basically just one problem with Package Manager and that has less to do with the program itself but with it is settings. It is not uncommon to find a package in the ports collection and not being able to install it via Package Manager because it ca not find the package. When using `#pkg_add -r via` the com-

Interview with Peter Hofer

Peter Hofer is one of the founders and key developers of DesktopBSD

Q: When did you start with DesktopBSD and what was the general idea behind it?

A: We started with DesktopBSD back in 2004 as two students using FreeBSD on a daily basis. We realized that FreeBSD would make a fine desktop operating system, provided it had a desktop environment and some additional software to complement it. DesktopBSD started as a loose collection of graphical tools for FreeBSD, but we already envisioned a fully-featured, stand alone operating system that could easily be installed by someone without prior experience with BSD.

In mid 2005 we could present our first release candidate and launched the final 1.0 release in March 2006. Looking back I can say we came a long way since the early days and are lot closer to achieving our goals.

Q: If you were to mention the most compelling feature of DesktopBSD, what would that be?

A: The installation routine is very easy to use and thus probably the most important feature for most desktop users. They do not want to know about blocks and cylinders, mounts points or network services. Most users just want to get their systems up and running and the DesktopBSD installer does just that. The graphical installer gives the new users a first taste of a fully featured and easy to use FreeBSD system. We believe that once they have experienced this, it makes them more willing to learn to work with it and get to know FreeBSD.

I also consider our package manager one of the centerpieces of DesktopBSD. It enables the users to easily install, upgrade and maintain software using the FreeBSD ports collection. That collection has almost 18.000 different software packages, everything from browsers and instant messengers to media players, office suites, scientific applications, games and much more. This way every user can easily expand his/her open source desktop.

Q: Developing an open source desktop is quite an undertaking. How many people are actively contributing to the development of DesktopBSD?

A: Our core team consists of three people. Oliver Herold is taking care of our community. He keeps in touch with our users and takes care of the publicity. Daniel Seuffert is very important for DesktopBSD as the organizer. He also represents our project at various events. Finally, myself, as the third member of the core team. I am the main developer and thus responsible for the most parts of our software.

We shouldn't forget that there are many other people active in the community. They are busy with translating DesktopBSD to other languages, they contribute to the handbook or help our other users in the forums, mailing lists or our IRC channel. In the end it is the cooperation between and contribution of many that pushes DesktopBSD forward.

Q: What do you consider to be the biggest challenges ahead for DesktopBSD?

A: One of our challenges is to work efficiently and get the things done in the time we have on our hands. None of us can work on DesktopBSD full time and there is only so much spare time. Maybe we make it more difficult by emphasizing stability as much as we do and take the time needed to test things more thoroughly, but we do feel this emphasis on stability is justified.

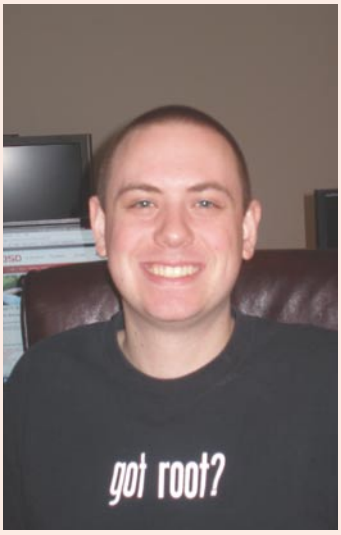
Another challenge is to broaden the user base of DesktopBSD. Linux is getting the most attention in the open source community with BSD being more of an outsider. That alone is a serious challenge in promoting any FreeBSD-based operating system.

Q: What can we expect, let's say, a year from now?

A: A year from now you can expect a release candidate of DesktopBSD 2.0, based on FreeBSD 7 and with a KDE 4 desktop. You will definitely see improvements in power management, enhancements to the package manager and better wireless LAN support.

Interview with Kris Moore

Kris Moore is the founder and one of the key developers of PC-BSD



Q: When did you start with PC-BSD and what was the general idea behind it?

A: I developed the idea of PC-BSD around the end of 2004 and we managed to have a first public beta release in the first quarter of 2005. PC-BSD was developed out of a personal frustration with the various package management methods in use by almost all open source operating systems at that time. All of those methods make each application and libraries part of the base operating system, which means that changes to one library or application could effect all others. In my opinion this was hurting true adoption of open source systems on the desktop. Apart from this, it also meant that all of your user applications were still dependent on the distributor of the operating system and this hinders true freedom.

What I had in mind at that time was a small and stable base operating system that would be separate from the applications run by the user. This led to developing the PBI system. The PBI systems aims to make applications completely self-contained and separate from the underlying operating system. This way, when packages or libraries change, the PBI's won't be affected.

Q: If you were to mention the most compelling feature of PC-BSD, what would that be?

A: For the reasons I mentioned already the PBI system is probably the most distinctive feature. But what is also interesting is that many users simply love the graphical installer, as it gives them a FreeBSD-based desktop without much pain and effort. You can have a typical system up and running in less than 15 minutes and that includes support for flash and 3D acceleration, to mention a few things.

Q: Developing an open source desktop is quite an undertaking. How many people are actively contributing to the development of PC-BSD?

A: We shouldn't forget that a lot of work is being done by the people that develop FreeBSD and maintain the ports and the packages. When it comes to PC-BSD specific work, we have now between five and ten *active* developers, which includes the people working on the graphical user interface and it's tool and the PBI system. That number will grow in the near future when we have our PBI autobuild system on line. This will allow more developers to help maintain the PBI's and monitor automatic PBI builds from ports. You could compare it to the system of port committers.

Q: What do you consider to be the biggest challenges ahead for PC-BSD?

A: One of the bigger challenges is the one we share with all open source operating systems and that is to provide good hardware support. You can see steady improvements all over the board due to enhancement in the FreeBSD base operating system. The biggest challenge at this point is probably having a working Flash 9 product, either BSD native or through Linux emulation (JS: there is a PBI for Firefox 2.0.0.7 with Flash 9 support that uses the Windows version of FF and wine). We do have Flash 7 support, but we see a growing number of websites that are upgrading to Flash 8 or 9.

Q: What can we expect, let's say, a year from now?

A: There are two main developments we'd like to incorporate in PC-BSD: FreeBSD 7 and the new KDE 4 desktop. A year from now you will also see a new upgrade manager. This will help you to keep your system up-to-date and apply patches. Secondly, it will also monitor the installed PBI's and give you the option to upgrade them as well.

mand line the package is downloaded and installed without a problem. This seems to be caused by the ftp-server that is used by Package Manager and it not being completely in sync. Other than that, this is a great tool for software management.

Focus: PBI's

The PBI's are unique to the world of BSD, though there are similar initiatives in the world of Linux. Just think about autopackage or klik. One of the key problems in installing and managing programs are the various dependencies. The phrase *dependency hell* comes to mind. Software management has improved and for most users dependency problems are taken care of by package managers. PC-BSD took a different route in solving the problem and designed the PBI's as complete, self-contained installers that take care of the program and all its dependencies without influencing the underlying operating system. You can see that literally in the file system, where PC-BSD has entries for */PCBSD and /Programs*.

The PBI's are now located at their own website. Clicking on the icon *Download PBIs* brings you there (*http://www.pbidir.com*). Apart from these, there is a collection of non-official PBI's that are available from *http://www.pbis.in*. The non-official PBI's provide installers for free closed source software or components (like codecs) or for proprietary software like Dreamweaver and Photoshop. They do not provide the software (since that would be illegal), but they provide an installer based on wine to go along with the original executables of the programs.

The official PBI website has various categories of software. You can surf on the website and search for the program you need. Then you select one of the mirrors to download the PBI. Once installed, the PBI installer launches automatically.

The concept is simple, but extremely powerful. One example: as yet there is not native BSD support for Flash 9 and that complicates matters for a group of users. On the PBI website you will find an installer for Firefox 2.0.0.7 with Flash 9 support. This is achieved by using the Windows version of Firefox along with Wine and integrate Flash 9 in that way.

There are a few downsides. At present there is only a limited collection of PBI's

though more than enough for a large group of users. Secondly, it is not flawless yet. Some PBI's were not designed to work with the 1.4.x release of PC-BSD and then fail. In various cases not all mirrors were operational resulting in error messages. One final issue is the lack of information when installing large PBI's of 100s of Mb's. As a user you get the impression that the installation has failed or is not progressing. However, it just takes a long time to process the package and it would be nice to see this in the progress indicator.

Where to go from here?

Both DesktopBSD and PC-BSD can be considered good open source desktops, though there are still various issues and rough edges that need to be ironed out before a large scale adoption can take place.

The PBI system emulates the experience of Windows users and that makes it the greatest asset in promoting PC-BSD to novice users. This does require some improvements to the current PBI system. The PC-BSD team understands this issue and work is being done on the PBI Build Server. The PBI Build Server *tracks* changes in the FreeBSD ports tree and generates a new PBI when a port has been updated. This way the PBI maintainers only have to take care of the configuration modules for the build process.

The second issue that requires attention is managing and upgrading the PBI's. The current system does indicate an update is available, but the user is then routed to the PBI site in order to download the PBI manually. This would not be much of a problem with a few packages, but it does become a nuisance with multiple packages.

Thirdly, PC-BSD does not have a graphical fronted for easy management of packages and ports. Once the PBI Build Server runs full steam it would not be necessary to revert to packages and ports to get new software and an update of the operating system is taking care of by the Online Update Manager. Whether this is a good thing or not depends on the level of expertise of the user, but considering the target audience there is no immediate need for such a fronted. However, that being so, it might be a good idea not even to mention ports and packages in the documentation for the end user.

The DesktopBSD tools are quite mature. The Package Manager interface could use a polish and could -in places- provide more feedback about the status of installs, especially when they fail. The second thing that requires some attention is the KDE desktop, especially the menu. The current release, for instance, has two entries called *Settings* and that is confusing.

Conclusions

PC-BSD and DesktopBSD provide two solid candidates for the open source desktop. DesktopBSD might have a stronger appeal to Linux users that wish to try out BSD, whereas PC-BSD should appeal more to Windows users. Both of them can be considered to be on a par with Ubuntu, Fedora and OpenSuse. Anyone looking into a good open source desktop should give them a try.

Interview with Matt Olander

On October 10, 2006 it was announced that PC-BSD was acquired by iXsystems. It was not the first time to see corporate interest in developing open source desktops. When we look at Linux we see Red Hat (Fedora), Novell (Suse) and Canonical (Ubuntu). Maybe not surprisingly, the more successful open source desktops are the ones that can combine strong developer communities and a solid corporate involvement. Matt Olander is Chief Technology Officer with iXsystems.

Q: Why did iXsystems decide to acquire PC-BSD?

A: Let's first say that iXsystems is a company with a long history of using FreeBSD and other *BSD's and offering FreeBSD professional services and support. We understand the strength of *BSD and its developer communities and have build our business around it. At iXsystems our business is to deliver open source friendly rackmount servers and storage equipment. A large amount consists of shipping servers that are installed with, configured and optimized for the FreeBSD operating system.

With this in mind it makes sense to support the development of PC-BSD and be actively involved in it. PC-BSD is rapidly encouraging the adoption of FreeBSD among new users. This will undoubtedly lead to a larger FreeBSD community. A larger user base makes it more attractive for more vendors to provide better support. All of this is good for iXsystems, FreeBSD and the open source world in general. For this reason we consider it a worthwhile endeavor to provide corporate support, financially, with equipment and with other resources.

Q: What plans does iXsystems have with PC-BSD?

A: Deploying PC-BSD in companies and organizations, both on a small or large scale, requires solid enterprise-class support. That will be one of the things that iXsystems is to offer this level of support. We will also offer packaged versions of PC-BSD for a nominal and we are working on a range of workstations and laptops that run FreeBSD/PC-BSD.

Q: Can you tell us a little bit more about the SpreadBSD campaign?

A: The SpreadBSD campaign is a fun undertaking designed to increase the visibility of several *BSD projects like FreeBSD and PC-BSD. We plan to generate more community involvement and add descriptive landing pages for other FreeBSD related projects like DesktopBSD, FreeNAS, pfSense, m0n0wall etc. as well. Community members can earn points via click-throughs and eventually those points can be traded for gear at the FreeBSD Mall. Gerard van Essen is really putting his heart behind the SpreadBSD campaign.

PC-BSD overview

Svetoslav P. Chukov

Everybody wants a stable and secure operating system and wants it at the cheapest possible price. In most cases it is GNU/Linux. But what about when you want something different, something not quite like GNU/Linux? Then you would do well to have a look at FreeBSD. In this article I will take an overview of a certain FreeBSD distribution called PC-BSD.

PC-BSD is a distribution intended to be useful for ordinary desktop users. It is different from FreeBSD, which targets highly skilled users and administrators. PC-BSD is a user-friendly distribution with nice graphical installation and tools.

Features

An easy and useful graphical installation:

- Suitable for desktop users, administrators, and highly experienced users as well
- Very good package system
- Easy-to-use
- Full desktop

That is what you will notice on your screen when you get PC-BSD booting. But before this can happen, you have to install it.

I think you will be surprised by the graphics you will find there, desktop themes, boot screens, login themes. They are all ready-to-go with PC-BSD. Most of the distributions out there just build a standard selection of packages and distribute them with standard logos and graphics, but PC-BSD does not stop there...

When I say *Easy and useful graphical installation*, I mean it: I have installed PC-BSD on my computer in just 20 minutes and 10 mouse clicks. This is very valuable for novice users that have trouble with full-featured installations with many options. PC-BSD is for people who would like to have a complete and powerful FreeBSD distribution in 20 minutes. The possibilities of usage range from servers to home desktops full of software to tiny old computers with only a web browser even to a thin and simple client.

Installation

Before we can start using our system we have to install it. I will be using basic CD/DVD boot. When the boot process starts up a graphical interface appears and the installation process begins. It is very easy to manage. Users familiar with other graphical installation programs will be satisfied to see how you can install the whole operating system with just a few mouse clicks. The main specifics you have to tell the installer are :

- System language, keyboard layout, and timezone
- General Installation choice (fresh install or update/repair)
- User accounts with passwords, console shell type, and your full user name
- Drive selection and partitioning
- System components

The system language, keyboard, and timezone are just the first steps to have PC-BSD booting on your computer. There are many languages available and this is significant feature, which allows for wide support for people around the world.

The next step is to choose your type of installation: fresh install, update, or rescue. When you click on the install option, a user accounts dialog appears asking for user names and passwords.

This is not necessarily a vital part of the installation, but it is mandatory. You can see from the above screenshots how easy and fast is to install and manage your system with PC-BSD. If you have an emergency situation and want a working system as soon as possible aspect rescue mode is the way to go. This is the most important feature of PC-BSD – easy and fast installation, a great packaging system, a wide range of available applications, and well preconfigured desktop.

After performing these steps, all you need to do is click next and choose what applications you need on your desktop. You can install extra packages at any time.

This screen will be the first graphic you will see after you log on to your system. That means PC-BSD is installed and ready to use!

Tools and components

Your system is now ready to work. First, take a look at what applications are there available. Everything in the KDE desktop panel is catalogued into categories and subcategories.

Usually there are lot of applications and options and some less experienced users are confused by this array of features, but PC-BSD has a well-organized panel to compensate. Very few people pay attention to how the panel menu is organized, but this is a very important aspect.

Many distributions simply drop all applications into some menu and it becomes difficult even for an experienced user to find out where the application, for which he is looking is placed. If you have previously tried before some of these distributions you will know what I mean.

So, a well organized menu is important for general usability.

Every user wants to have a one-click destination and that idea is realized by the PC-BSD developers. Another aspect you can notice is the different desktop theme. It is styled in the tone and spirit of the distribution as a whole.

That may not be a vital factor, but an important condition if you want to maximize the users' satisfaction. There is also good variety of desktop applications. Almost all the popular names are available as well as lesser known, but very useful and well-designed ones.

You can run Compiz-Fusion, manage software, start some office work, or run particular a server service. All these options are at most just a few steps away from use.

Figure 1. PC–BSD is booting

Compiz-Fusion is installed and preconfigured. There is also a Compiz-Fusion configuration utility from which you can further customize settings. You can choose from a system components menu. An update manager is also available, and extra optional packages can be installed via the *Add/Remove Software* tool and if you miss one of your favourite apps it is a breeze to add it.

As you will see that installation is really easy because of the PBI package manager.

Package management

PBI is package format that empowers PC-BSD. PBI stands for PC-BSD Installer. Some people may also call it Push-Button Installer. Programs under PBI are completely self-contained and self-installing, in a graphical format. A PBI file contains all the executable and data files and libraries necessary for the installed program to properly function, without dealing with broken dependencies and system incompatibilities. Here are some of the features of PBI:

- Graphical installation process
- Scripting support for control over the installation process
- Corruption detection (ensures the package integrity)

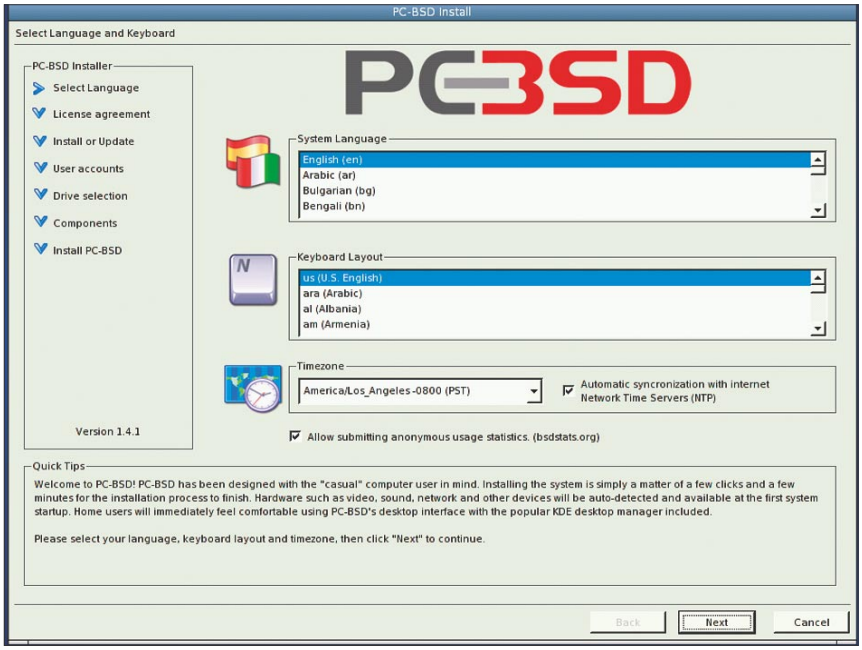


Figure 2. Installation language settings

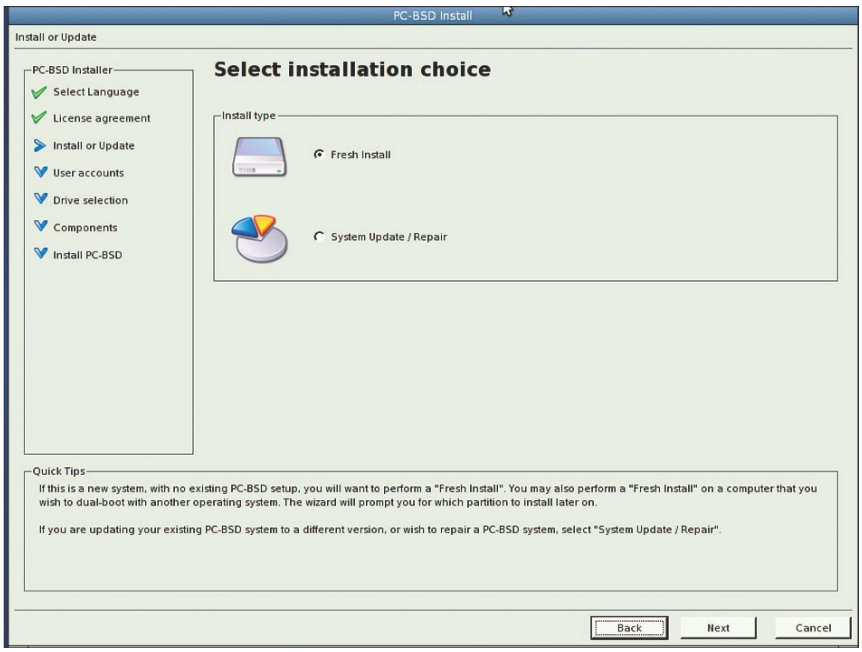


Figure 3. Type of installation

- Program Error Detection (takes action when a binary program fails)

The package system is an important and significant part of PC-BSD, but you need some tools to manage it, those being the *PBI Add/Remove* software tool and the online update manager.

With these tools you can manage and update the packages without knowing anything about the package itself. Let's take a look about how to add and remove applications. Click on *Panel->Settings-> Software & Updates->Add/Remove Software*.

A nice graphical window will appear with list of installed and available components. To install an application just click on *System Components* tab and then mark the desired application. After that the *Install* button will do the work for you.

All neccessary packages will automatically be calculated and you will be asked to load a CD with the needed target package. For that reason I suggest to download not only CD 1, but also CD 2 before you consider self-managing of the components. CD 1 of the set contains the basic system components while CD 2 contains additional software you may want to use.

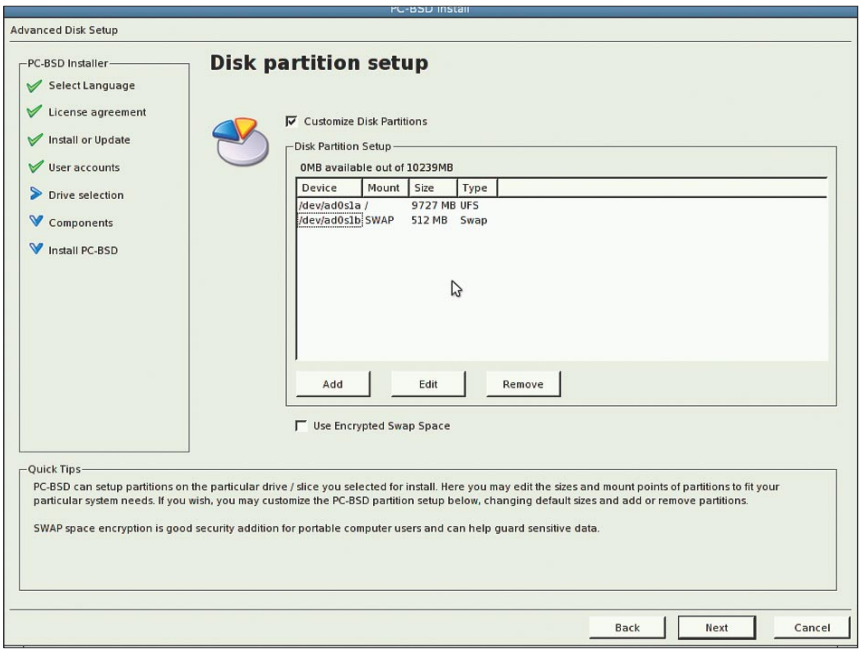


Figure 4. Disc partitioning

Figure 5. User accounts

Therefore, it is very useful to have both of them downloaded. As you can see, although this distribution is not as popular as other GNU/Linux and *BSD distros, there is no lack of applications, packages, or usability. In fact it is suitable for both basic as well as highly skilled users.

For example, if you want to run a workstation and want it fast, you will not have a problem for making that happen in 30 minutes (or even less than 30 minutes, but it depends on the computer performance).

Removal of packages is almost the same way as the installation. Click on the *Installed PBIs* and just choose a component, then click on the *Remove* button. It will automatically uninstall the appropriate packages. The entire process is handled by the system and the only thing the user need to do is to sit back and relax while the process completes.

