FREE DVD INSIDE OPENBSD 4.5

MAGAZINE

BSD

HOW TO GET STARTED

BSD SECURITY

OPENBSD 4.5

EXCLUSIVELY

- Interview with Matt Juszczak
- All about BSD Certification

INSIDE

POSTGRESQL, SHARED MEMORY AND BSD
BUILDASEARCH A FREEBSD WEB SERVICE
WEB SERVERS FOR EMBEDDED NETBSD
STAYING SECURE USING PC-BSD
STOP HACKERS WITH PROTECTION SCRIPT
SECURING OPENSSH SERVER
SSHFS ON NETBSD 5.0

Vol.2 No.4
Pride USD14.99
Issue 4/2009(6)
1898-9144

iXsystems™

800-820-BSDI

http://www.iXsystems.com

PCBSD™

http://www.pcbsd.org

FreeBSD.

FreeBSD is a registered trademark of The FreeBSD Foundation.

iXsystems Introduces the iX-Apollo Workstation

The iX-Apollo Extreme Series is iXsystems' first fully-qualified PC-BSD workstation. The stylish iX-Apollo Extreme series showcases silence and energy efficiency. The design of the workstation's chassis provides an ultra quiet environment. A whisper quiet 80PLUS® certified power supply offers increased energy efficiency, leaving a smaller impact on the environment as well as your power bill. The iX-Apollo Extreme Series uses the NVIDIA® GeForce® 9800 GT video card to deliver high-end graphics and its HybridPower™ technology saves energy during times of less intense video operations. Utilizing the new Intel® Core® i7 processors, the iX-Apollo Extreme supports up to eight logical cores of raw CPU processing power and up to 16GB of triple channel DDR3 memory.

The iX-Apollo Extreme Series includes the latest version of PC-BSD, Galileo, pre-installed and pre-configured. PC-BSD provides an easy-to-use desktop experience, featuring the KDE 4.2.3 desktop environment. PC-BSD exclusively features Push-Button Installers (PBIs), graphical utilities which easily install and remove software. With its inherent resistance to viruses, PC-BSD allows you to worry less about getting rid of viruses and spyware, and leap quickly and easily into the open source world. PC-BSD provides stability and security that you can count on.

Getting PC-BSD up and running is fast and easy, but having expert help on-hand to solve your problems can take your computing experience to new heights.By purchasing iXsystems Professional Services, you gain a team of PC-BSD and FreeBSD experts. These experts are well-versed in compiling software applications into PBIs, allowing you to install applications previously unavailable on PC-BSD. From optimizing your small office set-up to providing guidance on large-scale deployments, the iXsystems Professional Services Team will ensure that you get the most from your PC-BSD and FreeBSD systems.

*Monitor Sold Separately

iX-Apollo

- 16GB of DDR3 memory
- GigE LAN
- 3D capable nVidia grafix
- Whisper quiet chassis design
- High efficiency power supply

Editor's Note

Dear Readers,

Most people involved in Unix claim that BSD systems are one of the most secure OS's for our computers. Bearing that in mind, we decided to prove it by devoting this issue to BSD security. Another focus for this issue is OpenBSD, which in fact, is said to be the most secure in the BSD family. Our authors prepared a quick how-to for those who are just beginning and a few articles for people who already know what it is all about.
So, if you are having problems with protecting your computer and you spent the last few days thinking about how to make it secure, this issue is definitely a good start for you.

We did not forget about all you BSD users who do not favour OpenBSD much. You will find articles concerning NetBSD webservers, PC-BSD security and Postgresql on FreeBSD.

So stop worrying whether your system is secure or not – make it happen!