But PBI packages can be easily managed via console tools too. You can easily remove a PBI from the command line in the following way:

```
PBIdelete -remove pbi_path
```

where `pbi_path` is the name of the directory that holds the target PBI. For example, if you want to remove a PBI that is under */MyPrograms/myApplication* you can do it like this:

```
PBIdelete -remove myApplication
```

This will automatically uninstall it from your system. You should see the following lines after issuing the command:

```
Removing PBIPackage
Running custom removal script.
Removing Directory.
Cleaning PBI list.
```

PBIPackage was removed successfully! To install a PBI try this:

```
./PBIPackage -text
```

Then you should see something like this:

```
Installing PBIPackage
Running pre-install script...
Running install script...
LAUNCHCLOSE: /usr/local/bin/
myapplication
Installation complete!
```

Let me explain something about PBI package management. PBI is easy to manage, but that is not its only benefit. PBI can run custom pre-install, install, and removal scripts that makes the installation and removal process advanced and customizable.

If you have some experience with the MS Windows operating system, you will find some similarities between PBI and the Windows installer.

In fact PBI is very close to the Windows mode of thought. All the necessary files for proper working of the package, are built into the package itself. So, one can download a completely independent package and install it without problems or troubles with dependencies. But this does bring with it a negative – with all the necessary dependencies included in the package.

Do not be surprised if you have a several times larger for the corresponding program with a different package system. This difference is obvious if you have a look at RPMs and compare their size to PBIs.

The next part of software management is the update. There is another tool for this and it also has a simplified yet powerful interface. From this tool you can make updates, but you can also specify parameters for future updates which makes any upcoming adjustments to the system very easy. There are options for daily and weekly updates and the time for the future packages. At this specified time the update will be conducted automatically.

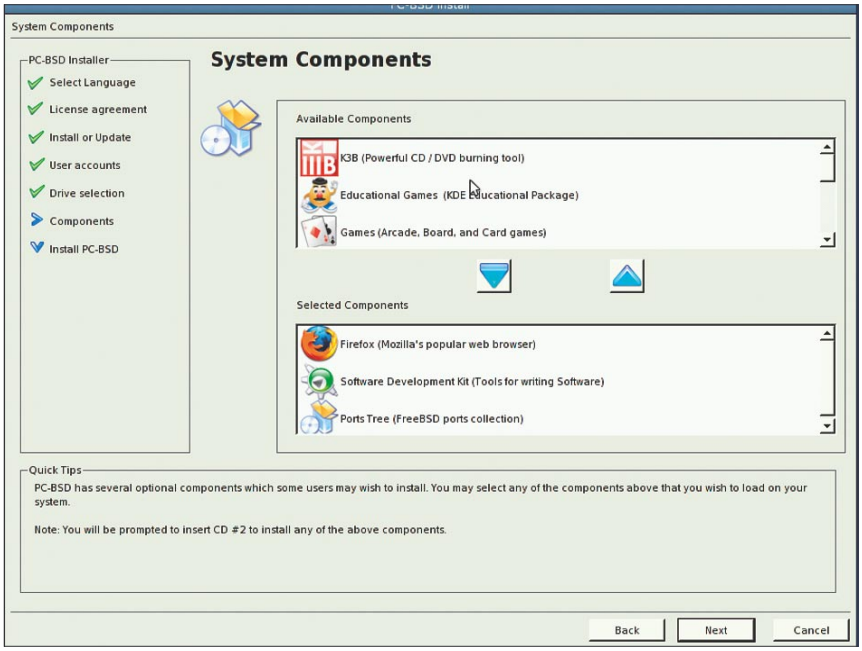


Figure 6. Systems components

Figure 7. Loading the desktop

Services management

PC-BSD offers a very comfortable tool for service administration. No knowledge of the back-end configuration is needed. The service manager is placed on the panel at *Settings->System Administration->Services* manager.

If you want to stop a particular system service, you just have to mark the service name and then click on *Stop*. Service will be stopped and its status will be changed to *Stopped*.

If you have to start a service do it in the same way. Just click on the service and then clicking *Start* will do the work for you. There are the additional buttons *Enable* and *Disable* for each service and you can use them in case you want to disable the running at boot time of this service at all. As you can see, this tool is very simple but can save you plenty of time.

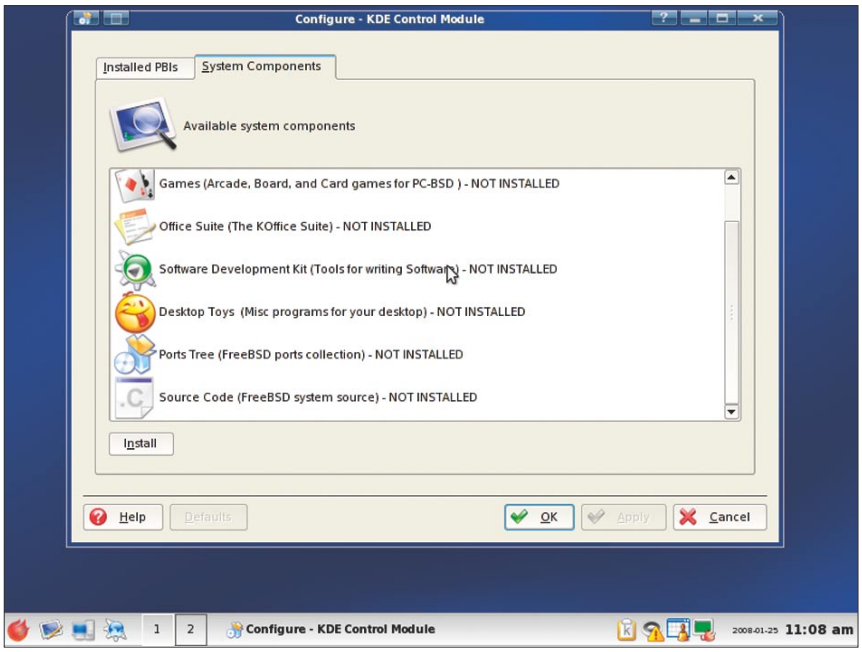


Figure 8. Software available on the system

System management

This tool is placed under *Settings->System administration->System manager*. As you can tell by its name, this is the tool that controls your system. It is responsible for keeping your system working properly. There are four tabs in this version and every tab has some specific job.

At the first tab one can see the PC-BSD logo as well as some basic system information and the button *Generate*. This button can export a system diagnostic sheet about your computer, so, if you would like some information in text format but do not quite know how to extract it, this is the tool you need.

The second tab is called *Kernel*. It gives you information about your booting kernel, time for boot delay, and the ATAPI DMA mode. Basically there are onlu a few options here.

Be careful with the *Selected kernel* option, because if you choose a kernel that is not right for your hardware, you may have unexpected results or even a non-booting system.

The third tab is about tasks like *Fetch system source* and *Fetch ports*. Basically the first option will perform a full update of your system, so it will take some time depending on your internet connection. I always stay up-to-date and I recommend the same to you. Make such updates at a regular basis so at every moment you can have an updated system.

The second option is about the update of the ports. This option will download the ports tree to a local directory for later work. The greatest benefit of this option is that after the ports tree is downloaded, you can compile and install whichever application you want. Many people still prefer to install their own compiled applications because of the fine tuning it allows. Optimization for performance, memory consumption and CPU instructions are just a few of all the possible enhacement. When used by an advanced user, manual compilation with the proper compiler options can improve the produced binary code in several ways. These make the resulting application faster and more responsive.

The last tab is about miscellaneous things like the splash boot image and languages. It is not a vital tab, but it is interesting to note how many languages are available. This is a demonstration how many supporters has PC-BSD around the world. And the amount is pretty good.

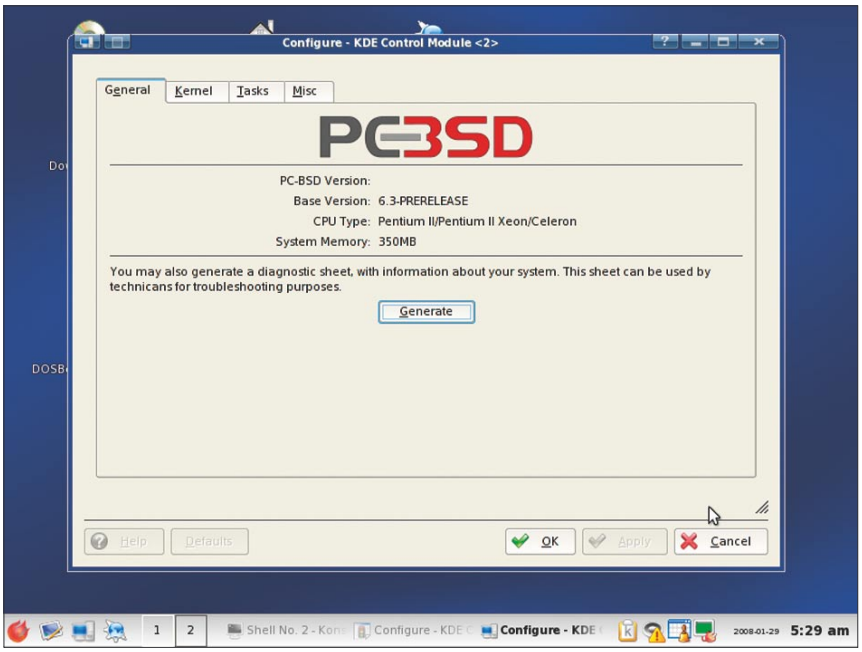


Figure 9. Configuration

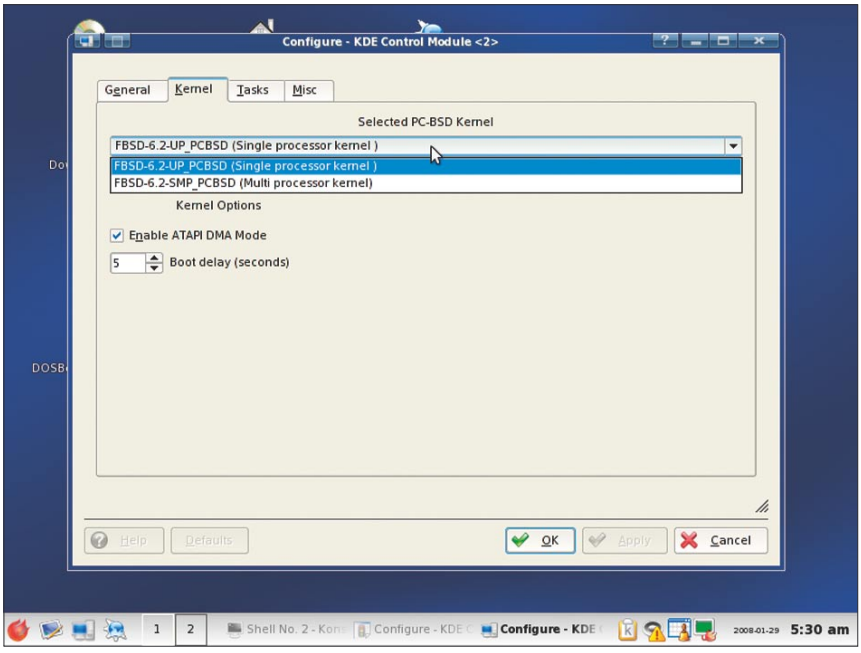


Figure 10. Kernel options

Summary

After that short exposition of features, it is time to summarize the purpose of this distribution, how the user works and benefits from it, and why somebody would have need to use it.

This distribution is intended to have a comfortable desktop interface. It targets the ordinary user, who does not nitpick about the kernel or the console advantages, but cares about the desktop usability and the user-friendly interface. The installation and system component management tools are built around are built around for ease of use. Despite these aspects, PC-BSD is not only for desktop users but administrators and highly-skilled gurus as well. So what exactly could be so interesting to a FreeBSD guru? To put it simply the FreeBSD itself. PC-BSD is built on top of FreeBSD, so everything that is available for the one is available for the other.

This interoperability means you can use the same the base system apps. But that is not all. The package manage-

ment, graphical installation, and update manager are the heart of the distribution and they are valuable for administrators too. Many administrators want to run a server with requirement of security, stability, easy to updating and managing. And these requirements fit exactly to PC-BSD.

As far as I understand the operating systems and the open source movement, every product has its own market, and that principle applies to an open-source or closed-source product. When I took a closer look at the reviewed distribution I noticed that it is very suitable for both the desktop and server markets. The *ordinary* user wants to have nice desktop without BSOD (*Blue Screen of Death*) and as easy as a using experience possible, including basic configuration and software management. The server admin wants to have a stable and secure operating system with many console tools. As a derived distribution, PC-BSD takes all the advantages and disadvantages of its parent. And the parent is the rock-solid FreeBSD, itself in BSD-UNIX. So, the people interested in administering the respective parent systems could successfully work with the children too.

PC-BSD is a very good distribution suitable for both the desktop and server arenas. It has a very useful installation manager with options for encrypted SWAP space and easy system packages manipulation. It also features easy-to-use but advanced and powerful package management.Another advantage is support for many languages that will help you if you are not a native English speaker. The full desktop application stack with the robust K Desktop Environment is great for novice and unexperienced users. Another feature is the great security resulted from the firewall based on the OpenBSD's Packet Filter which is known as world class firewall solution.

One of Packet Filter's important features is a *stateful inspection*. Stateful inspection refers to PF's ability to track the state, or progress, of a network connection. There is a state table used for storing information about each connection. So, based on this state table PF can quickly determine if a packet belongs to an already established connection. If it does, it is passed through the firewall without going through ruleset evaluation.

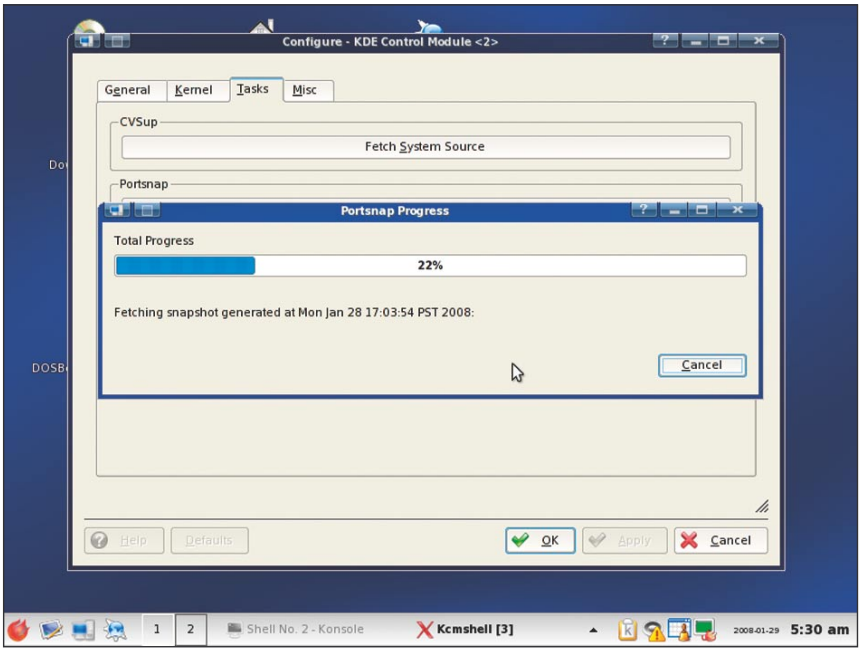


Figure 11. Update and system tasks

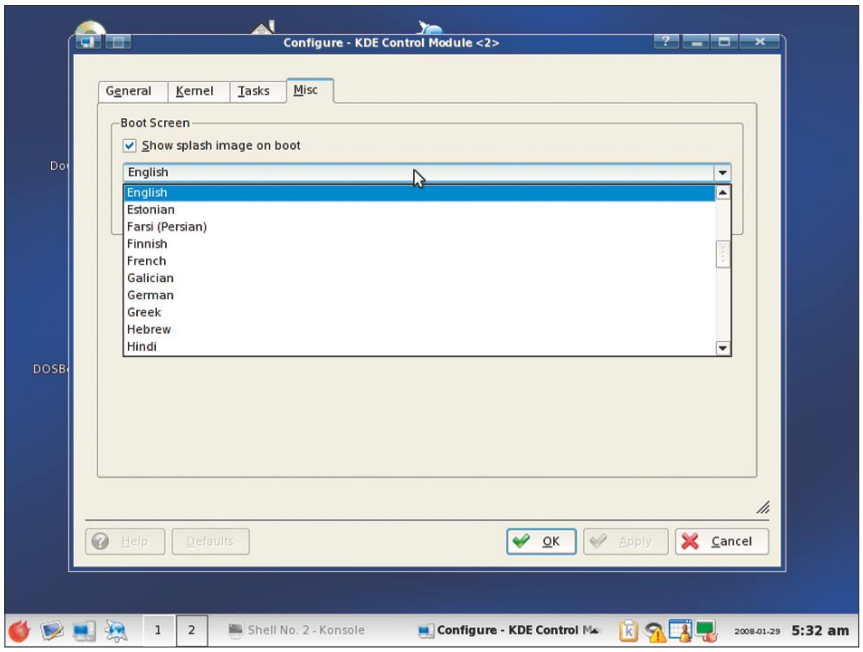


Figure 12. Additional options for the boot screen

Resources

One of the most incredible things about his distro are all the resources available all around the Internet. Too many open source projects lack up-to-date documentation and support from the community. PC-BSD, however, has good documentation and some very interesting sites in the network. You are probable asking where to get all these PBIs we discussed earlier? The solution is all within a single resource – *http://pbidir.com/*. This site has many applications packaged in PBIs ready to download. Also take a look at some of the more interesting sites:

- Main distribution site: *http://pcbsd.org/*
- General free community support page: *http://www.pcbsd.org/content/view/15/29/*
- Community forums: *http://forums.pcbsd.org/*
- Professional paid support: *http://www.ixsystems.com/*
- BSD certification: *http://www.pcbsd.org/content/view/17/31/*

Sguil 0.7.0 on FreeBSD 7.0

Richard Bejtlich

Sguil (www.sguil.net) is an open source suite for performing Network Security Monitoring (NSM). NSM is the collection, analysis, and escalation of [network-based] indications and warnings to detect and respond to intrusions. The author has been using Sguil on FreeBSD since Sguil's developer, Robert "Bamm" Visscher, created the application in 2001.

At the time of writing this article, Sguil 0.6.1 is the stable version. Sguil as provided by the Concurrent Version System (CVS) repository at Sourceforge is considered Sguil 0.7.0. Also at the time of writing, FreeBSD 7.0-BETA4 is available. Therefore, this article demonstrates how the author deployed Sguil 0.7.0 from CVS in late December 2007 on FreeBSD 7.0-BETA4. The author expects the method shown here to remain the same for released versions of each product. The main exception could be bug fixes to Sguil that happen before 0.7.0 is released.

Please note this article demonstrates one way to install Sguil on FreeBSD. As exemplified by the excellent Sguil on Red Hat HOWTO (*http://www.vorant.com/nsmwiki/index.php?title=Sguil_on_RedHat_HOWTO*), installing Sguil is a complicated and usually personal business. The steps in this document reflect the author's biases, but the net result is a working Sguil sensor and server on a single platform. This deployment is suitable for a test location or a small production site. Sensors monitoring heavier loads should separate the inspection functions from the database and server functions.

Groundwork

My personal preference is to provide users with the Bash shell. Therefore, I begin by adding that shell. I start by setting the `PACKAGESITE` variable to ensure I am using the packages I expect, namely those from FreeBSD 7.0 `RELEASE`.

```
# setenv PACKAGESITE
ftp://ftp.freebsd.org/pub/FreeBSD/ports/i386/
packages-7.0-release/Latest/
# pkg_add -r bash
```

Be sure to add two users in addition to having root's account. You need users *analyst* and *sguil* to use Sguil as explained here. User analyst is in the wheel group and can switch to root. User sguil is not in the wheel group. Wherever possible we run applications as user sguil. Although this guide shares user sguil among many applications, feel free to configure your system differently. In fact, please recognize that this article is not a guide to building a hardened Sguil system. This is only a configuration document to get all of Sguil's components working properly. With Bash installed, my user accounts are free to change their shells:

```
$ chsh -s /usr/local/bin/bash
```

Next, ensure that the sensor has an `/etc/hosts` entry other than the default *localhost*. In this article the sensor name is *taosecurity*:

```
127.0.0.1 taosecurity taosecurity.taosecurity.com
```

Finally, if you plan to use Snort (version 2.8.0.1 at the time of writing), you will probably want to register for the Sourcefire *Vulnerability Research Team* (VRT) rule set. Download *snortrules-snapshot-CURRENT.tar.gz* from *www.snort.org* into the `/tmp` directory for now. With that groundwork done I turn to installing Sguil sensor components.

Sensor Installation Preparation

When not indicated otherwise, I perform all actions as root. (Convenience here wins over being more granular. Please adjust if you feel the need.) The first step is to check out Sguil from CVS.

```
mkdir -p /usr/local/src
cd /usr/local/src
```

```
cvs -d:pserver:anonymous@sguil.cvs.sourceforge.net:/cvsroot/sguil login
cvs -z3 -d:pserver:anonymous@sguil.cvs.sourceforge.net:/cvsroot/sguil co -P sguil
```

Now set variables that will make life easier while installing these components. Precede each of the following with the *set* statement at root's tcsh command line.

```
SENSOR=`hostname -s`
SGUIL=sguil
ETCNSM=/usr/local/etc/nsm
SENSORDIR=/usr/local/src/$SGUIL/sensor
SERVERDIR=/usr/local/src/$SGUIL/server
SNORT=2.8.0.1
SNORTETC=/usr/local/src/snort-$SNORT/etc
```

Create these directories needed by Sguil sensor components and set ownership of certain directories to user sguil (Listing 1). Add the following packages. Note the tcltls package was custom-built by the author because the package needed was either not available or was not current enough at the time of writing.

```
pkg_add -r tcl84
pkg_add -r http://www.bejtlich.net/tcltls-1.5.0.tbz
pkg_add -r sancp
pkg_add -r tclX
pkg_add -r pcre
pkg_add -r libdnet
pkg_add -r automake19
pkg_add -r libtool
```

Copy the following configuration files into the proper location.

```
cp $SENSORDIR/snort_agent.conf $ETCNSM
cp $SENSORDIR/pads_agent.conf $ETCNSM
cp $SENSORDIR/pcap_agent.conf $ETCNSM
cp $SENSORDIR/sancp_agent.conf $ETCNSM
cp $SENSORDIR/sancp/sancp.conf $ETCNSM
```

Finally, create a link for tclsh, which is required by Sguil.

```
ln -s /usr/local/bin/tclsh8.4 /usr/local/bin/tclsh
```

With these steps completed it is time to install Snort.

Snort Installation

The author prefers installing Snort from source. At the time of writing the FreeBSD port was Snort 2.7.x, while 2.8.0.1 is available.

```
cd /usr/local/src/
fetch http://www.snort.org/dl/current/snort-$SNORT.tar.gz
tar -xzf snort-$SNORT.tar.gz
cd snort-$SNORT
mkdir -p /usr/local/snort-$SNORT
./configure --enable-dynamicplugin --enable-rulestate \
--enable-perfprofiling --enable-timestats  \
--enable-ppm --prefix=/usr/local/snort-$SNORT
make install
ln -s /usr/local/snort -$SNORT/bin/snort /usr/local/bin/snort
touch /nsm/taosecurity/snort.stats
```

If the installation was successful, output like the following should appear (Listing 2). With Snort at least starting properly, turn your attention to putting the configuration files in the proper locations.

```
cp $SNORTETC/classification.config $ETCNSM
cp $SNORTETC/gen-msg.map $ETCNSM
cp $SNORTETC/reference.config $ETCNSM
```

Listing 1. Directory Creation

```
mkdir -p /nsm/$SENSOR/dailylogs
mkdir -p /nsm/$SENSOR/portscans
mkdir -p /nsm/$SENSOR/sancp
mkdir -p /nsm/rules/$SENSOR
mkdir -p /var/log/snort
mkdir -p /nsm/rules/taosecurity/preproc_rules
mkdir -p /usr/local/etc/nsm
mkdir -p /nsm/archive
mkdir -p /nsm/load
chown -R sguil:sguil /nsm
chown -R sguil:sguil /var/log/snort
chown -R sguil:sguil $ETCNSM
```

Listing 2. Testing Snort

```
taosecurity# rehash
taosecurity# snort -V
,,_ -*>Snort! <*-> o" )~Version 2.8.0.1(Build 72) FreeBSD
> ''''  By Martin Roesch & The Snort Team: http://www.snort.org/team.html > (C)
Copyright 1998-2007 Sourcefire Inc., et al. > Using PCRE version: 7.4 2007-09-21
```

Listing 3. Barnyard Installation

```
cd/usr/local/src
fetch http://superb-east.dl.sourceforge.net/sourceforge/barnyard/barnyard-0.2.0.tar.gz
tar -xzf barnyard-0.2.0.tar.gz
cd barnyard-0.2.0
cp $SENSORDIR/barnyard_mods/configure.in .
cp $SENSORDIR/barnyard_mods/op_sguil.c ./src/output-plugins/
cp $SENSORDIR/barnyard_mods/op_sguil.h ./src/output-plugins/
cp $SENSORDIR/barnyard_mods/op_plugbase.c.patch ./src/output-plugins/
cp etc/barnyard.conf $ETCNSM
cd src/output-plugins/
patch op_plugbase.c < op_plugbase.c.patch
cd ../..
aclocal
autoheader
automake --add-missing --copy
autoconf
./configure --enable-tcl --with-tcl=/usr/local/lib/tcl8.4
make
make install
```

```
cp $SNORTETC/threshold.conf $ETCNSM
cp $SNORTETC/unicode.map $ETCNSM
cp $SNORTETC/snort.conf $ETCNSM/
snort.conf.$SNORT
ln -s $ETCNSM/snort.conf.$SNORT
$ETCNSM/snort.conf
```

Now it is time for rules.

```
cd /nsm
fetch http://www.emergingthreats.net/
emerging.rules.tar.gz
tar -xzf emerging.rules.tar.gz
cp /tmp/snortrules-snapshot-
CURRENT.tar.gz .
tar -xzf snortrules-snapshot-
CURRENT.tar.gz
cp /nsm/rules/* /nsm/rules/$SENSOR
```

We only use Oinkmaster here to create a sid-msg.map for the rules we have downloaded:

```
cd /usr/local/src
fetch http://superb-
east.dl.sourceforge.net/
sourceforge/oinkmaster/oinkmaster-
2.0.tar.gz
tar -xzf oinkmaster-2.0.tar.gz
```

Listing 4. Daemonlogger

```
cd /usr/local/src
fetch http://www.snort.org/
users/roesch/code/daemonlogger-
1.0.1.tar.gz
tar -xzf daemonlogger-1.0.1.tar.gz
cd daemonlogger-1.0.1
mkdir /usr/local/daemonlogger-1.0.1
./configure --prefix=/usr/local/
daemonlogger-1.0.1
make
make install
ln -s /usr/local/daemonlogger-
1.0.1/bin/daemonlogger
/usr/local/bin/daemonlogger
cp $SENSORDIR/log_packets.sh /usr/
local/bin
```

Listing 5. Patching Configuration Files

```
tar -xzf sguil-
0.7.0cvs.patches.tar.gz
cd $ETCNSM
patch -p0 < snort.conf.patch.2.8.0.1a
patch -p0 < barnyard.conf.p atch
patch -p0 < snort_agent.conf.patch
patch -p0 < pads_agent.conf.patch
patch -p0 < pcap_agent.conf.patch
patch -p0 < sancp_agent.conf.patch
patch -p0 < sancp.conf.patch
```

```
cd /nsm/rules
/usr/local/src/oinkmaster-2.0/contrib/
create-sidmap.pl> $ETCNSM/sid-msg.map
```

Finally, we install Barnyard to read and process Snort's Unified output.

```
cd /usr/local/src
```

See Listing 3. Testing Barnyard reveals the following.

```
taosecurity# rehash
taosecurity# barnyard -V
Barnyard Version 0.2.0 (Build 32)
```

Before leaving this section change ownership of tiles in the `$ETCNSM` directory.

```
chown sguil:sguil $ETCNSM
```

We have made plenty of progress but we still have other components to install.

PADS and Daemonlogger

Sguil 0.7.0 added native support for Matt Shelton's *Passive Asset Detection System* (PADS). Installation proceeds as follows. Note the use of a modified PADS version to add additional features needed by Sguil.

```
cd /usr/local/src
fetch http://demo.sguil.net/downloads/
pads-1.2-sguil-mods.tar.gz
tar -xzf pads-1.2-sguil-mods.tar.gz
cd pads-1.2-sguil-mods
./configure
make
make install
```

In this section I provide a configuration file you can use as a model for your own work.

```
fetch -o $ETCNSM http://
www.bejtlich.net/pads-taosecurity.conf
```

Next I start PADS to test it, then quickly kill it.

```
pads -c $ETCNSM/pads-taosecurity.conf
ctrl-C
```

Sguil and PADS use a FIFO to communicate. While PADS should be started by root, the pads.fifo does not need to be left as owned be root. I recommend changing ownership to user sguil.

```
chown sguil:sguil /nsm/$SENSOR/
pads.fifo
```

Next we turn to Daemonlogger, which I use here as a full content packet logger. Sguil has traditionally used a second instance of Snort running in packet logger mode for this function. Because Daemonlogger is primarily used to log packets, I use it here in conjunction with Sguil's `log_packets.sh` script (Listing 4). To ensure `log_packets.sh` restarts Daemonlogger every hour (as needed by Sguil and adjustable by the operator), I configure the crontab using a file I provide.

```
fetch http://www.bejtlich.net/log_
packets.sh.crontab
crontab -u root log_packets.sh.crontab
```

To alter `log_packets.sh` to use Daemonlogger, try the following patch. *http://www.bejtlich.net/log_packets.sh.patch* Apply it thus:

```
cd /usr/local/bin
patch -p0 < log_packets.sh.patch
```

We are almost done with the sensor components. Before finishing the sensor components we need to patch the configuration files.

Configuration File Patching

In order to simplify the process of showing how to edit the many configuration files used by Sguil's tools, I created a series of patches. For example, to show how the original snort.conf shipping with Snort 2.8.0.1 is different from the snort.conf.diff I edited, I created a patch.

```
diff -u snort.conf snort.conf.diff >
snort.conf.patch.2.8.0.1
```

The file snort.conf is the original, and snort.conf.diff contains my edits. I collected patches in the archive *http://www.bejtlich.net/sguil-0.7.0cvs.patches.tar.gz* for your use. You are free to edit these patches or the original configuration files, as you desire. The patches contain the following: see Listing 5.

Notice we do not apply the `sguild.conf.patch` yet. That happens later. With these steps done, the last sensor item is to test Snort with its configuration file. Again, run this as root.

```
snort -u sguil -g sguil -U -l /nsm/
taosecurity -m 122 -A none -i le0 -c
/usr/local/etc/nsm/snort.conf -T
```

Assuming Snort exists successfully, we're ready to turn to adding the Sguil components needed for server and database functionality.

Database Installation

Here we add packages needed by Sguil when it the system hosts the Sguil server and MySQL database (Listing 6). With the database configured, we turn to the Sguil daemon sguild.

Sguild

In this short section we will apply a patch to change sguild's configuration file, then make a change to CVS to account for what may be a bug.

```
cd $ETCNSM
cp $SERVERDIR/sguild.conf $ETCNSM
patch -p0 < sguild.conf.patch
```

Sguil from CVS as tested by the author demonstrated a problem that is fixed in the following two patches.

```
cd $SERVERDIR/lib
fetch http://www.bejtlich.net/
SguildLoaderd.tcl.patch
fetch http://www.bejtlich.net/SguildM
ysqlMerge.tcl.patch
```

If these fixes are not applied to Sguil when you try it, fix the code with these two patches.

```
patch -p0 < SguildLoaderd.tcl.patch
patch -p0 < SguildMysqlMerge.tcl.patch
```

Finally, add a user that the client will use to connect to the Sguil server and change ownership of the resulting users file.

```
cd $SERVERDIR
./sguild -c sguild.conf -u $ETCNSM/
sguild.users -adduser sguil
chown sguil:sguil $ETCNSM/sguild.users
```

That's it. We're ready to start activating all of these components.

Start Your Engines

I recommend starting these components as root. First, Snort.

```
snort -u sguil -g sguil -U -l /nsm/
$SENSOR -m 122 -A none -i le0 -c /usr/
local/etc/nsm/snort.conf -D
```

Second, SANCP.

```
sancp -d /nsm/$SENSOR/sancp/ -u sguil
-g sguil -i le0 -c
/usr/local/etc/nsm/sancp.conf -D
```

Third, start Daemonlogger through `log_packets.sh`.

```
/usr/local/bin/log_packets.sh start
```

Fourth, PADS.

```
pads -c /usr/local/etc/nsm/pads-
taosecurity.conf
```

All of the following components can be started as user sguil. I usually run each in a separate instance of screen(1). First, sguild.

```
cd $SERVERDIR
./sguild -c /usr/local/etc/nsm/
sguild.conf -u /usr/local/etc/nsm/
sguild.users
```

Second, `snort_agent.tcl`. Note the change to `$SENSORDIR` here and for all subsequent components.

```
cd $SENSORDIR
./snort_agent.tcl -c /usr/local/etc/
nsm/snort_agent.conf
```

Third, sancp_agent.tcl.

```
./sancp_agent.tcl -c /usr/local/etc/
nsm/sancp_agent.conf
```

Fourth, pads_agent.tcl.

```
./pads_agent.tcl -c /usr/local/etc/
nsm/pads_agent.conf
```

Fifth, pcap_agent.tcl.

```
./pcap_agent.tcl -c /usr/local/etc/
nsm/pcap_agent.conf
```

Sixth, Barnyard startup. Be sure to notice the directory change.

```
cd  /usr/local/etc/nsm
barnyard -c barnyard.conf -d /nsm/
$SENSOR -g gen-msg.map -s sid-msg.map
-f snort.log -w /nsm/$SENSOR/waldo.file
```

That is it! At this point sguild is ready to accept connections from the Sguil client. Installing the Sguil client is a story for another article.

If you have questions, I recommend perusing the archives of the sguil-users mailing list, or send a message to *sguil-users@lists.sourceforge.net*. Good luck!

Listing 6. Database Installation

```
pkg_add -r tcllib
pkg_add -r p0f
pkg_add -r tcpflow
pkg_add -r mysql51-server
pkg_add -r http://www.bejtlich.net/mysqltcl-3.03.tbz
Start the database.
/usr/local/bin/mysql_install_db --user=mysql
/usr/local/bin/mysqld_safe --bind-address=127.0.0.1 --user=mysql &
Next, configure it.
/usr/local/bin/mysql -e "CREATE DATABASE sguildb"
/usr/local/bin/mysql -D sguildb <
/usr/local/src/$SGUIL/server/sql_scripts/create_sguildb.sql
/usr/local/bin/mysql -e "GRANT ALL on sguildb.* to sguil@localhost"
/usr/local/bin/mysql -e "GRANT FILE on *.* to sguil@localhost"
/usr/local/bin/mysql -e "SET password for sguil@localhost=password('sguil
')"
/usr/local/bin/mysql -e "SHOW TABLES" sguildb
/usr/local/bin/mysql -e "SET password for root@localhost=password('r00t')"
/usr/local/bin/mysql --password=r00t -e "delete from user where
Password=''"
mysql
/usr/local/bin/mysql --password=r00t -e "FLUSH PRIVILEGES"
echo "mysql_enable=YES" >> /etc/rc.conf
echo "mysql_args=--bind-address=127.0.0.1" >> /etc/rc.conf
```

How to Dual-Boot Vista with BSD – a step-by-step approach

Jay Kruizenga

Walk the aisles of your local electronics store and you will view rows of PCs with Microsoft Vista pre-installed. For users who choose to install BSD, they are left with few options – either they can wipe Vista from the existing drive or they can setup a dual-boot system giving the user a choice of operating systems to boot.

Up until now this has been a painful task – one that most users avoid. Yes, it can be a nightmare to install BSD alongside Vista. Vista is known not to play right with others – Linux and BSD included. Therein lies the problem. Most users will not even consider a BSD installation for fear that they might injure their Vista drive and be left with nothing. No operating system. No support. No fun. Fortunately, there is now a way of setting up a dual boot system that works – and it is easy.

Now there is EasyBCD 1.7.1. Thankfully it has become easier to install BSD alongside Vista as a dual-boot operating system. Gone are the days of having to mess with command lines running the risk of destroying your entire system.

Developed by NeoSmart Technologies, EasyBCD is an award-winning solution that solves the dual-boot issue. There probably is no easier way to boot BSD right from the Vista bootloader than is possible using EasyBCD.

Still there are a few things that must be accomplished prior to using EasyBCD and that is the purpose of this article.

We are going to show you how to setup Vista so it dual-boots with BSD giving you the choice of what operating system you want to run during boot.

It is assumed that you have a Vista computer or are in the market for one. The following steps prepare your Vista hard drive to accept BSD.

Preparing Vista

Right now your entire drive is devoted to Vista. We need to free up some space for our BSD install. We will do this from inside Vista.

From the Vista *Start menu*, *click->Control panel->System and maintenance*. Scroll to the bottom of the open window until you find *Administrative tools*. Select *Create and format hard disk partitions*. Vista should prompt you with a permission message. Hit *Continue* to bypass.

A new window will appear in a few moments revealing both your *C:* and *D:* Windows drives.

It is the *C: drive* that we are interested in – that is where we will make room for BSD. It should be the largest drive as it contains all of your operating files.

In the center of the window, right-click the *C:* drive and choose *Shrink volume* from the pop-up menu. This will query the volume for potential space that may be freed up.

Once finished, you will see a new window that shows the following information – from top to bottom: total size before shrinking in MB, size of available shrink space in MB, a box in which to enter the amount of space to shrink in MB, and the total size of the Windows drive after shrinking in MB.

The second option – size of available shrink space in MB – is the total space that Vista was able to locate that could be made available for your new BSD install. By default this total size is listed in the third option – the only option which you are able to adjust.

You do not really need a large amount of space for BSD. However, if BSD is to be your default operating system (as it should) you will probably want to utilize 100% of the shrink space being made available by Vista.

Just run with the defaults given. To finalize the freeing up of BSD space click *shrink*.

Figure 1. Use the Vista control panel to Free BSD Space

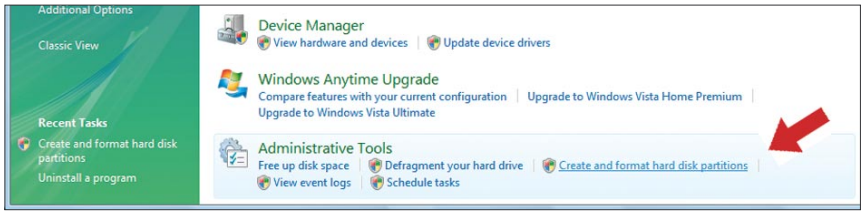


Figure 2. Using Vista we create a new partition

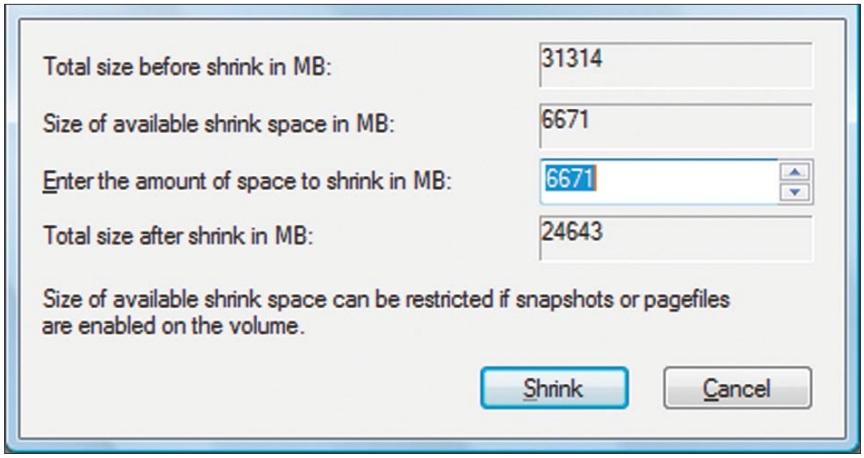


Figure 4. Amount of shrink space varies by system

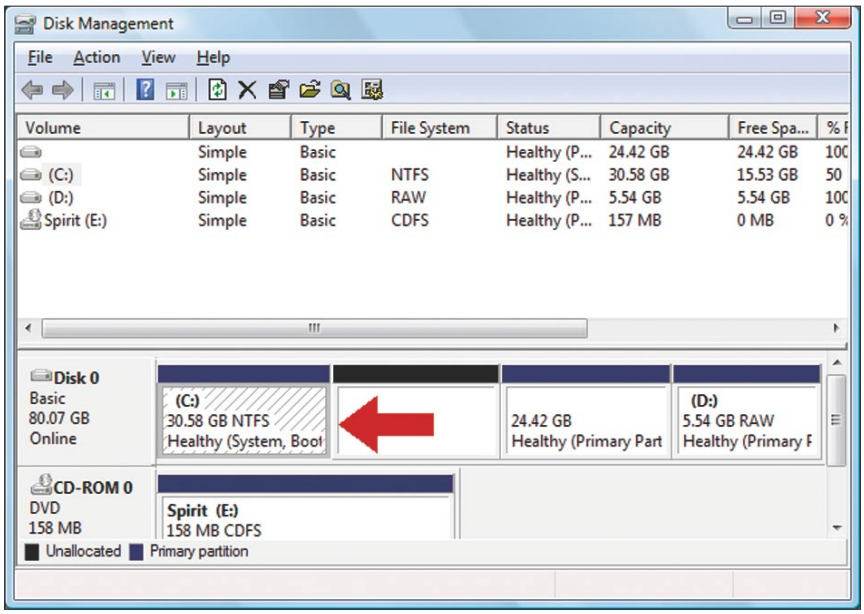


Figure 3. Right click drive to shrink

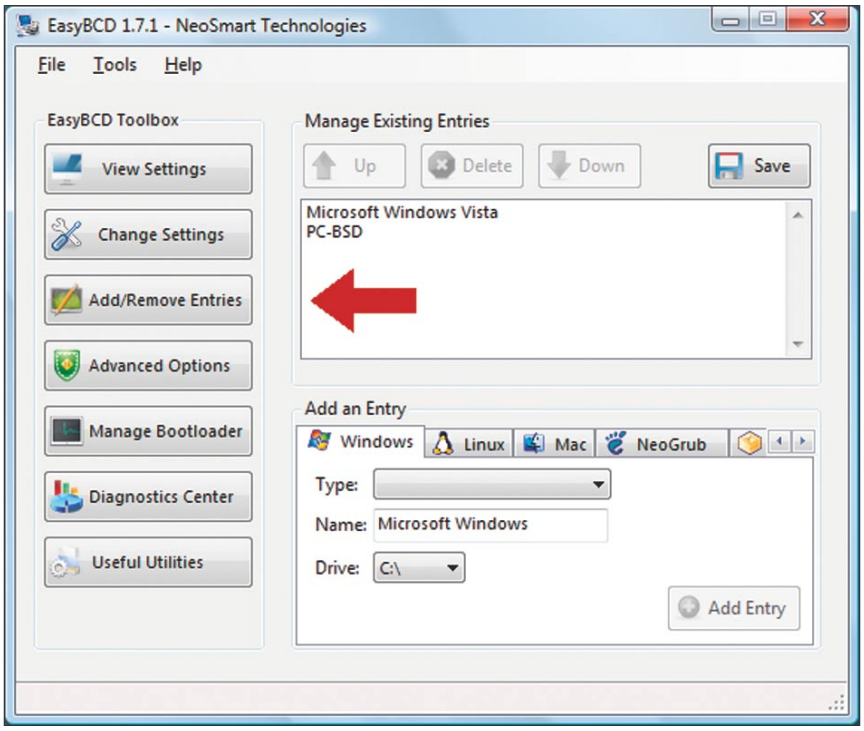


Figure 5. Easybcd makes it easy to add multiple operating systems

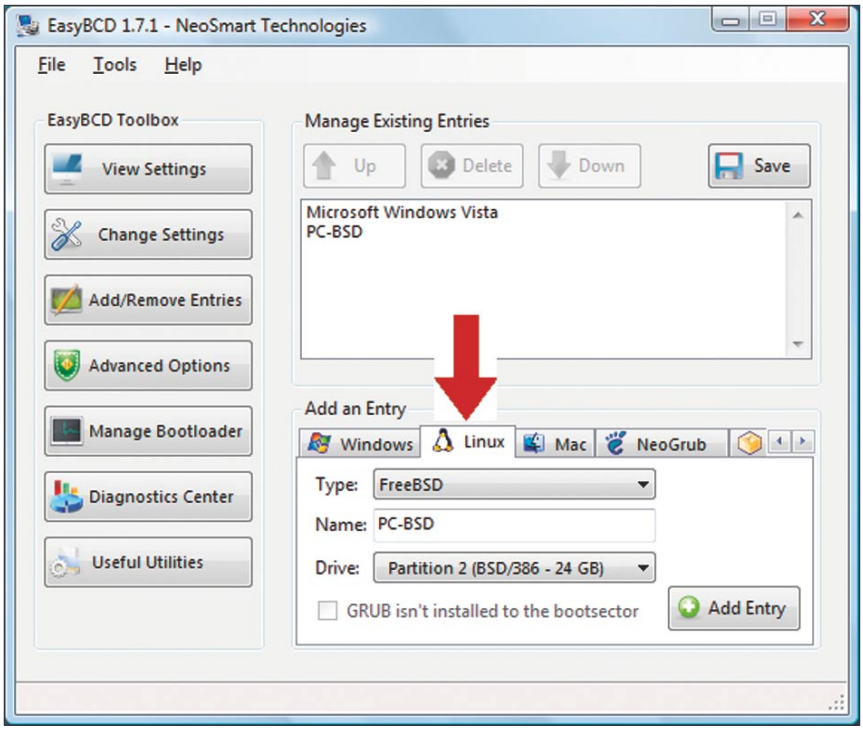


Figure 6. Last step in setting up dual boot bsd/vista system

Vista will take a short amount of time to finish the task. Once finished we are now ready to install BSD. Insert your BSD disk in the drive and restart Windows.

Note: your computer BIOS may be set to boot directly into Windows from the hard drive.

To enter your BIOS settings watch for the key to push when first booting – usually the *F8* function or *Delete* key. Look fast or you will miss it. From within your BIOS, find the settings to adjust boot order from hard drive (set as second) to CD (set as primary). Save settings and restart.

Installation of BSD

Install your flavor of BSD as usual. Be certain to utilize only the free partition. You do not want to take over the entire drive – that would be bad. When prompted to set up the bootloader (if needed), make sure to specify the bootsector of the partition where BSD is being installed and not the MBR of the hard drive. Doing so would write over the Vista bootloader. When install is complete, remove CD from drive and reboot.

You will note that restarting takes you back into Vista. No worries. You are still on the right track.

Now you need to download, install, and setup the EasyBCD program to recognize your BSD partition. From within Vista, point a browser to *http://neosmart.net/dl.php?id=1*. Browse down to find the download link – download to desktop. Install.

Start EasyBCD. Go to the *add/remove entries* screen and pick Linux from the tabs at top.

Pick the FreeBSD bootloader from the drop-down menu and give your entry a user-friendly name (name of your BSD flavor).

The most difficult part of the setup process is in the selection of the correct partition and bootloader.

Note: If you fail to see the option to boot either Vista or BSD at boot this is more than likely the place where you will find the problem.

Press *add entry* and reboot. You should see the Vista bootloader shortly after restarting computer.

Select BSD (or whatever name you called it) and continue to your first-run configuration.

Congratulations, you have successfully setup a dual-boot Vista and BSD drive. EasyBCD is not limited to just Vista.

It is an easy method of setting up the boot for any number of operating systems on the same or different drives.

About the Author

Author is a freelance writer and Linux/BSD advocate living in Grand Rapids, Michigan.

Bringing the *BSD UNIX Community Together in New York City

October 11-12, 2008, Columbia University

NYCBSDCon 2008

nycbsdcon.org

in association with: BUG NEW YORK CITY *BSD USER GROUP

Keep smiling, waste spammers' time

Peter N. M. Hansteen

When you are in the business of building the networks people need and the services they need to run on them, you may also be running a mail service. If you do, you will sooner or later need to deal with spam. This article is about how to waste spammers' time and have a good time while doing it.