Karolina Lesińska
Editor in Chief

MAGAZINE BSD

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

Contributing: Ivan Rambius Ivanov, Barry Fox, Jan Stedehouder, Diego Montalvo, Donald T. Hayford, Antti Kantee, Christian Brueffer, Marko Milenovic, James T. Nixon III, Svetoslav P. Chukov, Michael Hernandez, Mikel King, Josh Paetzel
Special thanks to Michael Cooter

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

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

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

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

Postal addres:
Software Media LLC
1521 Concord Pike, Suite 301
Brandywine Executive Center
Wilmington, DE 19803
USA
tel: 1 917 338 36 31
www.bsdmag.org

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

Print: ArtDruk www.artdruk.com

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

Cover image on the iStock licence

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

Phone + 31 (0) 36 5307118
Fax + 31 (0) 36 5407252

get started

how-to's

security corner

mms

column

let's talk

tips&tricks

Installing OpenBSD

Ivan “Rambius“ Ivanov

OpenBSD 4.5 is the latest version of OpenBSD released in May, 2009. This article will walk you through its installation in great details. For a quick start boot the attached DVD with OpenBSD 4.5.

Before you start the installation familiarize yourself with the OpenBSD FAQ (*http://openbsd.org/faq/index.html*) and especially with its Section 4 that describes the installation process in great detail. Have the FAQ available during the installation process. Gather information about your hardware – hard disks, RAM, network cards, information about your network and video card and monitor specifications if you are going to use graphical environment.

Decide how much disk space you will dedicate to OpenBSD hard disk partitions. If you install an OpenBSD server you will perhaps allocate a whole disk to it. If you install it on a desktop possibly with other OSes you have to decide what part of the disk will be given to OpenBSD.

Once you make the decision about the partitions they have to be divided in slices. Partition slices are a BSD concept and have no Linux or Windows equivalent. Each BSD partition can be split into slices and each slice (except the swap one) has a mount point.

The OpenBSD system is broken into installation sets – a base system set, a manual pages set, a compilers set, X Window related sets. You have to consider what media will you use to bootstrap the installation and what media to obtain the sets.

You can launch the installation by booting the DVD provided with the magazine. You can then download and install the OpenBSD sets from an FTP or an HTTP mirror. Or better yet you can purchase the official 3-CDs installation kit which can boot for all supported platforms, contains the installations sets, precompiled binary application packages for some platforms, the OpenBSD sources, the ports tree and some funny artwork. Buying the kit is an excellent way to support the OpenBSD project and it will save you some network bandwidth and download time during the installation. In this article we will install from the official CDs.

And once again, please read the OpenBSD 4.5 Installation Guide (*http://openbsd.org/faq/faq4.html*) from the FAQ. Understanding it will save you time and will prevent you from making errors and redoing the installation.

Performing the installation

After you finish the preparations boot from the installation CD. The installer loads the kernel, probes the available devices and gives you the following prompt:

```
(I)stall, (U)pgrade or (S)hell
```

Choose `(I)nstall` by pressing `i`. You will be shown a welcome message and some instructions and will be reminded to backup your data. If you have done so, confirm the installation with the next prompt:

```
Proceed with install? [no] yes
```

Next, the available hard disks are shown to choose on which OpenBSD will be installed. In our example the machine has two hard drives and we will install on the first one. Since there is another OS on it, we also choose the default option to not use the whole disk for OpenBSD.

```
Available disks are: wd0 wd1
Which one is the root disk? (or 'done') [done] wd0
Do you want to use *all* of wd0 for OpenBSD? [no]
```

The installer then starts fdisk, it prints the existing partition table and shows its prompt:

```
Enter 'help' for information
fdisk: 1>
```

To get a list of the commands type help/h; to read the full fdisk's man page type manual or m. The command print/p optionally followed by k, m or g prints the partition table in kilobytes, megabytes, gigabytes or by default in sectors: see Listing 1.

We add a new partition with the command edit/e followed by the number of the partition:

```
fdisk: 1> e 2
Partition id ('0' to disable) [0
- FF]: [0] A6
Do you wish to edit in CHS mode? [n]
offset: [0] 325186848
size: [0] 325058328

fdisk: *1>
```

The partition ID is a number specifying the file system that will be used on this partition – in the case of OpenBSD it is A6. You can find the list with the supported types and their IDs by pressing ? at this prompt. Next, CHS means Cylinders-Heads-Sector and in this mode we provide the boundaries of the partition with its starting and ending cylinders, heads and sectors; we do not want to use it here.