Assembling the parts: To take part of the fun and useful things in this article, you need a system with PF, the OpenBSD packet filter. If you are reading this magazine you are likely to be running all important things on a BSD already, and all the fully open source BSDs by now include PF, developed by OpenBSD but also ported to the other BSDs. On OpenBSD, it is *the* packet filter, and if you are running FreeBSD, NetBSD or DragonFlyBSD it is likely to be within easy reach, either as a loadable kernel module or as a kernel compile-time option.

Getting started with PF is surprisingly easy. The official documentation such as the PF FAQ is very comprehensive, but you may be up and running faster if you buy *The Book of PF* [1] or do what about 30,000 others have done before you: Download or browse the free forerunner from *http://home.nuug.no/~peter/pf/*. Or do both, if you like.

Network design issues

A PF setup can be, and to my mind should be, quite unobtrusive. For the activities in this article it does not matter much where you run your PF filtering, as long as it is somewhere in the default path of your incoming SMTP traffic. A gateway with PF is usually an excellent choice, but if it suits your needs better, it is quite feasible to do the filtering needed for this article on the same host your SMTP server runs.

Enter spamd

OpenBSD's spamd, *the spam deferral daemon* (not to be confused with the program with the same name from the SpamAssassin content filtering system), first appeared in OpenBSD 3.3. The original spamd was a tarpitter with a very simple mission in life. Its spamd-setup program would take a list of known bad IP addresses, that is, the IP addresses of machines known to have sent spam recently, and load it into a table. The main spamd program would then have any smtp traffic from hosts in that table redirected to it, and spamd would answer those connections *s-l-o-w-l-y*, by default one byte per second.

A minimal PF config

As man spamd will tell you, the bare minimum to get spamd running in a useful mode on systems with PF version 4.1 or later is:

```
table <spamd-white> persist
no rdr inet proto tcp from <spamd-white> to
any port smtp
rdr pass inet proto tcp from any to any port
smtp -> 127.0.0.1 port spamd
```

This means, essentially, that any smtp traffic from hosts that are not already in the table spamd-white will be redirected to localhost, port spamd, where you have set up the spam deferral daemon spamd to listen for connections. Enabling spamd, on the other hand, is as easy as adding `spamd_flags=""` to your `/etc/rc.conf.local` if you run OpenBSD or `/etc/rc.conf` if you run FreeBSD [2], and starting it with:

```
$ sudo /usr/libexec/spamd
```

or if you are on FreeBSD,

```
$ sudo /usr/local/libexec/spamd
```

It is also worth noting that if you add the `-d` for Debug flag to your spamd flags, spamd will generate slightly more log information, of the type shown in the log excerpts later in this article.

While earlier versions of spamd required a slightly different set of redirection rules and ran in blacklists-only mode by default, spamd from OpenBSD 4.1 onwards runs in greylisting mode by default. Let's have a look at what greylisting means and how it differs from other spam detection techniques before we exlore the finer points of spamd configuration.

Content versus behavior: Greylisting

When the email spam deluge started happening during the late 1990s and early 2000s, observers were quick to note that the messages in at least some cases could be fairly easily classified by looking for certain keywords, and the bulk of the rest fit well in familiar patterns.

Various kinds of content filtering have stayed popular and are the mainstays of almost all proprietary and open source antispam products. Over the years the products have developed from fairly crude substring match mechanisms into multi-level rule based systems that incorporate a number of sophisticated statistical methods. Generally the products are extensively customizeable and some even claim the ability to *learn* based on the users' preferences.

Those sophisticated and even beautiful algorithms do have a downside, however: For each new trick a spam producer chooses to implement, the content filtering becomes incrementally more complex and computationally expensive.

In sharp contrast to the content filtering, which is based on message content, *greylisting* is based on studying spam senders' behavior on the network level. The 2003 paper [3] by Evan Harris noted that the vast majority of spam appeared to be sent by software specifically developed to send spam messages, and those systems typically operated in a *fire and forget* mode, only trying to deliver each message once. The delivery software on real mail servers, however, are proper SMTP implementations, and since the relevant RFCs state that you MUST retry delivery in case you encounter some classes of delivery errors, in almost all cases real mail servers will retry *after a reasonable amount of time*. *Spammers do not retry.* So if we set up our system to say essentially *My admin told me not to talk to strangers* – we should be getting rid of anything the sending end does not consider important enough to retry delivering. The practical implementation is to record for each incoming delivery attempt at least:

- Sender's IP address
- The From: address
- The To: address
- Time of first delivery attempt matching 1) through 3)
- Time delivery of retry will be allowed
- Time to live for the current entry

At the first attempt, the delivery is rejected with temporary error code, typically *451 temporary local problem, try again later*, and the data above is recorded. Any subsequent delivery attempts matching fields 1) through 3) that happen before the time specified in field 5) are essentially ignored, treated to the same temporary error. When a delivery matching fields 1) through 3) is attempted after the specified time, the IP address (or in some implementations, the whole subnet) is *whitelisted*, meaning that any subsequent deliveries from that IP address will be passed on to the mail service.

The first release of OpenBSD's spamd to support greylisting was OpenBSD 3.5. spamd's greylisting implementation operates only on individual IP addresses, and by default sets the minimum time before a delivery attempt passes to 25 minutes, the time to live for a greylist entry to 4 hours, while a whitelisted entry stays in the whitelist for 36 days after the delivery of the last message from that IP address. With a properly configured setup, machines that receive mail from your outgoing mail servers will automatically be whitelisted, too.

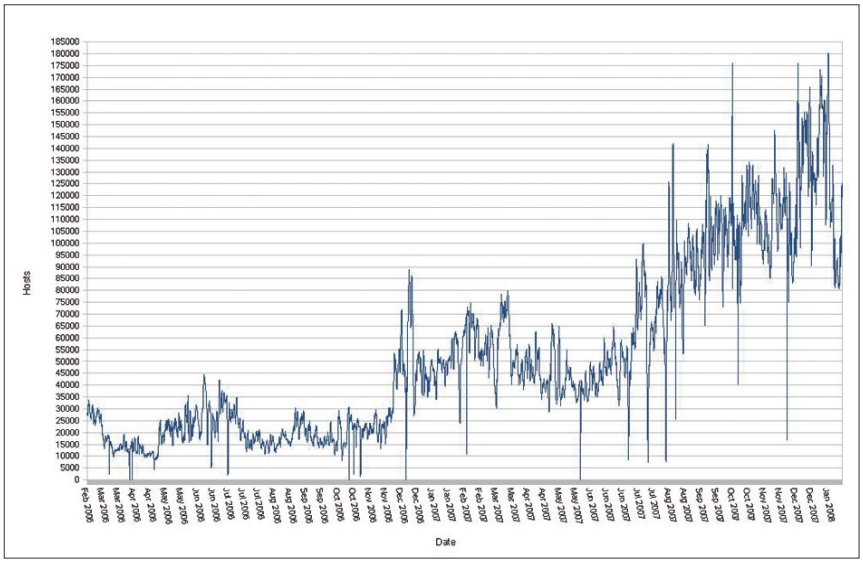


Figure 1. Number of hosts in the uatraps list

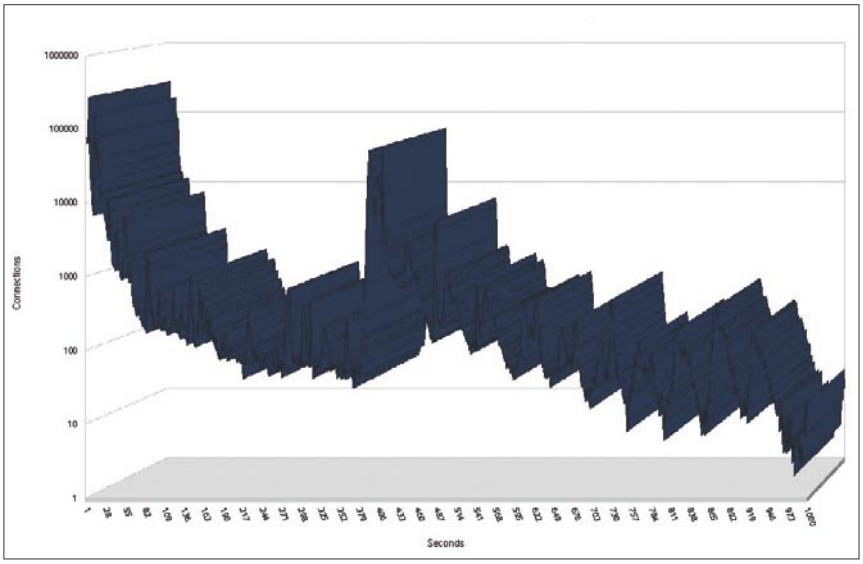


Figure 2. SMTP connections to smamd by connetion length

The great advantage to the greylisting approach is that mail sent from correctly configured mail servers will be let through. New correspondents will experience an initial delay for the first message to get through and their IP address is added to the whitelist. The initial delay will vary depending on a combination of the length of your minimum time before passing and the sender's retry interval. Regular correspondents will find that once they have cleared the initial delay, their IP addresses are kept in the whitelist as long as email contact is a regular affair.

And the technique is amazingly effective in removing spam. 80% to 95% or better reduction in the number of spam messages is frequently cited, but unfortunately only a few reports with actual numbers have been published. An often-cited report is Steve Williams' message on opensd-misc [4], where Steve describes how he helped a proprietary antispam device cope with an unexptected malware attack. He notes quite correctly that the blocked messages were handled without receiving the message body, so their apparently metered bandwidth use was reduced.

Even after more than four years, greylisting remains extremely effective. Implementing greylisting greatly reduces the load on your content filtering systems, but since messages sent by real mail servers will be let through, it will sooner or later also let a small number of unwanted messages through, and unfortunately it does not eliminate the need for content filtering altogether. Unfortunately you will still occasionally encounter some sites that do not play well with greylisting, see the references for tips on how to deal with those.

Do we need blacklists?

With greylisting taking care of most of the spam, is there still a place for blacklists? It is a fair question. The answer depends in a large part on how the blacklists you are considering are constructed and how much you trust the people who generate them and the methods they use.

The theory behind all good blacklists is that once an IP address has been confirmed as a source of spam, it is unlikely that there will be any valid mail send from that IP address in the foreseeable future.

With a bit of luck, by the time the spam sender gets around to trying to deliver spam to addresses in your domain, the spam sender will already be on the blacklist and will in turn treated to the *s-l-o-w* SMTP dialogue.

Knowing how a host makes it into a blacklist is important, but a clear policy for checking that the entries are valid and for removing entries is essential too. Once spam senders are detected, it is likely that their owners will do whatever it takes to stop the spam sending. Another reason to champion *aggressive maintenance* of blacklists is that it is likely that IP addresses are from time to time reassigned, and some ISPs do in fact not guarantee that a certain physical machine will be assigned the same IP address the next time it comes online.

Your spamd.conf file contains a few suggested blacklists. You should consider carefully which ones to use. Take the time you need to look up the web pages listed in the list descriptions in the *spamd.conf* file and then decide which lists fit your needs. If you decide to use one or more blacklists, edit your spamd.conf to include those and set up a cron job to let spamd-setup load updated blacklists at regular intervals.

The lists I consider the more interesting ones are the *nixspam* list, with a 4 day expiry and the *uatraps* list, with a 24-hour exiry. The nixspam list is maintained by *ix.de*, based on their logs of hosts that have verifiably sent spam to their mail servers. The uatraps list is worth looking into too, mainly because it is generated automatically by greytrapping.

Behavior based response: Greytrapping

Greytrapping is yet another useful technique that grew out of hands-on empirical study of spammer behavior, taken from the log data available at ordinary mail servers. You have probably seen spam messages offering lists of millions of verified email addresses available. However, verification goes only so far. You can get a reasonable idea of the quality of that verification if you take some time to actually browse mail server logs for failed deliveries to addresses in your domain. In most cases you will find a number of attempts at delivering to addresses that either have never existed or at least have no valid reason to receive mail.

The OpenBSD spamd developers saw this too. They also realized that what addresses are deliverable or not in your own domain is something you have complete control over, and they formulated the following rule to guide a new feature to be added to spamd:

"if we have one or more addresses that we are quite sure will never receive valid email, we can safely assume that any mail sent to those addresses is spam"

that feature was dubbed greytrapping, and was introduced in spamd in time for the OpenBSD 3.7 release.

The way it works is, if a machine that is already greylisted tries to deliver mail to one of the addresses on the list of known bad email addresses, that machine's IP address is added to a special local blacklist called *spamd-greytrap*. The address stays in the spamd-greytrap list for 24 hours, and any SMTP traffic from hosts in that blacklist is treated to the tarpit for the same period.

This is the way the uatraps list is generated. Bob Beck put a list of addresses he has referred to as *ghosts of usenet postings past* on his local greytrap list, and started exporting the IP addresses he collects automatically to a freely available blacklist.

As far as I know, Bob has never published the list of email addresses in his spamtrap list, but the machines at University of Alberta appear to be targeted by enough spammers to count.

At the time this article was written, the uatraps list typically contained roughly 120,000 addresses, and the highest number of addresses I have seen reported by my spamd-setup was just over 180,000. See Figure 1.

By using a well maintained blacklist such as the uatraps list you are likely to add a few more percentage points to the amount of spam stopped before it reaches your content filtering or your users, and you can enjoy the thought of actively wasting spammers' time.

A typical log excerpt for a blacklisted host trying to deliver spam looks like what you see in Listing 1.

This particular spammer hung around at a rate of 1 byte per second for 403 seconds (six minutes, forty-three seconds), going through the full dialogue all the way up to the DATA part before my spamd rejected the message back to the spammer's queue.

That is a fairly typical connection length for a blacklisted host. Statistics from my sites (see Figure 2) show that most connections to spamd last from 0 to 3 seconds, a few hang on for about 10 seconds, and the next peak is at around 400 seconds.

Then there is a very limited number that hang around for anywhere from 30 minutes to several hours, but those are too rare to be statistically significant.

Interaction with a running spamd: spamdb

Your main interface to the contents of your spamd related data is the spamdb administration program. The command:

```
$ sudo spamdb
```

without any parameters will give you a complete listing of all entries in the database, whether WHITE, GREY or others. In addition, the program supports a number of different operations on entries in spamd's data, such as adding or deleting entries or changing their status in various ways. For example:

```
$ sudo spamdb -a 192.168.110.12
```

will add the host 192.168.110.12 to your spamd's whitelist or update its status to WHITE if there was an entry for that address in the database already. Conversely, the command:

```
$ sudo spamdb -d 192.168.110.12
```

will delete the entry for that IP address from the database. For greytrapping purposes, you can add or delete spamtrap email addresses by using a command such as:

```
$ sudo spamdb -T -a wkitp98zpu.fsf@da
tadok.no
```

to add that address to your list of spamtrap addresses. To remove the address, you substitute `-d` for the `-a`. The `-t` flag lets you add or delete entries for TRAPPED addresses manually.

Hitting back, poisoning their well: Summary of my field notes

Up util July 2007, I ran my spamd installations with greylisting, supplemented by hourly updates of the uatraps blacklist and a small local list of greytrapping addresses like the one in the previous section, which is obviously a descendant of a message-id, probably harvested from a news spool or from some unfortunate malware victim's mailbox. Then something happened that made me take a more active approach to my greytrapping.

My log summaries showed me an unusually high number of attempted deliveries to non-existent addresses in the domains I receive mail for. Looking a little closer at the actual logs showed spam *backscatter*: Somebody, somewhere had sent a large number of messages with made up addresses in one of our domains as the `From:` or `Reply-to:` addresses, and in those cases the `To:` address was not deliverable either, the bounce messages were sent back to our servers. The fact that they were generating bounces to the spam messages indicates that any copies of those messages directed at actually deliverable addresses in those domains would have been delivered to actual users' mailboxes, not too admirable in itself.

Another variety that showed up when I browsed the spamd logs was the type illustrated in Listing 2, which could only mean that the administrators at that system had not yet learned that spammers no longer use their own `From:` addresses. Roughly at that time it struck me:

- Spammers, one or more groups, are generating numerous fake and nondeliverable addresses in our domains.
- Adding those generated addresses to our local list of spamtraps is mainly a matter of extracting them from our logs
- If we could make the spammers include those addresses in their to: addresses, too, it gets even easier to stop incoming spam and shift the spammers to the one-byte-at-a-time tarpit. Putting the trap addresses on a web page we link to from the affected domains' home pages will attract the address slurping robots sooner or later.
- Or the short version: Let's poison their well![5]

Over the following weeks and months I collected addresses from my logs and put them on the web page at *http://www.bsdly.net/~peter/traplist.shtml*.

After a while, I determined that harvesting the newly generated soon-to-be-spamtrap addresses directly from our greylist data was more efficient and easier to script than searching the mail server logs. Using spamdb, you can extract the current contents of the greylist with

```
# spamdb | grep GREY
```

which produces output in the format you see in Listing 3.

Where GREY is what you think it is, the IP address is the sending host's address, the third entry is what the sender identified as in the SMTP dialogue (HELO/EHLO), the fourth is the `From:` address, the fifth is the `To:` address. The next three are date values for first contact, when the status will change from GREY to WHITE and when the entry is set to expire, respectively. The final two fields are the number of times delivery has been blocked from that address and the number of connections passed for the entry. For our purpose, extracting the made up `To:`

Listing 1. Spamd's SMTP dialogue with a blacklisted host

```
Jan 16 19:55:50 skapet spamd[27153]: 82.174.96.131: connected (3/2),
lists: uatraps
Jan 16 19:59:33 skapet spamd[27153]: (BLACK) 82.174.96.131: <bryonRoe@boxe
rdelasgargolas.com> -> <schurkoxektk@ehtrib.org>
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: From: "bryon Roe"
<bryonRoe@boxerdelasgargolas.com>
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: To:
schurkoxektk@ehtrib.org
Jan 16 20:01:17 skapet spamd[27153]: 82.174.96.131: Subject: vresdiam
Jan 16 20:02:33 skapet spamd[27153]: 82.174.96.131: disconnected after
403 seconds. lists: uatraps
```

Listing 2. Spamd encounters clulessness

```
Jul 13 14:36:50 delilah spamd[29851]: 212.154.213.228: Subject: Consi-
dered UNSOLICITED BULK EMAIL, apparently from you Jul 13 14:36:50 deli-
lah spamd[29851]: 212.154.213.228: From: "Content-filter at srv77.kit.kz"
<postmaster@srv77.kit.kz> Jul 13 14:36:50 delilah spamd[29851]:
212.154.213.228: To: <skulkedq58@datadok.no>
```

Listing 3. Spamdb greylist output format

```
GREY|96.225.75.144|Wireless_Broadband_Router|<aguhjwilgxj@bn.cam
com.it>|<bsdly@bsdly.net>|1198745212|1198774012|1198774012|1|0
GREY|206.65.163.8|outbound4.bluetie.com|<>|<leonard159@data
dok.no>|1198752854|1198781654|1198781654|3|0
GREY|217.26.49.144|mxin005.mail.hostpoint.ch|<>|<earle@data
dok.no>|1198753791|1198782591|1198782591|2|0
```

addresses in our domains from backscatter bounces, it is usually most efficient to search for the "<>" indicating bounces, then print the fifth field. Or, expressed in grep and awk:

```
$ sudo spamdb | grep "<>" | awk -F\|
'{print $5}' |  tr -d '<>' | sort |
uniq
```

will give you a sorted list of unique intended bounce-to addresses, in a format ready to be fed to a corresponding script for feeding to spamd. The data in Listing 3 and the command line here would produce:

- *earle@datadok.no*
- *leonard159@datadok.no*

In some situations, the list will be a tad longer than in this illustration. This does not cover the cases where the spammers apparently assume that any mail with `From:` addresses in the local domain will go through, even when they come from elsewhere. Extracting the fourth column instead:

```
# spamdb | grep GREY | awk -F\|
'{print $4}' | grep mydomain.tld | tr
-d '<>' | sort | uniq
```

will give you a list of `From:` addresses in your own domain to weed out a few more bad ones from.

After a while, I started seeing very visible and measurable effects. At short intervals, we see spam runs targeting the addresses in the published list, working their way down in more or less alphabetical order. For example, in my field notes dated November 25, 2007, I noted earlier this month the address *capitalgain02@gmail.com* started appearing frequently enough that it caught my attention in my greylist dumps and log files.

The earliest contact as far as I can see was at Nov 10 14:30:57, trying to spam *wkzp0jq0n6.fsf@datadok.no* from 193.252.22.241 (apparently a France Telecom customer). The last attempt seems to have been ten days later, at Nov 20 15:20:31, from the Swedish machine 217.10.96.36.

My logs show me that during that period 6531 attempts had been made to deliver mail from *capitalgain02@gmail.com* via bsdly.net, from 35 different IP addresses, to 131 different recipients in our domains. Those recipients included three deliverable addresses, mine or aliases I receive mail for. None of those attempts actually succeeded, of course.

It is also worth noting that even a decreipt the Pentium III 800MHz at the end of the unexciting DSL line to my house has been able to handle about 190 simultaneous connections from TRAPPED addresses without breaking into a sweat. For some odd reason, the number of simultaneous connection at the other sites I manage with better bandwidth have not been as high as the ones from my home gateway. During the months I have been running the trapping experiment, the number of spamtrap addresses in the published list has grown to more than 10,000 addresses. Oddly enough, a my greylist scans still show up a few more every few days. Meanwhile, my users report that spam in their mailboxes is essentially non-existent. On the other side of the fence, there are indications that it may have dawned on some of the spammers that generating random addresses in other people's domains might end up poisoning their own well, so they started introducing patterns to be able to weed out their own made up addresses from their lists. I take that as a confirmation that our harvesting and republishing efforts have been working rather well.

The method they use is to put some recognizable pattern into the addresses they generate. One such pattern is to take the victim domain name, prepend "dw" and append "m" to make up the local part and then append the domain, so starting from *sia.com* we get *dwsiam@sia.com*.

There is one other common variation on that theme, where the prepend string is "lin" and the append string is "met", producing addresses like *linhrimet@hri.de*. Then again when they use that new, very recognizable, address to try to spam my spamtrap address *malseeinvmk@bsdly.net*, another set of recognition mechanisms are activated, and the sending machine is quietly added to my spamd-greytrap. And finally, there are clear indications that spammers use slightly defective relay checkers that tend to conclude that a properly configured spamd is an open relay, swelling my greylists temporarily. We already know that the spammers do not use `From:` addresses they actually receive mail for, and consequently they will never know that those messages were in fact never delivered. If you have read this far and you are still having fun, you can find other anecdotes I would have had a hard time believing myself a short time back in my field notes at *http://bsdly.blogspot.com/*. By the time the magazine has been printed and distributed, there might even be another few tall tales there.

References

[1] *The Book of PF*, by Peter N. M. Hansteen, No Starch Press December 2007, available in better bookshops or from the publisher at: *http://www.nostarch.com/pf.htm*.

[2] Note that on FreeBSD, spamd is a port, so you need to install that before proceeding. Also, on recent FreeBSDs, the `rc.conf` lines are `obspamd_enable="YES"` to enable spamd and `obspamd_flags=""` to set any further flags.

[3] The Next Step in the Spam Control War: Greylisting, by Evan Harris available at *http://greylisting.org/articles/whitepaper.shtml*.

[4] Available at *http://marc.info/?l=openbsd-misc&m=116136841831550&w=2*.

[5] Actually in the first discussions about this with my BLUG user group friends, we referred to this as *brřnnpissing* in Norwegian, which translates as *urinating in their well*. The more detailed descriptions of the various steps in the process can be tracked via blog entries at *http://bsdly.blogspot.com*, starting with the entry dated Monday, July 9th, 2007, *http://bsdly.blogspot.com/2007/07/hey-spammer-heres-list-for-you.html*.

About the Author

Peter N. M. Hansteen is the author of The Book of PF (No Starch Press, December 2007). Peter has been tinkering with computers and networks since the mid-1980s, found the Freenixes in the early 1990s and is a frequent lecturer on PF and other OpenBSD and FreeBSD topics. He is a consultant, sysadmin and writer based in Bergen, Norway and occasionally blogs at *http://bsdly.blogspot.com/*.

BSD CERTIFICATION.ORG

NEW CERTIFICATION EXAM

The BSD Certification Group proudly announces the availability of the BSD Associate Exam (BSDA), the entry level exam for BSD System Administrators.

The BSD Associate Exam is a written proctored certification exam in English only. The BSDCG has worked hard to make this psychometrically valid exam affordable worldwide. See the list of selected conferences and register for an exam seat for $75 USD at **www.bsdcertification.org**.

Get Involved.
Get Certified.
Get Ahead.

www.iXsystems.com

iXsystems is a proud sponsor of BSD Certification Group Inc.

NetBSD FreeBSD OpenBSD DragonFlyBSD

Defense in Depth and FOSS

Henrik Lund Kramshøj

Protecting your infrastructure is a tough job and one that requires a lot of attention. Meanwhile a lot of new tools are being published that can protect and some that can attack. This war between good and bad can take up a lot of resources, and how much is enough.

In this article, I will introduce Defense in Depth which can help you lessen the burden while increasing your security stance for your infrastructure and servers.

Defense in Depth is an expression which comes from a military background, but is being used in Information Security to mean multiple layers of security with the goal to delay or deter an attacker.

When I suggest you should use multiple layers I recommend that you use everything you can, as long as you can control it. I do not suggest implementing a Fort Knox, but then leave it alone to become an heap of systems that are not being used or updated.

So use everything you can within your available resources. Good security is the result you get from a long term strategy and using well balanced security measures to protect your investment and the value of your assets.

So to summarize why I want you to think about Defense in Depth:

- Multiple layers of security makes it difficult to get past all of them.
- Multiple layers of security allows one layer to be weak at times, as long as the other layers are in place.

We want the attackers to spend a lot of time and resources while they risk being detected more easily.

The rest of this article will assume that some kind of Unix environment is being used for servers and some examples services are being shown. It is assumed that the servers are being run by some small group of people. The main goal is to reach servers that are supposed to be reasonably secure, while not spending a lot of resources on administration. It could be a shared server for some smaller private websites or a community driven server.

The main goal of the changes proposed are to ease the burden on the administrators while making an attacker suffer and waste time.

Small changes that do not hurt the ease of use for authorized users, but add to the difficulty for attackers are preferred.

Easy to do but hard to circumvent

Everybody reading this article probably knows the *Secure Shell* (SSH) protocol and program which allow secure command line access across insecure networks. Most will also recognize the OpenSSH project which is the specific software being used on most Unix variants today for providing this login service.

The default OpenSSH configuration file for the server part is named `sshd_config` and often placed in `/etc/ssh/sshd_config`. If not located there use the locate command or manuals to locate it in your operating system.

Looking at this file it is easy to make some simple changes that can impact the security. Remember to make a backup of the original configuration file.

Easy to make changes are the Protocol and Port options. These are by default Protocol 2,1 and Port 22, which allow most client to connect easily – even really old clients that only support SSH Protocol version 1. This should be changed into Protocol 2 only – as people should upgrade to a newer client anyway. If a client does not support version 2 it must be quite old, and is likely to have other security problems. SSH Protocol version 1 has known flaws so by forc-

ing everyone to use version 2 the security is improved a little – without loss of functionality.

OK, that was a long explanation, so now I will move a little faster.

The port is still 22 – but why? Change the port and an attacker would have to do a port scan of the server to find the SSH port. By changing the port number to 24679 (choose something random between 1-65535) an attacker will have to target your server specifically and risk detection more easily. An added bonus is also that worms and other automated tools will not try this port number by themselves.

To finish this port number change, tell the administrators that they should add this to their SSH client configuration. Using the OpenSSH client this file is named `$HOME/.ssh/config`.

```
host server1 server2 server3
    port 24679
```

Hey, is not that security by obscurity – changing the port number does not prevent them from accessing the port! Yes, that is right – and security by obscurity alone does not prevent anything. In this case changing the port number makes it easier to see something is wrong, since any connection to port 22 is wrong and port scans to find the port number for SSH would easily be detected. On my servers I enjoy not having lots and lots of logs being generated because people and tools scan for SSH.

- Security by obscurity combined with real security often improves security.

Let's get some real security for the SSH daemon. Real security could be to decide only to allow access using key based authentication – instead of passwords. Why, because brute forcing a key is impossible while most research shows that people choose bad passwords. Using keys is a long term solution to this problem, you do not have to change password very often since it does not allow login.

Using keys for OpenSSH is easy. First you generate a key on a secure machine with `ssh-keygen -t dsa` or `ssh-keygen -t rsa` and copy the public key, named *.pub* like `id_dsa.pub` part onto the server as `$HOME/.ssh/authorized_keys` and make sure that permissions are set correctly by doing `chmod -R 700 $HOME/.ssh`. Key generation is a one-time job and adding the key to yet another server is very easy.

In fact this scheme allows you to publish your public key on the internet and people can give you access by installing that on a server, great for remote consulting.

When this is in place you can enjoy the added benefit of SSH key agents which can store the key in memory and allow secured access to the server without entering passwords. Just load in the key when logging in and the rest of the session you can log onto servers without having to remember and enter passwords all the time. If not doing this already do it immediately, it saves time for each and every logon. The tool is called ssh-agent in OpenSSH and the Putty installer package includes the Pageant program to do the same thing.

With keys in place you can disable password based logon by changing these options:

```
# Password authentication is disabled,
Note: BOTH must be set to "no"
PasswordAuthentication  no
ChallengeResponseAuthentication no
```

And while you are there remove the possibility of root logins, if enabled:

```
PermitRootLogin          no
```

If you like you can also list the specific users that are allowed to login to the server with SSH with options:

```
AllowUsers               john mike
tina joe lisa
# or based on groups
#AllowGroups              admguy oper
secadmin dbagroup
```

While possible to use of Deny options, to disallow logins. It is regarded as better security to make a policy of allowing what is good, rather than trying to enumerate all things bad.

If you use this strategy you do not have to risk creating a database or application user with a bad password which can be broken into.

Performing actions as root

So now that we cannot login to root directly from the network we have to get another way of performing administration. For this purpose I suggest that you use Sudo (su *do*). A lot of you might already use sudo, and a lot of people just use a command similar to

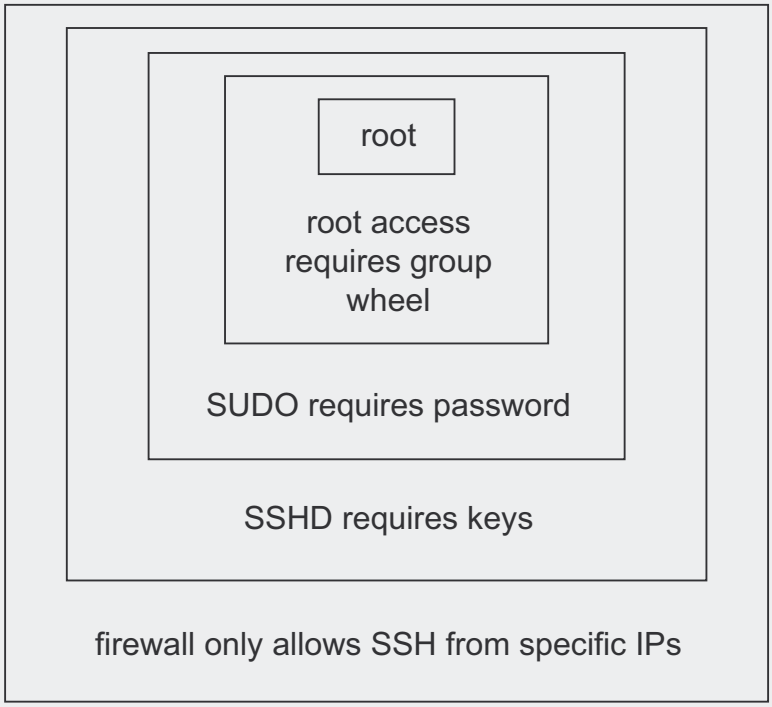


Figure 1. Security layers 1 uk

sudo su – to get to a root-like shell with all privileges.

This is hardly the best way to run sudo. Sudo is designed with security in mind and can be used much more efficiently.

First of all adding sudo to a command before running it is better, since you only run with high privileges for a short time – and will not accidently type `rm -rf *` in the wrong place: So I prefer just using commands like this:

```
sudo apachectl stop
sudo rndc reload
```

So first use is that you should keep away from using sudo to run shells. And when you need a superuser shell use the command `sudo -s` which is shorter to write. When using this command or `sudo ksh -l` if you need an environment similar to a real login environment you should require a password.

Having to type in passwords is a pain. This is fine since it is making it harder for you to do the wrong thing – running superuser shells all the time. With Sudo you also have to choose which password should be used to execute command as another user.

The options are to use the password of the user using Sudo, or use the password of the target user of the command. So go ahead choose if you want to use the users password or the superuser password for getting root shells and configure Sudo to do this.

The first is default, easy, recommended and can be enabled using *visudo* and uncomment the following part:

```
# Uncomment to allow people in
group wheel to run all commands
%wheel        ALL=(ALL)       ALL
```

Requiring the root password instead is a flag option called root. I recommend that you do not give out the root password. Use the default Sudo behavior.

This configuration need a comment on how to read it. The first part (`%wheel`) is the name of a group defined in `/etc/group`.

The `ALL= part` is the beginning of a definition of the commands which this group can run.

The first `ALL` in parentheses is a pattern specifying which hosts these commands are allowed on and the last `ALL` specifies that this group can in fact run any command.

Allowing `ALL` commands might be a bit to much, but you will not accidently lock yourself out.

Then to make your daily life easier you should add a number of things to this sudoers file. Since Sudo provides the possibility of fine grained security you can specify specific commands to be run without password.

The following is inspired by the sudo manual page and adapted.

First decide who should be allowed to run special commands and create group definitions: (you can also use Unix groups from `/etc/group` like `%wheel` if you like instead)

```
# User alias specification
User_Alias      FULLTIMERS = john,
mike, tina
User_Alias      WEBMASTERS = joe, lisa
```

Then add some command aliases like:

```
Cmnd_Alias      SHUTDOWN = /usr/sbin/
shutdown
Cmnd_Alias      REBOOT = /usr/sbin/
reboot
Cmnd_Alias      APACHECTL = /usr/sbin/
apachectl
Cmnd_Alias      HTTPDCONF = sudoedit
/var/www/conf/httpd.conf
Cmnd_Alias      SHELLS = /bin/sh,
/bin/ksh
```

and you can define some things that should be allowed:

```
FULLTIMERS ALL = (ALL) NOPASSWD:
SHUTDOWN, REBOOT, APACHECTL, PASSWD:
SHELLS
WEBMASTERS      ALL = NOPASSWD:
APACHECTL, PASSWD: HTTPDCONF
```

When this is in place the result is that most daily administration, rebooting and restarting the web server is done easily without password, without compromising the root password by letting everyone have it.

Another great feature of Sudo is also introduced, sudoedit – which can make editing specific files easier while being secure.

Note that since it is possible to edit httpd.conf a WEBMASTER could make the webserver run as root and circumvent the current setup, see below.

Can you explain a bit?

Yes, the groups and commands defined are being allowed due to the user specifications for administrators and webmasters. The specification says:

If you are in the user group named WEBMASTERS then you are allowed to run the following commands on ALL hosts.

The command(s) listed in alias APACHECTL do not need a password, while editing the httpd.conf defined by alias HTTPDCONF requires that you enter a password.

The great thing about Sudo and the sudoers file is that since you can name hosts you are able to produce a single sudoers for a quite large company and distribute it to all servers!

Web server configuration

This section assumes some basic knowledge of Apache HTTPD web server, which is configured using the httpd.conf configuration file.

The current setup has a small flaw, since the webmasters can edit httpd.conf – they can make it run as root, and by introducing a small program they would be able to execute things as root. This is bad and can be helped a little. We consider the webmasters trustworthy and the above is primarily in effect to avoid a user which is able to compromise an account to easily gain root – it would require the knowledge of the user password.

To help this problem we might decide to split up the main Apache configuration into separate files. First thing you should do is to make a copy of the existing httpd.conf, so you can easily find differences – if something does not work.

Then you should create files as needed like this, actual example from a server I have running:

- `httpd.conf` – main configuration file with options like DocumentRoot, ServerAdmin, ServerName and Listen directives.
- `ssl.conf` – if you use SSL then split this into a separate file.
- `virtual.conf` – everything regarding virtual hosts are moved into this file.

This file is usually the only one to be edited, when adding and removing virtual hosts.

The samples files included with Apache includes a lot of comments and can be hard to get a grip on. You can make it much easier to configure if you create clean versions.

So start by making a clean version, like the one proposed in the SecurityFocus article referenced below. My httpd.conf – without virtual hosts is less than 300 lines while the sample httpd.conf is more than 1000 lines. My ssl.conf is 78 lines, while the ssl-std.conf is 246 lines. Would you rather search for an option to change in 1300 lines or 550 lines?

Now we have main options in a single file httpd.conf, we have SSL specific options in another and we can allow the webmasters to edit the virtual.conf – so you might want to change Sudo configuration into:

```
Cmnd_Alias      HTTPDCONF = sudoedit
/var/www/conf/virtual.conf
```

This still does not prevent them from adding options other than Directory/VirtualHost options – but since nothing else should be in that file it is easy to spot, and can even be done programmatically. At least they would be less likely to mess up the configuration completely.

I also recommend that you start using version control on these files with tools like CVS or SVN since this makes it easy to remove things you do not need from these files – without loosing information and without having to drag along 100s of lines that are commented out. Having CVS and SVN check the files before adding them to a repository is also a way to make sure they are wellformed, before being put into production.

Securing Apache HTTPD

Securing Apache HTTPD web server is the subject of books, but some of the main things to do are:

- Run apache chrooted – another layer to go through
- Disable banner by disabling signature in generated web pages and version in HTTP header, security by obscurity attacker might have to probe before launching attacks
- Run with a minimal configuration, disable modules you do not need at compile time or by changing configuration files.

Some specific options that are easy to add are:

```
UseCanonicalName Off
ServerSignature Off
ServerTokens Prod
```

Having these extra layers does make a difference.

Conclusion

Use the tools you have to the fullest extent to make it harder to break into your system.

Most of the things described above are a one-time thing which can really improve your security for a long time. Remember to customize to your own environments and get into the habit of thinking about these things.

Things that you should also look into, when adding layers are:

- Firewalls which should of course deny everything and then allow specific services through. Services available should also only be available to those who need it. Apache HTTPD port 80 should be allowed from everywhere, while SSH should only be allowed from specific places.
- NTP having the correct time is really important and something you should always configure on all equipment! Managed switches, routers, servers and also workstations.
- Syslog is also great for centralizing logging. Centralized logging makes it easy to search, you do not have to log into each and every host. One single log also makes it easy to spot trends. Having logs sent to another host also makes it harder to zap log entries, removing evidence of an attack after owning the machine. Cleaning logs in a system using centralized logging would at least require hacking both the server and the centralized logserver.
- Integrity protection can also add yet another barrier, since using these technologies will help you detect changes to critical files Quickly. Some tools to consider are SamHain and Osiris. These tools should run at least once a day and send a report to administrators.
- Intrusion Detection software like Snort can also help detect when you are under attack, but these tools often require more attention and increase the burden on administrator. Use it if you have the ressources, but otherwise do not.

Links

- *http://en.wikipedia.org/wiki/Defense_in_depth*
- *http://www.kramse.dk/keys_en.html*
- *http://www.ranum.com/security/computer_security/editorials/dumb/*
- *http://www.gratisoft.us/sudo/*
- *http://www.chiark.greenend.org.uk/~sgtatham/putty/*
- *http://httpd.apache.org/*
- *http://www.securityfocus.com/infocus/1786*

About the Author

Henrik Lund Kramshøj is a freelance security consultant working from Copenhagen in teaching, pentesting and consulting. Henrik has a master degree in computer science from the University of Copenhagen DIKU, and also CISSP and CEH certifications. He was awarded the Certified Information Systems Security Professional (CISSP) designation after passing the requirements in May 2003. Former positions held are: senior consultant in WM-data, security researcher for Neupart A/S and group manager in VIGILANTe Inc. Henrik is also a known speaker on subjects such as UNIX security, Internet Privacy Issues, Using Cryptography technologies and Computer Forensics.

NetBSD on the NSLU2

Donald T.Hayford

If you own a Network Storage Link (NSLU2) from Linksys, you have a nifty, relatively inexpensive device with an Arm-5TE-compatible Xscale processor, 8 MB of flash memory, 32 MB of SDRAM, two USB ports, and a 10/100 Ethernet port.

Often referred to fondly as the *Slug*, you can use the NSLU2 in its off-the-shelf configuration as a network addressable storage (NAS) device that hosts USB flash or hard drives that can be accessed from your Windows computer. If you are a little bit adventurous, then you know that you can replace the firmware in your Slug and use it as NAS for Linux, BSD and Mac PCs, as well. And if you are really adventurous, then you have replaced your firmware with a version from the kind folks at *www.nslu2-linux.org* and boot up your NSLU2 into a full version of ... Debian Linux.

Yep, Linux. And you thought this was a magazine about BSD. Well, get prepared for another adventure and learn how to boot your NSLU2 into a full version of NetBSD.

What you will need:

- A Linksys NSLU2.
- A serial connection to your NSLU2. Once again, the good people at nslu2-linux.org have shown us the way at *www.nslu2-linux.org/wiki/HowTo/AddASerialPort*. *Please note that modifying your NSLU2 to provide an external serial port will void your warranty*. While you are at it, another good way to void your warranty is to remove the resistor that limits the clock speed of the NSLU2 to half of the Intel design speed. See the page on speeding up your Slug at *http://www.nslu2-linux.org/wiki/HowTo/OverClockTheSlug*.
- A computer running virtually any *nix compatible operating system. The instructions that you will see below are for Debian (i386) Linux (yes, I know, it is not a BSD, but at least it is not Windows). The build will require about 4 GB of disk space, so make sure you have enough room.
- A computer that can serve as a TFTP server (can be the same as in 3 above).
- A computer that can serve NFS files (also can be the same as in 3 above).

What you will end up with:

- A USB drive (either a flash drive or a full hard disk) with the Arm-5TE version of NetBSD that you can use to run NetBSD on your NSLU2,
- The knowledge of how to cross-compile NetBSD on another system, including how to build a memory-based disk that can be used to run a smaller, or disk-free, version of NetBSD
- A NSLU2 that will still boot up whatever version of software it currently boots. Because there are still some issues to work out, we aren't going to write anything to the flash memory at this time, so a simple power cycle will get your NSLU2 back to the same condition it was before it started.

A Little Background on NetBSD and the NSLU2

For those not familiar with NetBSD, it is a freely available and redistributable Unix-like operating system (*www.netbsd.org/about/*) that prides itself on being easily ported to a large number of different processors and machine configurations. Primarily, this is done from a single, common codebase, though there is some specific software for

each processor type, each peripheral type, and each machine configuration. For example, there are twenty one different *processor/peripheral/machine configurations* supported in the evbarm (evaluation boards – arm) architecture, one of which is the NSLU2. Support for the NSLU2 has been around since early 2006, mainly as a result of the efforts of software developers like Steve Woodford, Jason Thorpe (the current port-arm maintainer), Sam Leffler and Ichiro Fukuhara (and certainly others – forgive me for leaving your name out). If you are interested in the specifics for the NSLU2, you will find the NSLU2-only code in the *src/sys/arch/evbarm/nslu2* directory of the source code (more about getting the source code later) and more general Xscale-specific code in the *src/sys/arch/arm/xscale* directory. For those interested in learning more about embedded software, both are excellent locations to browse.

Unfortunately, some of the code you need to run NetBSD on your Slug, namely the firmware for the ethernet controller, is licensed by Intel in such a way that NetBSD can not include it in their distribution. As a result, building NetBSD for the NSLU2 will always require a little bit of manual tweaking. Though the NSLU2 has some support in 4.0-release, that support did not include instructions for adding the ethernet controller software from Intel. For that, we must turn to NetBSD-current. Because it is current, and because it is changing daily, you will occasionally find that something does not work exactly right. Be patient, check the mail lists and ask questions.

Getting and Installing the Source Code

The best method of getting the source code is with CVS. There are tar balls that you can download, but my experience with that route has not been totally satisfactory. Fortunately, using CVS is pretty painless. Assuming you have CVS installed, use the following commands at the command line interface (i.e., a system terminal). If you do not have CVS, you will have to install it for your system, the details of which will depend on what system you have (for Debian, it is *apt-get install cvs*). These command will create a new directory under your home directory called ~/net, get the source using cvs, and put it all into a directory `~/net/src`.

```
$ mkdir ~/net
$ export CVS_RSH=ssh
$ export CVSROOT=:ext:anoncvs@anoncvs
.NetBSD.org:/cvsroot
$ cd ~/net
$ cvs checkout -A -P src
```

Now, get the Intel proprietary code, as described by Steve Woodford in the file `~/net/src/arch/arm/xscale/ixp425-fw.README` (most of the steps below for compiling the firmware are shamelessly copied from Steve's excellent directions). We will create a directory to keep and compile this code that is separate from the NetBSD src.

```
$ mkdir ~/net/ixp-npe
```

Point your browser to *http://www.intel.com/design/network/products/npfamily/ixp400_current.htm* and select the *Download* (*without crypto*) option for the NPE Microcode v2.3 section. After registering with Intel and accepting their license agreement, download the file *IPL_ixp400NpeLibrary-2_3_2.zip* and save to the directory we just created. Then unzip the file.

```
$ cd ~/net/ixp-npe
$ unzip IPL_ixp400NpeLibrary-2_3_2.zip
```

Now, compile the firmware. If you are not comfortable using nano, or do not have it, use the editor of your choice to create the file `IxNpeMicrocode.h`.

```
$ cd ixp400_xscale_sw/src/npeDl
$ nano IxNpeMicrocode.h
```

Add these two lines to the file and save it.

```
#define IX_NPEDL_NPEIMAGE_NPEB_ETH
#define IX_NPEDL_NPEIMAGE_NPEC_ETH
```

Compile and run ixNpeDlImageConverter.c.

```
$ cc ixNpeDlImageConverter.c -o foo
$ ./foo
```

At this point, you should have a file in your directory called IxNpeMicrocode.dat that contains the firmware for the Ethernet controller in the Slug's processor. Copy this file to the `~/net/src/sys/arch/arm/xscale` directory (the same directory that the readme file was in).

```
$ cp IxNpeMicrocode.dat ~/net/src/
sys/arch/arm/xscale/
```

Cross-compiling NetBSD

NetBSD is one of the easiest operating systems to build that I have ever dealt with. We will exclusively use the file `~/net/src/build.sh` to build the cross-compiler and other tools as well as for building the operating system itself. To build the tools:

```
$ cd ~/net/src
$ ./build.sh -m evbarm -a armeb tools
```

Now, we need two configuration files, one for the normal kernel and one that describes the installation version of the kernel. The biggest difference between the two kernels is that the installation version has a subset of the userland (all of those files that make up the actual executable commands of the operating system) compiled in with it as a memory disk.

```
$ cd ~/net/src/sys/arch/evbarm/conf
$ cp ADI_BRH_INSTALL NSLU2_INSTALL

$ nano NSLU2_INSTALL
```

Change the line that reads

```
include "arch/evbarm/conf/ADI_BRH"
      to
include "arch/evbarm/conf/NSLU2".
```

Feel free to change the first and third lines to accurately reflect what the file does as well. Now, add an option to the NSLU2's normal configuration file.

```
$ cp NSLU2 NSLU2.orig
$ nano NSLU2
```

Change the line that reads

```
#options        FFS_EI          # FFS
Endian Independant support
        to
options         FFS_EI          # FFS
Endian Independant support.
```

Next, add `NSLU2_INSTALL` to the installation kernel compile targets:

```
$ cd ~/net/src/distrib/evbarm/
instkernel/instkernel
```

```
$ cp Makefile Makefile.orig
$ nano Makefile
```

Change the line

```
MDSETTARGETS=           ADI_BRH_
INSTALL         ${RAMDISK}      -
        to
MDSETTARGETS=     ADI_BRH_INSTALL
${RAMDISK}      -     \     NSLU2_
INSTALL          ${RAMDISK}      -
```

Finally, tell the build system to build the install version of the NSLU2 kernel.

```
$ cd ~/net/src/etc/etc.evbarm
$ cp Makefile.inc Makefile.inc.orig
$ nano Makefile.inc
```

Change the line

```
EVBARM_BOARDS= ADI_BRH to
EVBARM_BOARDS=  ADI_BRH NSLU2
```

Listing 1. DHCP Server Configuration file

```
#
# DHCP Server Configuration file.
#   see /usr/share/doc/dhcp*/dhcpd.conf.sample
#

ddns-update-style ad-hoc;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;

# the address of your nameserver

option domain-name-servers xxx.xxx.xxx.xxx;
default-lease-time 2592000;
allow bootp;
allow booting;

# most stuff goes here

subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers        192.168.1.1;
        range 192.168.1.110 192.168.1.189;
        }

group {

      # IP address of your TFTP server

      next-server 192.168.1.102;
      option routers 192.168.1.1;
      default-lease-time 2592000;
      host slug {

                #MAC address of your slug

                hardware ethernet 00:18:39:a2:26:7c;

                # allows your slug to use NFS for mounting root

                fixed-address 192.168.1.240;
                option root-path "/client/root";
                }
        }

# end of DHCP Server Configuration file.
```

Now for the slow part. First, build the standard kernel, then the release sets that include the base operating system as well as the installation version of the kernel. This process is thoroughly covered in chapter 30 of the NetBSD guide (*http://www.netbsd.org/docs/guide/en/index.html*).

```
$ cd ~/net/src
$ ./build.sh -u -m evbarm -a armeb
kernel=NSLU2
$ ./build.sh -u -U -m evbarm -a armeb
build
$ ./build.sh -u -U -m evbarm -a armeb
release
```

While you are waiting for the build to finish, take a look at the file `~/net/src/distrib/evbarm/instkernel/ramdisk/list`. This file controls the layout of the memory disk that is built into the install kernel. The first non-comment line shows where to find the files that will be used in the memory disk, and the rest of the file shows what those files are. An easy way to build a special version of NetBSD is by changing this file to control what files are available to the install kernel.