We specify the beginning of the partition as an offset in sectors from the beginning of disk. In this example we want it immediately after partition #1. Finally we give the size of the partition in sectors as well. The OpenBSD partition should not overlap with any other partition. After you do your changes print the partition table and compare the boundaries of the partitions to confirm there is not overlapping. If you have made an error in the sectors arithmetics, edit the partition with the same e 2 command.

The prompt now has a * which means there are unsaved changes. We save them with `write/w` command and quit fdisk: see Listing 2.

After we create the OpenBSD partition with fdisk, we have to divide it in slices with the disklabel utility: see Listing 3.

Each slice is assigned a letter. The a slice is the root filesystem `/`. The c slice represents the whole disk and cannot be altered. The b slice is reserved for swap – if you do not want swap simply do not create a slice under the letter b. The slices' sizes and layout depend greatly on your needs. For example if you use database systems which by default create their files in `/var` you may dedicate a slice to `/var`. As a minimum it is good to create a slice for `/home` as it will help you to upgrade the OS without disturbing the users' home directories. We will now create a number of slices just for illustartion. The disklabel command to add a slice is a and the command to print the slices table is p. If you do not like the existing layout you can reset it with the command D. The changes in the slices are written with w command: see Listing 4.

After you quit disklabel you verify the mount points. When you confirm them it creates the filesystems. The biggest part of the installation is competed.

Next comes the network configuration. Since we have the installation sets on a CD at this point we do not really need network connection and we skip it.

```
System hostname: (short form, e.g.
foo) denica
Configure the network? [yes] no
```

If you are going download the installation sets from Internet it is crucial you set up your network: see Listing 5.

Next you have to type the root's password and to retype it:

```
Password for root account? (will not
echo)
Password for root account? (again)
```

Listing 1. Printing the Partition Table

```
fdisk: 1> p g
Disk: wd0 geometry: 60801/255/63 [466 Gigabytes]
Offset: 0 Signature: 0xAA55
Starting Ending LBA Info:
#: id C H S - C H S [ start: size ]
-------------------------------------------------------------------
0: 06 0 1 1 - 7 254 63 [ 63: 0G] DOS > 32MB
1: A5 8 0 1 - 20241 240 63 [ 128520: 155G] FreeBSD
2: 00 0 0 0 - 0 0 0 [ 0: 0G] unused
```

Listing 2. Writing the Partition Table

```
*1> w
Writing MBR at offset 0.
fdisk: 1> p g
Disk: wd0 geometry: 60801/255/63 [466 Gigabytes]
Offset: 0 Signature: 0xAA55
Starting Ending LBA Info:
#: id C H S - C H S [ start: size ]
-------------------------------------------------------------------
0: 06 0 1 1 - 7 254 63 [ 63: 0G] DOS > 32MB
1: A5 8 0 1 - 20241 240 63 [ 128520: 155G] FreeBSD
2: A6 20241 241 1 - 40475 226 63 [ 325186848: 155G] OpenBSD
3: 00 0 0 0 - 0 0 0 [ 0: 0G] unused
fdisk: 1> quit
```

Listing 3. Disklabel Slices Table

```
Initial label editor (enter '?' for help at any prompt)
> p
OpenBSD area: 325186848-650245176; size: 325058328; free: 325058328
# size offset fstype [fsize bsize cpg]
c: 976773168 0 unused
i: 128457 63 MSDOS
j: 325058328 120520 unknown
```

It should be hard to guess the root's password and sometimes I use password generators to generate random passwords.

Now we provide the location of the OpenBSD installation sets:

```
Location of the sets? (cd disk ftp
http or 'done') [cd]
Available CD-ROMs are: cd0
Which one contains the install media?
(or 'done') [cd0]
```

Listing 4. Creating Slices

```
> a a
offset: [325186848]
size: [325058328] 5g
Rounding to cylinder: 10475262
FS Type: [4.2BSD]
mount point: [none] /
> a b
offset: [335662110]
size: [314583066] 4g
Rounding to cylinder: 8401995
FS Type: swap
> a d
offset: [344064105]
size: [306181071] 50g
Rounding to cylinder: 104872320
FS Type: [4.2BSD]
mount point: [none] /usr
> a e
offset: [448936425]
size: [201308751] 25g
Rounding to cylinder: 52436160
FS Type: [4.2BSD]
mount point: [none] /var
> a f
offset: [501372585]
size: [148872591] 10g
Rounding to cylinder: 20980890
FS Type: [4.2BSD]
mount point: [none] /var/log
> a g
offset: [522353475]
size: [127891701] 20g
Rounding to cylinder: 41945715
FS Type: [4.2BSD]
mount point: [none] /tmp
> a h
offset: [564299190]
size: [85945986]
Rounding to cylinder:
FS Type: [4.2BSD]
mount point: [none] /home
> w
> q
```

```
Pathname to the sets? (or 'done')
[4.5/i386]
```

If you configured the network you can obtain the sets from an OpenBSD mirror. Choose one from a numbered list of servers: see Listing 6.

No matter how you locate the sets, you have to choose which sets exactly you want: see Listing 7.

The required sets are bsd – the kernel, base45.tgz – the base system and `etc45.tgz` – the files in /etc. The optional, but recommended ones are `comp45.tgz` – contains the compiler, headers and libraries and `man45.tgz` – contains the manual pages. The sets that start with `x` are related to the X Window system. We choose to install all sets. If you are preparing a headless server with no graphical environment you can exclude the X Window sets. After we choose the sets it proceeds with installing them.

The final step of the installation is to enable or disable sshd and ntpd and choose the time zone: see Listing 8.

Here we choose to run sshd and to not run ntpd. You may choose otherwise if you wish.

This pretty much concludes the installation. Now we halt the machine and reboot the newly installed OpenBSD system.

Adding a new user after the first boot

After the new system boots we login into it as root. Up to now root is the only user we have and its powers are to much to be used for day-to-day work. We use the command useradd to add a new user with its `-m` option to create the user's home directory and then the command passwd to give it a password.

```
# useradd -m rambius
# ls /home
rambius
# passwd rambius
Changing local password for rambius.
New password:
Retype new password:
```

It is a good practice to limit the use of root, but still you have to be able to run administrative commands. We can give the new user the ability to become root or to execute commands as root.

The first way is to add it to wheel group which will enable it to use su and switch to root:

```
# user mod -G wheel rambius
# groups rambius
users wheel
```

Now exit the root session and log in as the regular user: see Listing 9.

By adding a user to wheel group you enable him to potentially execute whatever commands the user wants as root. The other way is to allow him/her to execute only certain commands as root using the utility sudo. Back as root invoke

```
# visudo
```

and the following line in it

```
rambius ALL=(ALL) SETENV: ALL
```

In this case we allow again the regular user to execute all the commands as root, but we have much control to restrict the commands. Check sudo(8) for more information. Now login as the regular users and execute:

```
$ sudo id
Password: <type user password>
```

A difference between the two methods is that for su you have to provide root's password; for sudo you have to provide the password of the user calling sudo and it prompts for that password almost every time you use it.

It is time to configure the network if we have not done so during the installation. Each network interface is configured in a file called `/etc/hostname.<interface>`. To find out your interfaces use ifconfig: see Listing 10.

We will configure em0 Ethernet interface. If you use dhcp it takes two commands:

```
# echo dhcp > /etc/hostname.em0
# sh /etc/netstart
```

If you want to statically assign an IP address, it will take you four commands:

```
# echo "inet 192.168.1.15
255.255.255.0" > /etc/hostname.em0
# echo "192.168.1.1" > /etc/mygate
```

```
# vi /etc/resolv.conf
search mydomain.com
nameserver 192.168.1.1
nameserver 192.168.1.2
lookup file bind
```

Now will install some application to our new system. They are shipped as OpenBSD packages, which consist of the precompiled applications plus some packing information like their dependencies. Again you can install them from various sources – CDs, FTP servers, etc. You specify the location of the packages in the PKG PATH environment variable and then you install them with `pkg` add. Here is an example how to install from a CD:

```
$ sudo mount /dev/cd0a /cdrom
$ export PKG_PATH=/cdrom/`uname -r`/
packages/`machine -a`
$ sudo pkg_add kdebase
```

This will install the package kdebase and all its dependencies. You can use the same command but different `PKG PATH` to install a package from an OpenBSD FTP mirror: see Listing 11.