When the build is finished, you will find a number of zipped files under `~/net/src/obj/releasedir/evbarm` that you will need. We need the install kernel and the regular kernel. Let's put them someplace easy to find.

```
$ mkdir ~/net/kernels
$ cd ~/net/src/obj/releasedir/evbarm/
binary/kernel
$ gunzip -c netbsd.bin-NSLU2.gz >~/
net/kernels/netbsd.bin
$ cd ~/net/src/obj/releasedir/evbarm/
installation/instkernel
$ gunzip -c netbsd-NSLU2_
INSTALL.bin.gz >
   ~/net/kernels/netbsd-NSLU2_
INSTALL.bin
```

You also need to put the operating system files someplace where the installer can find it. These can be found under `~/net/src/obj/releasedir/evbarm/binary/sets`. The easiest way to do this is to use an NFS server that the installer can get the files from. If the base directory of the files served by the NFS server is `/client/home`, then:

```
$ cd ~/net/src/obj
$ cp -r releasedir /client/home/
```

Listing 2. Startup Message from the NSLU2_INSTALL Kernel

```
Loaded initial symtab at 0xc043a710, strtab at
0xc04608fc, # entries 9094
pmap_postinit: Allocated 9 static L1 descriptor
tables
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001,
2002, 2003, 2004, 2005,
    2006, 2007, 2008
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All
rights reserved.

NetBSD 4.99.52 (NSLU2_INSTALL) #0: Sat Feb  2 19:07:
05 EST 2008
        hayford@debian:/home/net/src/sys/arch/evbarm/
compile/obj/NSLU2_INSTALL
total memory = 32768 KB
avail memory = 24056 KB
mainbus0 (root)
cpu0 at mainbus0: IXP425 266MHz rev 1 (XScale core)
cpu0: DC enabled IC enabled WB enabled LABT branch
prediction enabled
cpu0: 32KB/32B 32-way Instruction cache
cpu0: 32KB/32B 32-way write-back-locking Data cache
ixpsip0 at mainbus0
com0 at ixpsip0 addr 0xc8000000-0xc8000fff: ns16550a,
working fifo
com0: console
ixp425_intr_establish(irq=15, ipl=3, func=c027ce18,
arg=c10ab200)
ixpclk0 at ixpsip0 addr 0xc8005000-0xc800502f
ixpclk0: IXP425 Interval Timer
ixpdog0 at ixpsip0: Watchdog Timer
slugiic0 at ixpsip0: I2C bus
slugbutt0 at ixpsip0: Power and Reset buttons
slugled0 at ixpsip0: LED support
ixpio0 at mainbus0
ixpio0: configuring PCI bus
pci0 at ixpio0 bus 0
ohci0 at pci0 dev 1 function 0: vendor 0x1033 product
0x0035 (rev. 0x43)
ixp425_intr_establish(irq=28, ipl=1, func=c0284108,
arg=c1105000)
ohci0: interrupting at INTA
ohci0: OHCI version 1.0
usb0 at ohci0: USB revision 1.0
uhub0 at usb0
uhub0: vendor 0x1033 OHCI root hub, class 9/0, rev
1.00/1.00, addr 1
ohci1 at pci0 dev 1 function 1: vendor 0x1033 product
0x0035 (rev. 0x43)
ixp425_intr_establish(irq=27, ipl=1, func=c0284108,
arg=c1108000)
ohci1: interrupting at INTB
ohci1: OHCI version 1.0
usb1 at ohci1: USB revision 1.0
uhub1 at usb1
uhub1: vendor 0x1033 OHCI root hub, class 9/0, rev
1.00/1.00, addr 1
ehci0 at pci0 dev 1 function 2: vendor 0x1033 product
0x00e0 (rev. 0x04)
ixp425_intr_establish(irq=26, ipl=1, func=c0284fbc,
arg=c110a800)
ehci0: interrupting at INTC
ehci0: companion controllers, 3 ports each: ohci0
ohci1
usb2 at ehci0: USB revision 2.0
uhub2 at usb2
uhub2: vendor 0x1033 EHCI root hub, class 9/0, rev
2.00/1.00, addr 1
ixme0 at mainbus0: IXP4xx MicroEngine Support
ixp425_intr_establish(irq=3, ipl=1, func=c03d5004,
arg=c1103000)
ixp425_intr_establish(irq=4, ipl=1, func=c03d5004,
arg=c1103000)
ixpnpe0 at ixme0 NPE-B
ixp425_intr_establish(irq=1, ipl=1, func=c03d6d64,
arg=c1102000)
npe0 at ixpnpe0: Ethernet co-processor
npe0: remember to fix rx q setup
npe0: Ethernet address 00:18:39:a2:26:7c
rlphy0 at npe0 phy 1: RTL8201L 10/100 media interface,
rev. 1
rlphy0: 10baseT, 10baseT-FDX, 100baseTX, 100baseTX-
FDX, auto
clock: hz=100 stathz=0 profhz=0
ixp425_intr_establish(irq=5, ipl=2, func=c03d4248,
arg=00000000)
iic0 at slugiic0: I2C bus
xrtc0 at iic0 addr 0x6f: Xicor X1226 Real-time Clock/
NVRAM
ixp425_intr_establish(irq=22, ipl=1, func=c03d9ccc,
arg=c10b4000)
ixp425_intr_establish(irq=29, ipl=1, func=c03d9c60,
arg=c10b4000)
ixp425_intr_establish(irq=28, ipl=1, func=c03da06c,
arg=c1102f00)
ixp425_intr_establish(irq=27, ipl=1, func=c03d9fc0,
arg=c1102f00)
ixp425_intr_establish(irq=26, ipl=1, func=c03d9f14,
arg=c1102f00)
ixp425_intr_establish(irq=5, ipl=2, func=c03d9e30,
arg=00000000)
md0: internal 3075 KB image area
umass0 at uhub2 port 1 configuration 1 interface 0
umass0: Maxtor OneTouch III, rev 2.00/0.01, addr 2
umass0: using SCSI over Bulk-Only
scsibus0 at umass0: 2 targets, 1 lun per target
```

Booting the NetBSD Installer

The NSLU2 comes with RedBoot, a flash-based utility that loads and executes the version of Linux that comes with the Slug. How that works is not important here (If you are interested, you can download the RedBoot code from the same site as the IXP firmware.), except that at the very beginning of this process, RedBoot will wait two seconds to allow the user to type a *control-C* into the serial port, halting the normal boot process. At this point, you can instruct RedBoot to load software from a tftp server instead of from flash. We will use that to boot the NetBSD installer.

Actually setting up the tftp server is sometimes complicated, but lots of good examples are available on the web. My suggestion is to Google *tftp server howto your-operating-system* (For example, I found *http://www.linuxtopia.org/online_books/installation_guides/FC7_fedora_core_7_installation_guide/ap-pxe-server.html* for Fedora 7.); more than likely you will find numerous sites with instructions that you can tailor for your system. You will also need to set up a DHCP server, which means setting up a dhcpd.conf file in your /etc directory. Here is mine: Listing 1.

If you use a router as a DHCP server for your network, you will have to temporarily disable it when you want boot up the NSLU2. When you set up the tfpt server, you will need to specify a location where the daemon can find the files it is asked to serve. Most people use `/tftpboot` as this location. Copy the two kernels you will need to this location.

```
$ cp ~/net/kernels/*bin /tftpboot/
```

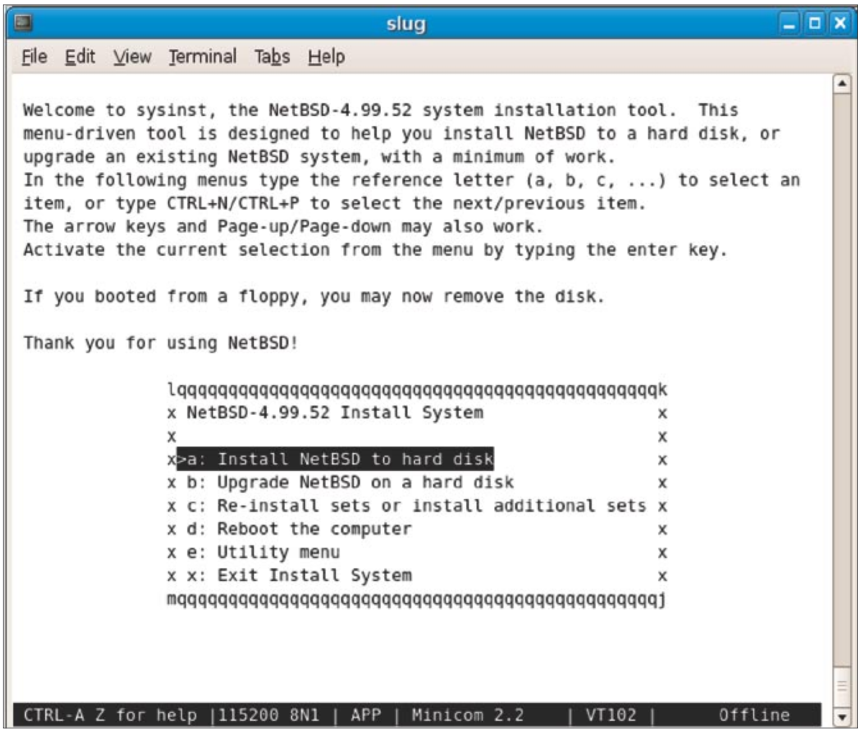


Figure 1. The startup screen for the NetBSD Installer

Listing 3. If problems occur during system installation, select menu item 'x' for a shell prompt

```
lqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqk
x NetBSD-4.99.50 Install System                  x
x                                                x
x>a: Install NetBSD to hard disk                 x
x b: Upgrade NetBSD on a hard disk               x
x c: Re-install sets or install additional sets  x
x d: Reboot the computer                         x
x e: Utility menu                                x
x x: Exit Install System                         x
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj
```

Set your serial port to 115200 N81 (115 *kBaud*, no parity, eight total bits, and one stop bit), using a serial communication program like minicom. Turn on the NSLU2 and hit *control-C* when you see the following:

```
RedBoot(tm) bootstrap and debug
environment [ROMRAM]
Red Hat certified release, version 1.92
- built 15:16:07, Feb  3 2004
Platform: IXDP425 Development Platform
(XScale)
Copyright (C) 2000, 2001, 2002, Red
Hat, Inc.
RAM: 0x00000000-0x02000000,
0x000723a0-0x01ff3000 available
FLASH: 0x50000000 - 0x50800000, 64
blocks of 0x00020000 bytes each.
== Executing boot script in 2.000
seconds - enter ^C to abort
^C
RedBoot>
```

Tell RedBoot the address of the tftp server, then where and which kernel to load. Once the kernel is loaded, execute it.

```
RedBoot> ip_address -h 192.168.0.2
IP: 192.168.0.1/255.255.255.0,
Gateway: 192.168.0.1
Default server: 192.168.0.2, DNS
server IP: 0.0.0.0
RedBoot> load -r -b 0x200000 netbsd-
NSLU2_INSTALL.bin
Using default protocol (TFTP)
Raw file loaded 0x00200000-0x007ab98f,
assumed entry at 0x00200000

RedBoot> g
```

At this point, you should see (Because -current changes frequently, the text you see may not match the example exactly.): Listing 2.

Once the Slug has located all of the hardware it starts the NetBSD installer, as shown in Figure 1. Pick your installation language, then follow the install procedure. There are two points where I sometimes have problems. NetBSD's disklabel is a little fussy; if it does not like the label that is already on the disk, it might refuse to do anything, and the installation will fail. Pop out of the NetBSD installation when you see: Listing 3.

You can clear out the previous disklabel using the technique detailed in the NetBSD documentation (*http:*

//www.netbsd.org/docs/guide/en/chap-misc.html#chap-misc-delete-disklabel):

```
# dd if=/dev/zero of=/dev/sd0c bs=8k
count=1
```

then restart the installer with:

```
# sysinst
```

A second problem I have encountered occurs when setting up NFS for the installer. Part of this setup involves entering the parameters for the Slug's network settings, including the address of your network nameserver. To test the network, the software will try to ping the nameserver. My nameserver, however, does not respond to pings so the installation software thinks that the network is not installed properly. You have, however, the option to continue on anyway. If necessary, you can exit the installation if necessary, and ping the NFS server directly to make sure the network is setup properly.

When the installation is completed, the installer may complain about missing installation sets. If you watched the installation and saw that files like *base.tgz*, *comp.tgz* and the like were installed onto your disk, you can proceed with the next step. Otherwise, select the upgrade option and make sure these files are installed.

Listing 4. Installing NetBSD on your NSLU2

```
RedBoot> load -r -b 0x200000 netbsd.bin
Using default protocol (TFTP)
Raw file loaded 0x00200000-0x004b2d67, assumed entry at 0x00200000
RedBoot> g
[ Kernel symbol table missing! ]
pmap_postinit: Allocated 9 static L1 descriptor tables
Copyright (c) 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005,
    2006, 2007, 2008
    The NetBSD Foundation, Inc.  All rights reserved.
Copyright (c) 1982, 1986, 1989, 1991, 1993
    The Regents of the University of California.  All rights reserved.

NetBSD 4.99.50 (NSLU2) #6: Fri Feb  1 20:34:39 EST 2008
        hayford@debian:/home/net/src/sys/arch/evbarm/compile/obj/NSLU2
total memory = 32768 KB
avail memory = 27048 KB
[...snip...]

boot device: <unknown>
root device:
use one of: npe0 sd0[a-h] sd1[a-h] ddb halt reboot
root device: sd0a
dump device (default sd0b): sd0a
file system (default generic):
root on sd0a dumps on sd0a
sd0: fabricating a geometry
mountroot: trying ffs...
root file system type: ffs
xrtc0: xrtc_clock_read: failed to read rtc at 0x0
xrtc0: xrtc_clock_read: failed to read rtc at 0x0
WARNING: preposterous TOD clock time
WARNING: using filesystem time
WARNING: CHECK AND RESET THE DATE!
init path (default /sbin/init):
init: copying out path `/sbin/init' 11
/etc/rc.conf is not configured.  Multiuser boot aborted.
Enter pathname of shell or RETURN for /bin/sh:
Terminal type? [unknown] vt102
Terminal type is vt102.
We recommend creating a non-root account and using su(1) for root access.
#
```

Booting NetBSD

The hard part is done; breath a sigh of relief and reboot your Slug again, stopping RedBoot as before. This time, boot the standard kernel (*netbsd.bin*): Listing 4.

Congratulations, you now have a working version of NetBSD running on your Slug! Note that you have to tell it what drive to mount as root, in my case `sd0a`. If you let the installer setup your disk, it will be the same for you as well. You can see from the memory sizes listed above that you now have 3 MB more available RAM then you did when you booted the installer, since the normal kernel does not include a memory disk.

Now that you are running NetBSD, go through the normal setup by adding a non-root user, and configuring the system the way you want. If you are new to NetBSD, take a look at The NetBSD Guide (*http://www.netbsd.org/docs/guide/en/index.html*). Another good thing to do is:

```
$ man afterboot
```

which will suggest a number of items to check or setup for a new system.

What Next?

I can think of several improvements to what we have done. Configuring the kernel to automatically set root as `/dev/sd0a` would keep the system from stopping and asking for the root drive during the boot process. Configuring both the install kernel and the normal kernel to use the secure shell daemon (`sshd`) would allow us to both install and run NetBSD without requiring the physical modification necessary to provide the serial port. And finally, having the installer write the bootup code to flash memory will make the whole boot process painless and automatic.

About the Author

Don Hayford is a Research Leader at Battelle Memorial Institute, specializing in the development of embedded hardware and software. He can be reached at: don@donhayford.com. In his spare time, Don enjoys fishing, cooking, and seeing new places.

OpenBSD pf – the firewall on fire

Girish Venkatachalam

In this article we will take a look at the king of firewalls – OpenBSD pf. pf has an interesting history as it was developed due to some interesting licensing problems with the firewall code in OpenBSD at that time. This happened well over six years ago. A genius named Daniel Hartmeier set out to hack the OpenBSD kernel in a beautiful Swiss town and something that began as a patch to the kernel to add firewalling functionality has today become the ultimate tool not only for performing the tasks of a firewall but also for QoS, spam control and traffic normalization.

Subsequently pf was ported to FreeBSD and NetBSD. For those familiar with Linux iptables, pf offers a refreshing change and a whiff of fresh air. OpenBSD is also a great OS with a lean kernel and astounding code quality. And pf makes an excellent addition to an already excellent networking subsystem. Though you can theoretically use pf for firewalling using any of the BSD OSes, I would advise you to run OpenBSD to enjoy pf and its power in its fullest glory.

We shall take a look at some of the most useful features that pf has. The code is evolving and the possibilities are endless but we have to start somewhere.

Before we dive into the details like configuration, a short overview will help put things into perspective. pf can do the following in addition to several other things.

- Filter network traffic based on UDP/TCP port numbers and host names.
- Specify filter in any manner – coarse of fine grained control is provided. For instance, you can specify that you can block all UDP traffic or all TCP traffic to/from a particular host.
- The rules, grammar and config file syntax are all so intuitive, user friendly and powerful that one wonders how simple things can get.
- Powerful queuing and traffic management with ALTQ subsystem. You can divide bandwidth consumption in your network in such a way that nobody abuses excess bandwidth.
- You can drop/accept packets arbitrarily based on a probability function. You can say that you will drop 20% of the incoming ssh attempts or ICMP probes. This can be really powerful when combined with other advanced functions.
- Control the rate at which TCP connections arrive, or the total number of simultaneous connections. If someone tries to connect to you too often you can block him using the 'tables' features I will be discussing below.
- Configure state table characteristics like timeout memory consumption and how often you want to refresh state tables and protocol specific timeouts.
- Do sophisticated NAT and other translation functions like transparent traffic reflecting with *rdr* rules. You can redirect traffic to a particular machine or even set of machines and ports to a different machines or ports. This is really powerful.
- Do redirection and routing based on filter parameters. For instance you can specify that packets from certain networks will go with a different source IP address when leaving the network while NAT happens or you could say that incoming packets will be redirected to different webservers for load balancing.
- And much more.

Unfortunately in this article I will not be able to cover all of this in enough depth. I will however give you enough rope so you can pick up and start exploring more. First some basics.

What is a firewall?

Good question. What is the need for restricting traffic? It is some kind of valve that we use in the physical world to control the flow of water for example.

With firewalls things can get exceedingly complex. As with all other things with computers, there are a lot of variables, parameters and possibilities.

And it is complicated further with the fact that nearly all the processing happens inside the OS kernel and in the present day's networking speeds, packets are pushed at gigabyte rates. Introducing filtering will surely slow things down. Unless this is done cleverly and with minimal overhead, things can go awry.

This is where the genius of Daniel Hartmeier comes in. He has written such excellent code, so well integrated with the rest of OpenBSD kernel and its networking subsystem and he also came up with sophisticated data structures like red black trees and Patricia trees for state tables and routing tables.

With the result that when it comes to raw performance, nothing today can match the throughputs that pf can give. Since everything is maintained in a state table, packet processing speeded up in a big way.

There is no need to hit multiple rules before passing a packet which is known to belong to a particular connection. There is no concept of connection in the UDP and the raw protocols over IP, but still pf tries its level best to guess thing based on source and destination IP addresses. So much for firewalling.

What is network address translation?

It is a very deep topic and I humbly request you to refer to the resources of this article for more information on this topic. In short non routable IP addresses defined in RFC1918 cannot be sent out to the Internet since the same range of IP addresses are used in private addresses around the world. So before the 'private' IP address leaves for the Internet, its source IP address has to be replaced with a globally unique routable IP address.

Similarly when packets come in, they have to be translated appropriately to make it reach the IP address in the private network. This diagram will put things in perspective see Figure 1.

Let us now get into how pf works. pf basic operation: see Figure 2. The diagram is fairly self-explanatory. I will delineate certain concepts so you can appreciate how things work. Only then you can use the tool effectively for solving real life problems.

Any firewall/router has to have at least two interfaces. Such devices are always at the frontier to the network. This is exactly like the guards or sentries you depute to guard forts.

In the case of pf, however, it has to take care of multiple gates or entry points. It is not just entry points it has to protect. It has to protect exit points as well. And pf can act at any time. At the time when traffic is coming in thro' any interface or when it goes out of any other interface.

It is one instance of pf code running inside the kernel that takes care of all this. Most real life installations I would suspect to have only two interfaces. One will be the external interface and the other will be internal.

Most of the filtering functions will happen on the external interface and redirections are more likely to happen in the internal interface.

But there are no fixed rules in these things. As far as pf is concerned interface is an interface – that is all. It could even be physical, virtual or one of the emulations of interfaces.

Either way, pf gets in the middle and does its job at the batting of an eyelid.

The main strength of pf I believe is the same as the strength of the great C programming language. It is economy of expression. Say a lot in few words. And also by leaving out the complexity and extra features, you can build really powerful rules in a simple manner. I shall demonstrate with a simple example to start with. A simple basic firewall see Listing 1.

This is a really simple firewall that performs NAT on the external interface. Now you can talk to the external world without buying an expensive router/firewall from a big company with a fat price tag.

You can do it for free with an old OpenBSD machine and pf. We are specifying that any packets with the source IP address range given in the $int_network can be allowed out through the external interface.