Here several packages named emacs are found and we have to specify which one exactly we want.

It is possible to specify several package locations in `PKG PATH` separated by colons and they are searched by the order in which they appear. The package is installed from the first place in which it is found.

```
# PKG_PATH=/cdrom/`uname -r`/packages/
`machine -a`:
> ftp://ftp.cse.buffalo.edu/pub/
OpenBSD/`uname -r`/packages/`machine
-a`
$ sudo pkg_add kde-i18n-bg
```

Starting the graphical environment

In the package example below we added KDE environment and now we will show how to use it. First, start X Window System as regular user to verify it works normally:

```
$ startx
```

You should get the default window manager with an xclock and an xterm on the screen. Press [*Ctrl-Alt-Backspace*] to kill *X*. Create a new `.xinitrc` and start KDE in it and invoke startx:

```
$ echo exec startkde > ~/.xinitrc
$ startx
```

You can also get a graphical login. Edit `rc.conf.local`.

```
# echo "xdm_flags=" >> /etc/
rc.conf.local
```

Listing 5. Network Setup with DHCP

```
Configure the network? [yes]
Available interfaces are: em0
Which one do you wish to initialize? or 'done' [em0]
Symbolic (host) name for em0? [denica]
The media options for em0 are currently
media: Ethernet autoselect (100baseTX full-duplex)
Do you want to change the media options? [no]
IPv4 address for em0? (or 'none' or 'dhcp') [dhcp]
Issuing hostname-associated DHCP request for em0.
IPv6 address for em0? (or 'rtsol' or 'none') [none]
DNS domain name? (e.g. 'bar.com') [mydomain.com]
DNS nameserver (IP address ot 'none') [1.2.3.4]
Use the nameserver now? [yes]
Default IPv4 route? (IPv4 address, 'dhcp' or 'none') [dhcp]
Edit hosts with ed? [no]
Do you want to do any manual network configuration? [no]
```

Listing 6. Locating Sets on FTP mirror

```
Location of the sets? (cd disk ftp http or 'done') [cd] ftp
HTTP/FTP proxy URL? (e.g. 'http://proxy:8080', or 'none') [none]
Display the list of the known ftp server? [no] yes
...

List skipped for brevity
...

Server? (IP address, hostname, list#, 'done' or '?') 78
Use active mode ftp [no]
Server directory? [pub/OpenBSD/4.5/i386]
Login? [anonymous]
```

Listing 7. Available Sets

```
[X] bsd
[X] bsd.rd
[ ] bsd.mp
[X] base45.tgz
[X] etc45.tgz
[X] misc45.tgz
[X] comp45.tgz
[X] man45.tgz
[X] game45.tgz
[ ] xbase45.tgz
[ ] xetc45.tgz
[ ] xshare45.tgz
[ ] xfont45.tgz
[ ] xserv45.tgz
Set name? (or 'done') [bsd.mp] all
```

Specify which window manager you want to use with the graphical login in `.xsession` in your home directory:

```
$ echo startkde > ~/.xsession
```

After you reboot the machine you will see the graphical login.

Next steps and additional resources

Now when you have a working OpenBSD system, you can dive in it and explore it. `afterboot(8)` man page is the place to start reading about it. If you want more information about the command used in this article you can check:

- `fdisk(8)`, `disklabel(8)`, `mount(8)`, `fstab(5)` to see how to create partitions and slices and how to mount them;
- ifconfig, `hostname(1)`, `hostname.if(5)`, `myname(5)`, `mygate(5)`, `dhcp(8)` to understand how configure your network with dynamic or static address;
- `sshd(8)` and `ssh(8)` to remotely login to your machine using Secure Shell;
- `user(8)` to add, modify and remove users in your system;
- `su(1)` and `sudo(8)` to allow users to execute commands as root;
- `pkg add(1)` and pkg delete to add, update and remove packages;
- There are many graphical windows managers besides KDE – Gnome, Xfce, icewm, etc. Give some of them a try – you may find them easier and faster to use than KDE
- Rebuild OpenBSD from sources. It is a great way to understand more about the system. You can obtain them from the official CDs or download them.