The variables defined with a = sign and referenced with the $ symbol are

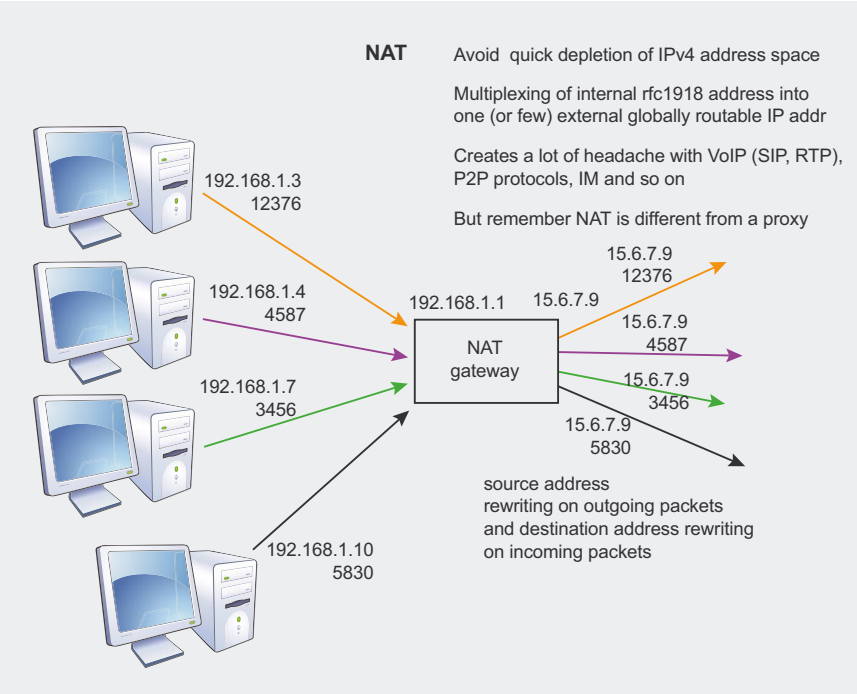


Figure 1. Network address translation

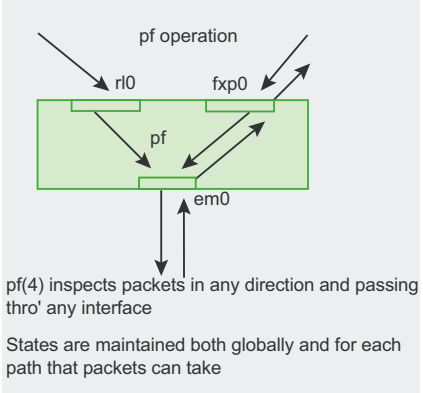


Figure 2. pf basic operation

called macros and help you remember rules and manipulate them easily. They also help you keep the ruleset unchanged when you change IP addresses, port numbers or other things even a network interface card.

You should also notice few other things from the above example.

You cannot specify rules or translation directives in any order. There is a fixed ordering mandated by pf. The traffic normalization rules, QoS , NAT and filter rules should always be specified in that particular order. And the packets also get processed in the same order.

This means that the filtering sees the packets that have undergone translation already: either NAT or redirection. pf ruleset optimization is an art that has to be learnt by experience. You can achieve huge performance improvements by simply reordering rules or leaving out the redundant ones.

But the biggest gain is by maintaining state. Starting from OpenBSD 4.1, all rules maintain state by default. This is a great feature.

Once things get stateful, the efficiency of filtering and rules processing get improved in a big way.

This is because it is the rules traveral that takes time and processing. If you have to pass through 50,000 rules it can take a long time. Instead if the packet hits a state defined by the 100th or the 1000th rule, the rest of the rules are bypassed.

You should also remember to use the 'quick' keyword in specifying rules. This keyword is a special directive to tell pf that if this rule is matched, then you can straight away pass or drop the packet without looking further down.

As with everything else, the devil is in the details and you will become a master with experience. You can also tweak a lot of things like the memory used by the firewall machine for maintaining states, the timeout parameters for TCP connection closure and dropped connections and so on.

Let us now look at a slightly advanced example. Before we wind up. Using pf to control incoming ssh connection attempts:

- As I mentioned in the beginning, pf can measure TCP connection rates and restrict a single host or multiple hosts from connecting too often to us.
- This rule comes under the category of source tracking.
- Obviously this will work only when you maintain state.
- For instance, pass in on $ext_if from any to port http modulate state max-src-conn 40 will allow only 40 simultaneous TCP connections to arrive at the external interface to our web servers.

Pay particular attention to the *modulate state* parameter. This is the same as *keep state* parameter which is default now. But with a twist. pf modulates TCP sequence numbers to protect us against session hijacking and man in the middle attacks.

What use is a firewall tool if we cannot monitor packet statistics? Of course pf has a very advanced mechanism for monitoring and administration of the firewall.

pf administration

The most important tool for this purpose is an appropriately named executable called pfctl. For example,

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

will parse the pf config file and load the rules into memory. You can enable the filter and disable it with:

```
# pfctl -e
```

and

```
# pfctl -d
```

You can flush all states with:

```
# pfctl -Fa
```

Or you could inspect all the statistics with:

```
# pfctl -sinfo
```

If you specify `-vv`, you get even more information with the `-i` flag.

OpenBSD ships with a modified version of tcpdump that can read the packet dumps produced by pf. There is a daemon called pflogd which reads packets in tcpdump format and writes to a log file from `/dev/pflog0`.

pf also has mechanism to synchronize states between multiple firewalls connected physically in the same network. This happens through a protocol over IP called pfsync.

OpenBSD also has a redundancy protocol called CARP and using pfsync, you can build a redundant firewall architecture. As you have seen, pf is really powerful. And no mistake.

On the 'Net

- NAT traversal and P2P traffic: *http://linuxjournal.com/9004*
- OpenBSD pf faq: *http://www.openbsd.org/faq/pf*
- pf.conf(4) man page

Listing 1. A simple basic firewall

```
# cat /etc/pf.conf

int_if=udav0
ext_if=fxp0

int_network= { 192.168/16, 10/8 }

block in

nat on $ext_if from ! ext_if to any -> $(ext_if):0
pass out on  $ext_if from $int_network to any
pass in  to port { 25, http, rsync } from any
```

USENIX Upcoming Conferences

The Sixth International Conference on Mobile Systems, Applications, and Services (MobiSys 2008)

Jointly sponsored by ACM SIGMOBILE and USENIX

JUNE 17–20, 2008, BRECKENRIDGE, CO, USA
http://www.sigmobile.org/mobisys/2008/

2008 USENIX Annual Technical Conference

JUNE 22–27, 2008, BOSTON, MA, USA
http://www.usenix.org/usenix08

2nd International Conference on Distributed Event-Based Systems (DEBS 2008)

Organized in cooperation with USENIX, the IEEE and IEEE Computer Society, ACM SIGSOFT, and ACM SIGMOD

JULY 2–4, 2008, ROME, ITALY
http://debs08.dis.uniroma1.it/

2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)

Co-located with USENIX Security '08

JULY 28–29, 2008, SAN JOSE, CA, USA
http://www.usenix.org/evt08

2nd USENIX Workshop on Offensive Technologies (WOOT '08)

Co-located with USENIX Security '08

JULY 28, 2008, SAN JOSE, CA, USA
http://www.usenix.org/woot08
Submissions due: June 1, 2008

Workshop on Cyber Security Experimentation and Test (CSET '08)

Co-located with USENIX Security '08

JULY 28, 2008, SAN JOSE, CA, USA
http://www.usenix.org/cset08

17th USENIX Security Symposium

JULY 28–AUGUST 1, 2008, SAN JOSE, CA, USA
http://www.usenix.org/sec08

3rd USENIX Workshop on Hot Topics in Security (HotSec '08)

Co-located with USENIX Security '08

JULY 29, 2008, SAN JOSE, CA, USA
http://www.usenix.org/hotsec08

22nd Large Installation System Administration Conference (LISA '08)

Sponsored by USENIX and SAGE

NOVEMBER 9–14, 2008, SAN DIEGO, CA, USA
http://www.usenix.org/lisa08

Symposium on Computer Human Interaction for Management of Information Technology (CHIMIT '08)

Sponsored by ACM in association with USENIX

NOVEMBER 14–15, 2008, SAN DIEGO, CA, USA
http://www.chimit08.org

ACM/IFIP/USENIX 9th International Middleware Conference (Middleware 2008)

DECEMBER 1–5, 2008, LEUVEN, BELGIUM
http://middleware2008.cs.kuleuven.be

8th USENIX Symposium on Operating Systems Design and Implementation (OSDI '08)

Sponsored by USENIX in cooperation with ACM SIGOPS

DECEMBER 8–10, 2008, SAN DIEGO, CA, USA
http://www.usenix.org/osdi08

7th USENIX Conference on File and Storage Technologies (FAST '09)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

FEBRUARY 24–27, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/fast09

First USENIX Workshop on Hot Topics in Parallelism (HotPar '09)

SPRING 2009, LOCATION TBD
http://www.usenix.org/hotpar09
Submissions due: October 17, 2008

Instant Messaging with Jabber/XMPP

Eric Schnoebelen

Instant Messaging-What a Concept! To be able to talk, (or chat as it is commonly called), to someone on the other side of the planet in real-time. WOW! In the 1970's, when instant messaging first appeared, it was used as an inter office tool, for users of systems like UNIX, to leave messages for others using the same machine, working it way up to a device to communicate with others in the local network.

Who would have thought that from those humble beginnings, Instant Messaging would become a way of life for most of the people on earth?

There are many different IM (Instant Messaging) programs – AIM and Yahoo being the most commonly recognized. There is one though, a little less re-cognized, but much more efficient, JABBER. When Jeremie Miller first released the code for Jabber in 1999, I am sure he did not have any idea just how big it would become. With over 40 million users, on several different operating systems like Windows, Linux, and Macintosh it makes it one of the most versatile programs out there. The IETF approved the Jabber protocols in 2004 under the name XMPP (*Extensible Messaging and Presence Protocol*), and today companies such as IBM, Apple, Sun, Nokia,Sony, Digiun and Psion all support XMPP. XMPP is used as the official internal instant messaging system with in companies such as Hewlett Packard,EDS, and many others.

Some of the many advantages to running Jabber are that it is an ad free, open and secure alternative to to other services such as AIM,ICQ,MSN and Yahoo. (Plus, using transports, you can connect to all your friends on these services through your Jabber accounts). Jabber is understandable, and proven effective, with tens of thousands of Jabber servers running around the world. The applications associated with Jabber include network management, collaboration tools, file sharing, gaming, remote systems and content syndication. Jabber can be set aside from the public networks and used for things such as intra office communications, having built in security features like SASL and TLS built into the core specifications.

There are a variety of client and server implementations for Jabber/XMPP. Almost every operating system has at least one client implementation and most have a server implementation. A very incomplete list of clients: Listing 1. The (very incomplete) list of servers include: Listing 2.

Building an XMPP/Jabber server

In todays installment, we are going to install and configure one of the jabber servers. When we are done, you will have a fully operating jabber server capable of joining the federation of jabber servers on the public Internet. Things you are going to need:

- Access to your DNS configuration (to install SRV records)
- A compiler (duh.. :D)
- The jabberd 2.1.19 source code
- Root access to install any prerequisite packages. (or the willingness to muck around installing packages in a non-default location as a non-root user.)
- The prerequisite packages/sources.
- A jabber client program. On UNIX, Psi and Pigdin are good. See above for more clients, or *http://www.jabber.org/clients*.

The packages that jabberd 2.1.19 depends upon are:

- expat
- OpenSSL
- libidn
- libiconv
- gnusasl
- sqlite

The configuration we are going to use will use an sqlite database, to avoid the complexities of setting up a real relational database server such as MySQL, Oracle, or PostgreSQL. If you are going to going to be having a lot of users, you'll definitely want to read up on setting up the relational database, and having `jabberd2` use it.

Personally, I much prefer to use a package manager of some sort to build/maintain my software. The packaging system of preference here is the pkgsrc system (*http://www.pkgsrc.org*), which grew out of the NetBSD project but now supports 14 different operating systems.

To build/install using pkgsrc, use the package wip/jabberd2 (currently at version 2.1.19), found on as part of the pkgsrc-wip project on sourceforge. Hopefully, by the time this is printed, 2.1.19 will be migrated into the mainstream pkgsrc as chat/pkgsrc,and an even newer version will be hanging out in wip.

To build from pkgsrc, change directory to the appropriate jabber server, wip/jabberd2 or chat/jabberd2, and type (as root) *make install clean-depends clean*. pkgsrc will proceed to install/update all dependencies, and then build and install the jabber server.

I will stop plugging pkgsrc now..

Packages almost certainly for other operating systems and environments, including FreeBSD's ports system (im/jabberd2). I was unable to find the expected Debian or RPM packages,which is surprising.

To build from source, it is pretty much the old *configure; make; make install* routine.

jabberd2 wants a non-privileged user to run the executables,so creating a jabberd user and group would be a good idea about now.

After extracting the tarball, use the following options to configure. The point here is to put all the files in (roughly)the same place. See Listing 3.

Notes

The LDFLAGS have been set to hard code in the runtime library path for the supporting packages provided by the pkgsrc system on this computer. `--with-extra-include-path` and `--with-extra-library-path` exist for the same reasons. Had I built the supporting packages by hand, and installed them into `/usr/local`, those paths would read `/usr/local/{include,lib}`.

Had the prerequisites been provided by the base system,they would be unneeded.

Now for the traditional `make && sudo make install` dance.

Configuration

DNS Configuration. Jobber/XMPP requires a valid, working DNS configuration. Since federation between XMPP servers occurs using domain/host names,valid mappings must exist in DNS.

A publicly accessible IP address is also required, as part of inter-server communication is *dial-back* where the remote server looks up the requesting server's name and contacts the provided IP address(es).

It should go without saying (but I will say it anyway), XMPP servers require a public IP address.

An installation can *get away* with just having a DNS A record for the jabber server, and using that as the domain/hostname. However, one of the largest XMPP sites in the world, GoogleTalk, requires that DNS SRV records for the server exist for it to *dial-back* the server.

An example of DNS SRV records for XMPP are the following (from the local live installation):

```
_xmpp-server._tcp.cirr.com IN SRV
5 0 5269 jabber.cirr.com.
_xmpp-client._tcp.cirr.com IN SRV
5 0 5222 jabber.cirr.com.
```

The above configuration allows Jabber ID's such as *eric@cirr.com* to be routed to *jabber.cirr.com* for handling.

If your DNS provider does not support SRV records, ask them to do so, or change providers. Or, be willing to accept that you won't be able to connect with users on Google Talk.

First up, the hostname to use should be decided upon. A past convention is `jabber.<domain>.<tld>`. With DNS SRV records, it can be simply `<domain>.<tld>`.

Firewall configuration

XMPP requires open communications on two TCP ports. The clients communicate with the server on port 5222, while the servers talk between each other on 5269. Those ports need to be open inbound to allow both clients and servers to connect.

jabberd software Configuration

With that information, it is time to start working our way through the configuration files of jabberd2. All the configuration files are XML (not entirely surprising, since XMPP itself is an XML stream based protocol).

The configuration files are in `/usr/local/etc/jabberd` (if you have built from scratch) or `/usr/pkg/etc/jabberd` (if you are using `pkgsrc`.) They are *router.xml*, *router-users.xml*, *router-filter.xml*, *resolver.xml*, *c2s.xml*, *s2s.xml*, and *sm.xml*.

Listing 1. Clients list

```
pigdin        -- Open Source, runs on MacOS, Linux, Unix, Windows
psi           -- Open Source, MacOS, Linux, Unix, Windows
jbother       -- Open Source, written in java
iChat         -- MacOS component
Exodous       -- Freeware, runs on Windows
```

router.xml:

The router process is the core of `jabberd2`. It routes XMPP packets between the various components of the server, and the outside world.

The only thing that *must* be changed in this file is the shared secret (at `<router><local><secret></secret></local></router>` in the XML) used by legacy components. It defaults to *secret*. You want to change it, or *rogue* components could connect to your server, and route packets through it.

Another parameter to consider altering is the `<router><check><keepalive></keepalive></check></router>` parameter. For every connection/session that has been idle at least `<keepalive/>` seconds, a single whitespace character is sent. This allows the server to learn of broken TCP sessions clients and peers, and allows resetting of various idle timers on NAT type devices between the servers and it is clients. Somewhere between 5 and 15 minutes (300 and 900 seconds) seems like a reasonable idea. The default is 0, or no `<keepalive/>`s.

A third parameter to review and consider altering is the `<router><log></log></router>` clause. By default, jabberd `2.1.19` logs via syslog at `LOG_LOCAL3`. This can be changed to another facility, or may be changed to log directly to a file. If logging via syslogd, now is the time to configure syslogd to route those messages someplace useful.

The final parameter to consider changing is the `<router><local><pemfile></pemfile></local></router>` parameter. It is the path to the SSL certificate to be used by the server to communicate with peers and clients. XMPP will work just fine over a non-SSL channel, but also supports SSL (configured via TLS) for secure communications.

router-users.xml:

router-users.xml is the password file for the jabber server. It defines the user names and passwords the various components use to connect to the router.

A minimal change is to change the shared secret. It is in the xml element `<users><user><name>jabberd</name><secret></secret></user></users>` and defaults to *secret*. Change it. You will use that in the rest of the jabberd components. Per component username/password pairs may also be defined, although this is not necessary.

router-filter.xml:

router-filter.xml is *packet filter* for XMPP communications within the server, and with external servers and components. At the moment, largely uninteresting, but as experience is gained, may prove useful for limiting troublesome users and sites.

resolver.xml:

resolver.xml configures the DNS resolver process. All the other components hand off DNS name resolution to the resolver to avoid using blocking resolver library implementations.

The first configuration item(s) to alter are `<resolver><router><user></user></router></resolver>` and `<resolver><router><pass></pass></router></resolver>`. These should match entries provided in router-users.xml.

The next item to consider is `<resolver><log></log></resolver>`. See the `<log/>` discussion for router.xml, it applies here too.

The final item to consider is `<resolver><router><pemfile></pemfile></router></resolver>`. See the corresponding discussion for router.xml.

c2s.xml:

c2s.xml configures the client interface component within the server. All clients initially connect to this component.

As might be expected by now, the first configuration item to be altered is `<c2s><router><user></user></router></c2s>` and `<c2s><router><pass></pass></router></c2s>`. These should match the entries in router-users.xml.

The next items are `<c2s><log></log></c2s>`, `<c2s><router><pemfile></pemfile></router></c2s>` and `<c2s><local><pemfile></pemfile></local></c2s>`. c2s has two `<pemfile></pemfile>` blocks, one for communicating with router, and the other for communicating with the XMPP clients.

A new item requiring attention is the `<c2s><local><id></id></local></c2s>` block. This segment is used to configure the name of the local jabber instance.

This is also known as the servers JID, Jabber Identifier. The server's JID should be listed in the `<id></id>` block. Other parameters can be defined with in the opening `<id>` tag. The text block in c2s.xml describes the choices.

If the server is expected to handle a large number of clients concurrently, `<c2s><io><max_fds></max_fds></io></c2s>` should be increased from the default value. (A separate relational database, such as PostgreSQL or MySQL should also be considered for the data store).

A familiar item that you may wish to update is `<c2s><check><keepalive></keepalive></check></c2s>`, which acts

Listing 2. Servers list

```
djabberd
ejabberd      --  written in Erlang, Open Source runs on Windows, Linux, Unix, MacOS
jabberd 1.x   --  written in C, Open Source runs on Windows, Linux, Unix, MacOS

jabberd 2.x   --  written in C, Open Source runs on Windows, Linux, Unix, MacOS
Jabber XCP    --  commercial product, runs on Windows, some Unix-en
Merak

Openfire      --  written in Java, Commercial & Open Source runs on Unix, Linux, Windows
SoapBox Server
Sun Java System IM
Tigase
```

like a keep-alive between c2s, and the XMPP clients.

A new block in the configuration file (relative to the other configuration files) is `<c2s><authreg></authreg></c2s>`. This controls the database back ends used to contain the authorization information. Since we're using sqlite, all the other back ends should be removed from this declaration.

Set `<c2s><authreg><module></module></authreg></c2s>` to be *sqlite*. Verify the `<c2s><sqlite><dbname></dbname></sqlite></c2s>` file pathname is correct for the planned installation. FYI: the comments at the top of the SQL script for creating the sqlite database suggest one name for the database file (jabberd2.db), while the xml configuration files use a different one (sqlite.db).

s2s.xml:

s2s.xml configures the server-to-server component, s2s, in the jabberd `2.1.29` server. s2s manages the connections between local components and other XMPP servers.

As should be expected by now,`<s2s><router><user></user></router></s2s>`and`<s2s><router><pass></pass></router></s2s>` need to be updated to match the contents in router-users.xml. Optionally, `<s2s><router><pemfile></pemfile></router></s2s>` should be updated as well, followed by `<s2s><log></log></s2s>`, and `<s2s><check><keepalive></keepalive></check></s2s>`.

There are two sets of `<pemfile></pemfile>'s`, one for communicating with router (`<s2s><router><pemfile/>`), and another for communicating with the remote servers (`<s2s><local><pemfile/>`).

sm.xml:

sm.xml configures the session manager component of the jabber server.

sm.xml defines, as the `<sm><id></id></sm>` the domain portion of the users JID. The hostname/domain name within this block *must* be resolvable on the public internet, if you wish other XMPP servers to contact you.

The now-standard tags need to be updated. `<sm><router><user></user></router></sm>` and `<sm><router><pass></pass></router></sm>`, `<sm><router><pemfile></pemfile></router></sm>` for connecting to router, and `<sm><log></log></sm>` for the choosen logging style.

A new clause is `<sm><storage></storage></sm>`. It defines the database back end to be used to store off line messages, and other information. Since we're using sqlite, all others should be removed. Set the `<sm><storage><driver></driver></storage></sm>` to be *sqlite*. Verify the path to the database space (`<sm><storage><sqlite><dbname>`), and alter as necessary.

Listing 3. Configuration

```
#!/bin/sh
export LDFLAGS=-R/usr/pkg/lib
./configure --prefix=/usr/local \
            --disable-db \
            --disable-ldap \
            --disable-mysql \
            --disable-pam \

            --disable-pgsql \
            --enable-anon \
            --enable-fs \
            --enable-pipe \
            --enable-sasl=gsasl \
            --enable-sqlite \
            --enable-ssl \

            --sysconfdir=/usr/local/etc/jabberd \
            --without-libiconv-prefix \
            --without-libintl-prefix \
            --with-extra-include-path=/usr/pkg/include \
            --with-extra-library-path=/usr/pkg/lib
```

Having set up all the configuration files, now is the time to initialize the database spaces. jabberd 2.1.19 comes with a set of SQL scripts to initialize the database for the different back ends. Since we're using sqlite, as your jabber user, execute

```
``sqlite3 /var/db/jabber/sqlite.db < \
        /usr/pkg/share/examples/jabberd/db-setup.sqlite''
```

(assuming the pkgsrc paths, if you are not using pkgsrc, the db-setup.sqlite script will be in tools subdirectory of your jabberd 2.1.19 distribution.)

The moment of truth

Now comes the moment of truth. Starting the server for the first time. As your jabber user, execute `jabberd`. It will use `/usr/pkg/etc/jabberd/jabberd.cfg` to start the components in the correct order. If everything is correctly configured, the jabberd program will not return, and you will find c2s, s2s, router, resolver and sm running as the jabber user.

If it does not come up immediately, read the log file(s) and follow the hints provided by the error messages there.

Now, to connect to your newly created jabber server, using a jabber client. You'll have to create an account, which most of the common jabber clients support. (it is called `in-band registration`.)

After you have had fun talking to yourself for a little bit, convince the spouse(equivalent) or a friend to set an account for themselves, and play a bit on your private IM server.

To see verify if your new XMPP server is configured properly to federate, try connecting to *howie@jabber.cirr.com*. Howie is an Eliza-like program, responding to messages sent to it over the public XMPP network.

If you get a response from howie, then you're federated with the rest of the public network.

Future topics

In future articles, we'll discuss how to configure jabberd to use SSL certificates; how to build and install *transports* to allow jabber users to connect to the proprietary networks; building some of the clients; and build build jabber robots, such as *howie@jabber.cirr.com*.

Interview with FreeBSD developer Jeff Roberson

One of the major improvements in FreeBSD 7.0 is the performance increase that every owner of MP machines is experiencing. How did they get such a great result?
I have interviewed Jeff Roberson, the creator of the ULE scheduler, to learn more about the work done by FreeBSD developers in SMP land.