Listing 8. sshd and ntpd setup

```
Start sshd(8) by default? [yes]
Start ntpd(8) by default? [no]
Do you expect to run the X Window System? [no] yes
What timezone are you in? ('?' for list) [Canada/Mountain] Europe/Sofia
```

Listing 9. Testing su

```
OpenBSD/i386 (denica) (ttyC0)
login: rambius
Password:
$ id
uid=1000(rambius) gid=1000(rambius) groups=1000(rambius), 0(wheel)
$ su -
Password: <type root password>
# id
uid=0(root) gid=0(wheel) groups=0(wheel), 2(kmem), 3(sys), 4(tty),
5(operator), 20(staff), 31(guest)
```

Listing 10. Showing network interfaces

```
$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33204
groups: lo
inet 127.0.0.1 netmask 0xff000000
inet6 ::1 prefixlen 128
inet6 fe80::1%lo0 prefixlen 64 scopeid 0x4
wpi0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
lladdr 00:13:02:5b:66:b7
groups: wlan
media: IEEE802.11 autoselect
status: no network
ieee80211: nwid "" 100dBm
bge0: flags=8802<BROADCAST,SIMPLEX,MULTICAST> mtu 1500
lladdr 00:15:c5:0a:28:69
media: Ethernet autoselect (100baseTX full-duplex)
status: active
enc0: flags=0<> mtu 1536
```

Listing 11. PKG PATH

```
$ PKG_PATH=ftp://ftp.cse.buffalo.edu/pub/OpenBSD/`uname -r`/packages/
`machine -a`
$ export PKG_PATH
$ sudo pkg_add emacs
Ambiguous: emacs could be emacs-21.4p7 emacs-21.4p7-no_x11
emacs-22.2p0 emacs-22.2p0-gtk emacs-22.2p0-no_x11
$ sudo pkg_add emacs-22.2p0-no_x11
```

Upgrading to OpenBSD 4.5

When we bootstrapped the installer from the CD we received the following prompt:

```
(I)stall, (U)pgrade or (S)hell
```

If you have an older version of OpenBSD you may upgrade it. The upgrade is similar to a clean installation. You have to choose which partition holds the existing OpenBSD system, which slice has the root filesystem, what other slices you want to mount for the installation and which sets you will fetch and install. After you upgrade you can also upgrade the existing packages with pkg_add -u command.

About the Author

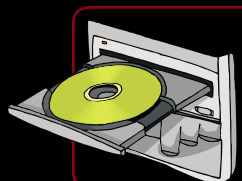
Ivan Ivanov is a Bulgarian software developer working in New York for Ariel Partners LLC. His main area of expertise is software project automation. He is also a member of the New York City BSD User group.

What's new in OpenBSD 4.5?

This is a partial list of new features and systems included in OpenBSD 4.5. For a comprehensive list, see the changelog leading to 4.5.

- New/extended platforms:
  - Initial ports to the xscale based gumstix platform and the ARM based OpenMoko
- OpenBSD/sparc64
  - New vdsk(4) and vnet(4) drivers provide support for virtual I/O between logical domains on Sun's CoolThreads servers, including UltraSPARC T2+ machines.
- Workstations and laptops with UltraSPARC IIe CPUs can now scale down the CPU frequency to save power.
- Improved hardware support, including:
  - Several new/improved drivers for sensors, including: The cac(4) driver now has bio and sensor support.
- The mpi(4) driver now has bio and sensor support.
- New gpiodcf(4) driver for DCF77/HBG timedelta sensors through GPIO pins.
- New schsio(4) driver for SMSC SCH311x LPC Super I/O devices.
- The it(4) driver now supports IT8720F chips.
- The it(4) driver now supports FAN4 and FAN5 sensors for `IT8716F/IT8718F/IT8720F/IT8726F` chips.
- The owtemp(4) driver now supports Maxim/Dallas DS18B20 and DS1822 temperature sensors.
- The km(4) driver now supports AMD Family 11h processors (Turion X2 Ultra et al).
- The lm(4) driver now supports W83627DHG attachment on the I^2C bus.
- The lmenv(4) driver now has better support for the fan sensors on lm81, adm9240 and ds1780 chips.
- The sdtemp(4) driver now supports ST STTS424 chips.
  - The em(4) driver now supports ICH9 IGP M and IGP M AMT chips, and link status detection has improved.
- The sdmmc(4) driver now supports SDHC cards.
- The msk(4) driver now supports Yukon-2 FE+ (88E8040, 88E8042) based devices.
- The iwn(4) driver now supports Intel WiFi Link 5100/5300 devices.
- The wpi(4) and iwn(4) drivers now support hardware CCMP cryptography.
- The ath(4) driver now has WPA-PSK support.
- age(4), a driver for Attansic L1 gigabit Ethernet devices was added.
- ale(4), a driver for Atheros AR81xx (aka Attansic L1E) Ethernet devices was added.
- mos(4), a driver for Moschip MCS7730/7830 10/100 USB Ethernet devices was added.
- jme(4), a driver for JMicron JMC250/JMC260 10/100 and Gigabit Ethernet devices was added.
- run(4), a driver for Ralink USB IEEE 802.11a/b/g/Draft-N devices was added.
- auacer(4), a driver for Acer Labs M5455 audio devices was added.
- ifb(4), a driver for Sun Expert3D, Expert3D-Lite, XVR-500, XVR-600 and XVR-1200 framebuffers (accelerated).
- wildcatfb(4), an X driver for Sun Expert3D, Expert3D-Lite, XVR-500, XVR-600 and XVR-1200 framebuffers (unaccelerated).
- sunffb(4), an accelerated X driver for Sun Creator, Creator 3D and Elite 3D framebuffers.
- vdsk(4), a driver for virtual disks of sun4v logical domains.
- vnet(4), a driver for virtual network adapters of sun4v logical domains.
- vrng(4), a driver for the random number generator on Sun UltraSPARC T2/T2+ CPUs.
- The vcons(4) driver is now interrupt driven.
- ips(4), a driver for IBM SATA/SCSI ServeRAID controllers was added.
- udfu(4), a driver for device firmware upgrade (DFU) was added.
- Many improvements were made to the acpi(4) subsystem.
- The umsm(4) driver supports several new EVDO/UMTS devices.
- The mfi(4) driver now supports the next generation of MegaRAID SAS controllers.
- New vsbic(4) driver for the MVME327A SCSI and floppy controller on mvme68k and mvme88k machines.
- The re(4) driver now supports 8168D/8111D-based devices, and multicast reception on 8110SB/SC-based devices.
- The ehci(4) driver now supports isochronous transfers.
- S/PDIF output support has been added to the ac97(4), auich(4), auvia(4) and azalia(4) drivers.
- azalia(4) mixer has been clarified and simplified, support for 20-bit and 24-bit encodings has been added.
- The gbe(4) frame buffer driver now supports acceleration.
- New tools:
  - ypldap(8), an YP server using LDAP as a backend.
  - xcompmgr(1) was added to xenocara.
- New functionality:
  - The libc resolver(3) may now be forced to perform lookups by TCP only using a new resolv.conf(5) option. The nameserver declaration in resolv.conf(5) has also been extended to allow specification of non-default nameserver ports.
- apropos(1) has two new options (-S and -s) to allow searching by machine architecture and manual section.
- aucat(1) now has audio server capability. Audio devices can be shared between multiple applications. Applications can run natively on fixed sample rate devices or on devices with unusual encodings. Multi-channel audio devices can be split into smaller independent subdevices.
- aucat(1) now has a deviceless mode, in which it can be used as a general purpose audio file format conversion utility (to mix, demultiplex, resample or reencode files).
- ifconfig(8) can now list channels supported by an IEEE 802.11 device.

For more details visit *www.openbsd.org*



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

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

Postgresql, shared memory and BSD

Barry Fox

This series of articles will compare a basic installation of Postgresql on various flavors of BSD, and compare the performance between them given similar shared memory settings.

This first article will discuss the configuration on FreeBSD, including a quick overview of the install process. We will also go through some shared memory settings in the postgresql.conf file (and in the operating system) that will be modified in order to compare the relative performance (on the same machine, and also across operating systems). The goal is to get a sense of how the shared memory settings affect performance over a few simple queries, and how the various BSD operating systems differ.It is not meant to provide any *real world* configuration advice, as the settings are somewhat contrived in order to provide a basis for experimentation.