Could you introduce yourself?

I am *jeff@freebsd.org* and I have been a committer since 2002. I have worked on the kernel memory allocator, filesystems, SMP, scheduling, threading, and other bits and pieces. I have been writing software professionally for 10 years starting at Microsoft working on what would become Windows 2000 in the networking group. I presently am an independent contractor specializing in high-performance FreeBSD and Linux kernel projects. My wife, Christine, and I live on the island of Maui in Hawaii where I do some competitive bicycle racing and we both enjoy hiking, the beach, etc.

What is changed in SMP land from 6.x to 7.0?

There were many significant improvements in SMP for 7.0 and I am sure I will miss some. Attilio Rao re-implemented sx locks which combined with work by John Baldwin, Robert Watson, and myself lead to the most significant gains for multi-threaded applications simply be replacing the lock on the file descriptor table with a modern primitive. A lot of work was driven by Kris Kennaway who runs the ports build cluster and also does excellent data gathering using our lock and code profiling tools. Robert also improved unix domain socket scalability and Scott Long has removed Giant from cam, our scsi subsystem. I got back on the horse again with ULE and got some impressive gains there now that the kernel is more finely locked.

I also broke up our single scheduler lock into per-cpu run-queue locks, sleep-queue locks and turnstile locks. The default threading library was switched to libthr, which is our 1:1 threading library. This again had a significant performance impact, especially on larger machines. Robert and Kip Macy have been looking heavily at network contention on 10 gigabit networks and we have seen some improvements there as well.

In addition to these things that I pointed out specifically there are dozens of other things which I might not be as aware of or involved in that contribute to the whole. The reason ULE, my scheduler, is able to now show significant gains on some workloads is because the kernel is now scalable enough to allow the concurrency.

Did you completely remove the Big Giant Lock?

We have eliminated Giant from all of the common cases. However, there are some legacy drivers and filesystems which still require it. I believe the network stack is now 100% free as are nfs client/server, ffs/ufs, and all mainstream storage drivers. The cases that remain are unlikely to cause performance problems although it does increase the complexity of the kernel because we have several compatibility layers to deal with these old drivers.

The SMPng effort took a lot of years and had different phases, 5.x, 6.x, and now 7.x. After all this work, you can now focus on perfomance improvements and optimizations. Reading the mailing lists, for example *performance@freebsd.org*, it seems you have now a system that can be rapidly improved. How is this possible? Good design at the beginning of SMPng?

I believe the work required to convert a large codebase to SMP is quite often underestimated. However, FreeBSD all along has stressed good techniques and a solid foundation ahead of quick results. FreeBSD 5.x provided a foundation and a model for doing SMP which is a derivative of BSD/OS which again is a derivative of the Solaris model. We created interrupt threads, locking primitives, and infrastructure for automatic verification

of lock order and use. We did this knowing we would be slowing the system until the next phases could be complete.

In 6.x we removed the giant kernel lock from much of the system. So finally in 7.x we are able to optimize as developers have increasingly had access to 4-8 core machines. We have actually done fairly well in tracking releases with readily available hardware. The vast majority of systems are 1 or 2 sockets and while 5.x and 6.x were not fantastic for SMP scalability they performed well enough on commodity machines. And now 7.x is out in time to meet the demands of multi-socket quad-core chips.

From the benchmarks you run, which subsystem showed the best performance improvements?

We focused on the sysbench benchmark using both mysql and postgresql as servers. FreeBSD 6.0 was extraordinarly underperforming on quad CPU systems and greater. So we started using the mutex profiler and quickly identified a number of issues that ended up giving something like a 500% performance boost on an 8 way machine. After about a month of a couple of people hacking in their spare time we are regularly faster than Linux in a variety of machine configurations. As far as the subsystems involved, it was a number of upper layer kernel services like file descriptors, unix domain sockets, and lower level things like scheduling.

Looking at the benchmarks it seems that FreeBSD 7.0 has reached some incredible scalability results! What happened when Linux folks saw FreeBSD 7.0 outperforming Linux on an 8-core machine running PostreSQL and then MySQL?

We at first were not certain that the results were valid because Linux has such a good reputation for scalability and the initial results were so bad. Specifically, on an 8 core machine, once we had more than ~12 mysql threads running the performance dropped to that of a single cpu machine.

I have had a relationship with some Linux kernel developers as I often do contract work that involves Linux. I brought our results to their attention in an attempt to validate them and understand what was going on. At first I got the boilerplate *Your test is broken*, *MySQL is broken* denial. However, eventually they seemed thankful and resolved some serious bugs of their own. While the developers are typically very courteous, with perhaps a little bit of friendly competitiveness, the Linux user response has been overwhelmingly negative. I have received all manor of hate mail and slander via my blog and email since publishing these results.

What has happened since this initial finding is a very healthy back and forth with Linux as each team optimizes their system further. As long as we all stay friendly I think this will be good for both projects. We have shared results and ideas and hopefully this will continue.

Why did FreeBSD 7.0 outperform Linux?

In this specific MySQL test there are many factors. Initially Linux had slightly better performance at peak but scaled very poorly. This was due to lock contention in their kernel. At this moment we still scale better and I believe that is due to better scheduling decisions. It seems clear that their kernel is not hitting any significant lock contention in this workload and neither is ours. The introduction of CFS has actually cost them about 20% on this test on 8 core machines so until that stabilizes it will be hard to compare.

I think the important message with this release is that we're no longer behind the curve on scalability and we can go toe to toe with Linux and perform as well or better. There will be workloads where we fall behind and others where we are head but bad SMP performance is no longer a reason to avoid FreeBSD.

How much is the scheduler important for the performance of a system?

On a uniprocessor the scheduler really only has opportunities to make performance worse by context switching more frequently. However, multiprocessor systems with complex cache structures and topologies provide opportunities for the scheduler to actually improve performance by placing related threads near to each other or better distributing work among cores.

If you compare a modern system using CPU affinity and one without you can see differences of 30-40%. So that is quite a large impact. Switching from O(1) to CFS in Linux costs them ~20% on one benchmark. So you can see in this simple case good scheduling decisions have a rather large impact.

The trick most of the time is to schedule infrequently enough to get good performance, but frequently enough and with the right choices to get good behavior. It turns out that this is a very difficult task and one with many competing optimizations. Things that improve one workload will often harm another.

It is important to point out that the scheduler is just one part of a whole that determines the system behavior. Early in ULE's development CPU affinity did not help as much as I had hoped but this really turned out to be because everything was contending on Giant in the kernel. With all workloads serialized there was little gain to be had from smarter scheduling. So ULE is only able to perform well now due to the hard work of many other people in making a scalable kernel.

Which scheduler will be the default in FreeBSD 7.0? Why?

One thing that I really like about FreeBSD is that it is a relatively conservative and very well planned operating system. We have somewhat regular releases that are controlled by a release engineering team that have one of the most difficult and thankless jobs in the project. Our release engineering team and our developers had a long discussion about ULE as a default and decided that it was too soon for 7.0. This is not based on any negative observation of ULE's performance or stability but only because it was proposed too late in the release cycle to make such a significant change. If there are problems we do not want regular users to discover them after installing 7.0. So it is now the default in current, again, and will probably be the default for 7.1.

FreeBSD in general is very cautious about change. People get excited about flashy new features but they must be well understood and well vetted before they are accepted. ULE was the default in 5.x for a period as well but it turned out to be too soon. Not having it the default actually takes some pressure off of me and gives me more latitude in developing it, however, I do hope to see people benefiting from my work as well.

by *Federico Biancuzzi fed@bsd.it*

What is in a certification?

Mikel King

Recently the BSD Certification Group celebrated its official *corporate* 2nd anniversary, without party hats, cake or other fan fare. Although not a monumental milestone, it is still one worth noting that during these few years all of the BSD OSes reflected on the exam have had at least one major release. Yet despite the ever changing OS landscape the group has successfully, crafted an exam for certifying entry-level BSD Unix Systems Administrators, firmly establishing this as one of the most unique Open Source projects.

From it is inception the BSD Certification has been an open community supported effort that truly sets BSD Certification group's work apart from any other certification system. Throughout it is few short years the team collected data from a multitude of surveys to define the 7 knowledge DOMAINs as outlined on the site *http://www.bsdcertification.org/index.php?NAV=Certification*. Additionally the group defined what an entry level administrator is and drafted a clear set of skills required by each candidate. Finally building this into a BAMP based system to store the questions drafted by the *Subject Matter Experts* (SMEs) for the subsequent exams.

To give you an idea of what a question might look like I asked an associate of mine (Michael Hernandez) who is familiar with OpenBSD to take a stab at writing a question for one of the DOMAINs. Please understand that Mr. Henandez has absolutely no affiliation with the BSD Certification group, and this is just an arbitrary question.

Joe, an admin for somecorp, inc., is logged in as root on an OpenBSD 4.2 system. He is having trouble opening up `archive.tar.bz2` via the command `tar xjf archive.tar.bz2`. The error he gets says: `tar: unknown option j`. This is most likely because:

- `bunzip2` is not installed and so tar cannot decompress `.bz2` files
- `gzip` is not installed and so tar cannot decompress `.bz2` files
- OpenBSD does not support `bunzip2` and therefore cannot decompress `.bz2` files
- The version of tar included with OpenBSD does not support the `j` option for decompressing `.bz2` files

Once the question database was full of these sorts of questions, came the rather grueling process of verifying these questions for accuracy and relevancy. Furthermore, in order to ensure properly proctor-able exam there had enough questions covering each DOMAIN to afford the correct level of randomness and objectivity in the testing system. In that light examining this question more closely you will note some ambiguous information as well as superfluous content. The author does side step some of this by tossing in the closing phrase *This is most likely because*: However I think with a few minor changes we can make this more specific. After completing a new installation of OpenBSD you have been asked to install your company's utility packages. However, the command `tar xjf überCorp.utilities.tar.bz2` fails with an *unknown option j* error message. Select the most likely cause from the following:

- `bunzip2` is not installed and so tar cannot decompress the file
- gzip is not installed and so tar cannot decompress the file
- You forgot the – in front of the arguments
- The version of tar included with OpenBSD does not support the `j`option

Essentially the same question only worded a bit more cleanly. In order to complete that rewrite SMEs familiar with DragonBSD, FreeBSD, NetBSD as well as OpenBSD would have to meet on this question to rephrase it in a way that would satisfy the certifications core objectives. Assuming that the group was satisfied with this questions specificity then the question would pass onto the beta proctoring phase. Volunteers from the community who actually took the beta exams helped establish the necessary a baseline for the production exam. As of this writing there have been several workshops held by the SMEs to further refine the questions and prune the ones that were inappropriately/confusingly phrased, too specific, too difficult, or even too easy. In some cases questions had too many correct answers or worse none at all.

Finally as the BSD Associate exam entering it is last stages of beta testing and review I felt it was appropriate to take a moment to reflect on the team's achievements. As well as expose all of the hard work that the members put forth in creating this product. From the fundraising efforts to cover the costs of he professional psychometric revue and proctoring expenses and the efforts of the translation teams which will bring the exam to another 20 languages, there is quite a bit of work left to complete. Assuming that all goes according to plan the certification will be ready as planned for 2008, and then the team will jump right into development of the BSD Professional certification.

This BSD Certification project is not about creating some new application, yet it will yield a tangible product just the same. One that I feel the entire community can take great pride in helping to achieve. I hope that I have shed some light on the work of a very worthwhile BSD project.

About the author

Mikel King has been working in the Information Services field for over 20 years. He is currently the CEO of Olivent Technologies, a professional creative services partnership in NY. Additionally he is currently serving as the Secretary of the BSD Certification group as well as a Senior Editor for Daemon News.

great subscriber offer

SAVE $20!

Get your copy of BSD Magazine and save $20 of the shop price

Three easy ways to order

- *visit: www.buyitpress.com/en*
- *call: 001 917 338 3631*
- *fill in the form below and post it*

Why subscribe?

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

BSD Magazine ORDER FORM

☐ **Yes**, I'd like to subscribe to
BSD Magazine from issue ☐ ☐ ☐ ☐
1 2 3 4

Order information
(☐ individual user/ ☐ company)

Title

Name and surname

address

postcode

tel no.

email

Date

Company name

Tax Identification Number

Office position

Client's ID*

Signed**

Payment details:

☐ USA $39.99
☐ Europe 29.99€
☐ World 29.99€

I understand that I will receive 4 issues over the next 12 months.
Credit card:
☐ Master Card ☐ Visa ☐ JCB ☐ POLCARD
☐ DINERS CLUB

Card no. ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐ ☐☐☐☐
Expiry date ☐☐☐☐ Issue number ☐☐
Security number ☐☐☐

☐ I pay by transfer: Nordea Bank
IBAN: PL 49144012990000000005233698
SWIFT: NDEAPLP2

Cheque:
☐ I enclose a cheque for $ _____
(made payable to Software-Wydawnictwo Sp. z o.o.)

Signed

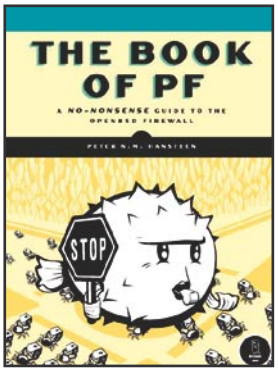
Terms and conditions:
Your subscription will start with the next available issue.
You will receive 4 issues a year.

* if you already are Software-Wydawnictwo Sp. z o.o. client, write your client's ID number, if not, fill in the chart above
** I enable Software-Wydawnictwo Sp. z o.o. to make an invoice

The Book of PF

by Peter N.M Hansteen

Biased review ahead



This review is going to be biased. First of all I love OpenBSD, I love PF and I have meet Peter who is a nice guy to talk to. But we are getting ahead here. This book is obviously about PF, what is that? PF is the Packet Filter developed for OpenBSD and then ported to several other BSD systems. PF is a modern firewall system which performs great, like many others, but which has a built-in language which makes it very easy to understand the ruleset and create a better firewall.

Note

To be fair the filtering language of PF was in the first versions very similar to the IP Filter by Darren Reed. Credit goes to him for making IP Filter in the first place, I learnt a lot about firewalls from using it. As explained in the book PF was actually the child of need when IP Filter was removed from OpenBSD. So PF was invented and at some time Peter Hansteen wrote his famous web page *Firewalling with OpenBSD's PF packet filter*. From this source he has then managed with help from No Starch Press to produce an important book about the best firewall for Open Source systems.

Compared to web page version

With this source the first question from a potential reader might be, how does it compare to the web page. Why should I buy this when I can download and print. The content of the book is arranged similarly to the web page, but better. The layout is better since the people at No Starch knows how to layout pages and the typography which makes reading a pleasure. Peter has also written new paragraphs and introductory sections which are much better and makes the overall reading from cover to cover better. So to answer the question: the book is way better than the web page and easier to read. Further the format, a book, as compared to printed paper is much nicer when sitting at home reading or as I did when you bring the book along to read a chapter.

Contents

Since not all have read the web page I will try to summarize what the book is about, and why it does matter as an extension of the current available reference and other information about PF. The book is about PF, and not only about PF on OpenBSD. Since Peter uses PF on OpenBSD he does remind people that not all features are available on FreeBSD and NetBSD – but this book is not just about OpenBSD – it really is about PF. The chapters of the book goes from enabling PF with the simplest possible rulesets on OpenBSD, FreeBSD and NetBSD through expected firewall/gateways to advanced networks like: wireless networks, bigger networks with DMZ subnets, bandwidth shaping with ALTQ and even logging and statistics. Judging from the number of pages it should not be possible, the book is only about 150 pages, but the way Peter has organized it makes it possible.

Writing style

Peter has a unique writing style and be warned, I do not think everybody will enjoy it, unless prepared for it. This book is not a HOWTO with complex and magic instructions which you can follow and not learn from. This book is about educating you the reader to become the local PF guru by having a master guide you onto the path and pushing you forward.

What you need to succeed with this book is access to a computer running OpenBSD, FreeBSD or NetBSD. You will need this access to try out the instructions and to learn. Peter is not spoonfeeding you – you will need to make an effort to learn, and learn by doing.

While you tinker with PF you also need access to the internet, not all the time – but when you want to check the state of PF in FreeBSD for example you will need to go to the FreeBSD PF web page. This information could of course have been included, but why? Including information that will soon be outdated is not the style for Peter, rather he has digested and decided to include references where appropriate and not include a lot of copy paste from other sources.

When Peter wrote this book he also makes it clear that he is not just teaching the available features, but the process of developing gateways with PF. His way of expanding simple *block in all* ruleset into a fully working examples with DMZ are fun to read and a beginner will learn not just the syntax of a firewall, but what makes a good firewall. If you need the syntax, which we all do, go to the materials from the extensive Appendix A with links to internet resources.

Having a book with the process is going to last longer than a book listing just the features in the current version. So this book will be worth it for years ahead, even though PF is in rapid development. He also presents his view of the world, and while I might not agree to everything – I consider greylisting evil – he does make some good arguments about which features to use and why. He does not just present a solution, he explains the why in the solution. When you get more experience with PF and firewalls you can always modify his solution to fit your needs.

Target audience

From my viewpoint this book is for everyone who uses PF. Regardless of operating system and skill level this book will teach you something new and interesting. The instructions are precise enough to get the beginner started, while the seasoned PF user will be compelled to update rulesets to include the best current practice for improved readability and performance. I have used PF since it was included in OpenBSD and yet I have something to try out immediately.

Conclusion

This book is a great version of the *Firewalling with OpenBSD's PF packet filter* web page which is a joy to read from cover to cover. The content is presented in a compressed format that will make the interested reader eager to try PF in practice. Combined with the official PF User's guide it will make you proficient in PF.

I can recommend buying this book and at the same time download his online web page.

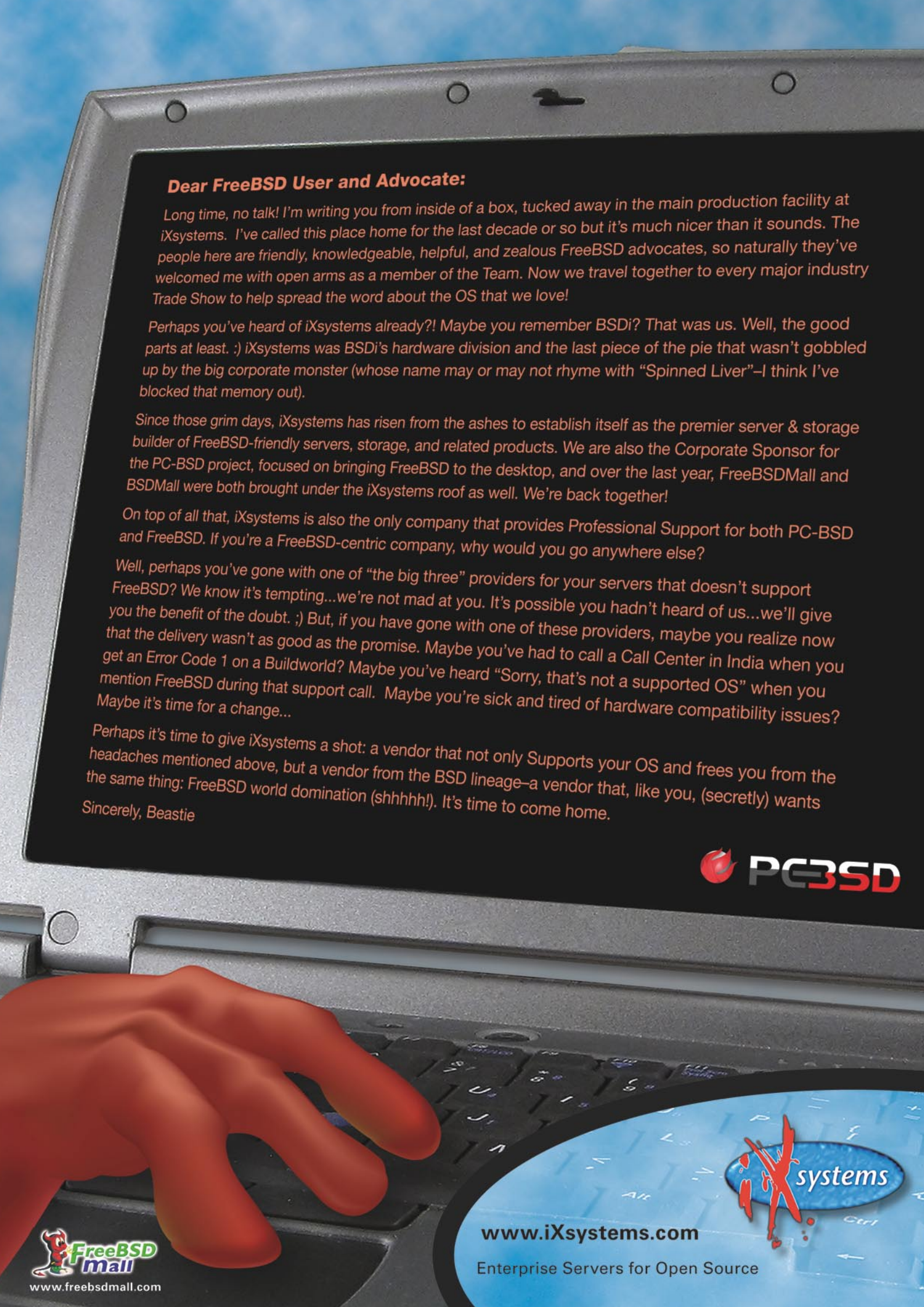
A big thank you goes to Peter, the OpenBSD project and especially Daniel Hartmeier for giving us PF.

by *Henrik Lund Kramshøj, hlk@kramse.dk*

In the next issue:

- The BSD certification
- Building an OpenBSD SAMP server with content filtering proxy
- Virtualisation in PC-BSD

Next issue of BSD Magazine available in September !


Dear FreeBSD User and Advocate:
Long time, no talk! I'm writing you from inside of a box, tucked away in the main production facility at iXsystems. I've called this place home for the last decade or so but it's much nicer than it sounds. The people here are friendly, knowledgeable, helpful, and zealous FreeBSD advocates, so naturally they've welcomed me with open arms as a member of the Team. Now we travel together to every major industry Trade Show to help spread the word about the OS that we love!
Perhaps you've heard of iXsystems already?! Maybe you remember BSDi? That was us. Well, the good parts at least. :) iXsystems was BSDi's hardware division and the last piece of the pie that wasn't gobbled up by the big corporate monster (whose name may or may not rhyme with "Spinned Liver"–I think I've blocked that memory out).
Since those grim days, iXsystems has risen from the ashes to establish itself as the premier server & storage builder of FreeBSD-friendly servers, storage, and related products. We are also the Corporate Sponsor for the PC-BSD project, focused on bringing FreeBSD to the desktop, and over the last year, FreeBSDMall and BSDMall were both brought under the iXsystems roof as well. We're back together!
On top of all that, iXsystems is also the only company that provides Professional Support for both PC-BSD and FreeBSD. If you're a FreeBSD-centric company, why would you go anywhere else?
Well, perhaps you've gone with one of "the big three" providers for your servers that doesn't support FreeBSD? We know it's tempting...we're not mad at you. It's possible you hadn't heard of us...we'll give you the benefit of the doubt. ;) But, if you have gone with one of these providers, maybe you realize now that the delivery wasn't as good as the promise. Maybe you've had to call a Call Center in India when you get an Error Code 1 on a Buildworld? Maybe you've heard "Sorry, that's not a supported OS" when you mention FreeBSD during that support call. Maybe you're sick and tired of hardware compatibility issues? Maybe it's time for a change...
Perhaps it's time to give iXsystems a shot: a vendor that not only Supports your OS and frees you from the headaches mentioned above, but a vendor from the BSD lineage–a vendor that, like you, (secretly) wants the same thing: FreeBSD world domination (shhhhh!). It's time to come home.
Sincerely, Beastie
PCBSD
FreeBSD mall
www.freebsdmall.com
ix systems
www.iXsystems.com
Enterprise Servers for Open Source