Initial installation

Installing Postgresql is a straightforward task.It will compile easily from source (it requires gnu make, so that must be installed first, and follows the standard `configure/make/make` install path), and by default things are installed in `/usr/local/pgsql`.

Before the database engine can be started, it must be initialized. The initialization process basically sets up the initial configuration stuff, and preps everything. As the command runs, you will see things scroll over the console: in Listing 1.

The path at specified with the `-D` switch is the database root directory. All the configuration files, and data will reside within this directory. As this is a test machine, I am simply placing it within the home directory of the postgres user.However on a production box, a more thoughtful location might be preferred.

The initdb command will automatically set the various shared memory values depending on what it thinks is appropriate based on the system resources. These settings may be sufficient for many small databases, however a large frequently accessed database will require additional configuration.

At this point the engine can be started (Listing 2). The engine is now running with the default configuration. Usually the next step in the configuration will be to edit the `pg_hba.conf` file to set up the connection settings (what hosts are allowed to connect to the engine, etc.), as well as to edit the `postgresql.conf` file. We will be looking into the `postgresql.conf` file in a little bit, but first we will go through creating a database, loading in a bit of data, and then running some timing tests to see how the default configuration works.

Setting up the initial tests

The createdb command creates a new database (it is a well named command). There are options that would allow the database to go into a different tablespace (which is important when administering a live system, as it allows for more flexible volume management). It is also possible to limit the number of concurrent connections allowed on the database (which can be useful to reduce resource contention). For our purposes, we will go with the defaults.

```
$ /usr/local/pgsql/bin/createdb test
$
```

psql is the `shell` for postgresql, and it allows us to run SQL, and view the results. To load in some sample data, I will first create a new table, and then copy the data from a file: see Listing 3.

The HINT/LOG messages are important, as they point to some optimizations that can be done in with the postgres logging. It means that the checkpoints are happening closely together, which is not optimal for performance.

Changing the checkpoint configuration parameters would help reduce the number of checkpoints (by making each checkpoint handle more data). For our purposes we are not too concerned about this, although on a live system we would be.

The table is now populated with some data: see Listing 4.

We will be using *explain analyze* in order to see what the system is doing to while running the query, however it does not provides the best method to see the total run time (because it can greatly slow things down, to the point of taking 10+ hours to anal yze a query that normally takes 10-15 minutes). This SQL query counts all the rows in the table, it does a sequential scan over the entire table, and takes just about 20 seconds to run.

After loading in a second table with some data, some more complicated queries can be run. Joins are the real performance killer in most database applications. This is because they work by taking a cross product from the two tables, resulting in a massive number of rows to be examined(and keeping track of all these rows takes a lot of resources). For example: see Listing 5.

This run takes 24000 seconds! However if the query is run like this: see Listing 6.

We get a run time of 14 minutes, and 8 seconds (the two *select current_time* are added just to give us an idea how long the query in question took to actually run). Obviously the two are slightly different, however both need to touch the same number of pages. The added execution time in the explain version is a result of the added overhead required for the profiling of the query.

Of note in the explain, we see that a sequential scan is run over both tables. If we try to do another run using database indices we obtain: see Listing 7.

In this case, the database indices are not used, even though they exist. To determine why this is, we need to look into the shared memory on the system, and analyze how the memory and disk is being used during the query execution. Lets look a little more into what shared memory does, and what increasing it might do to help this queries performance.

Shared Memory

Generally, on any computer, shared memory is a pool of memory that different processes can all access in order to share information amongst themselves. As memory is a very limited resource, the operating system will place limits on how much shared memory can be set aside for a process (or a pool of processes). This generally helps keep the system running as expected.

Table 1. Some Postgresql shared memory parameter

Name	Use
shared_buffers	The amount of memorythat is used to cache data
max_fsm_pages	The maximum number of free pages that will be tracked.

Listing 1. Postgresql initialization

```
$ /usr/local/pgsql/bin/initdb  -D /home/postgres/DB
The files belonging to this database system will be owned by user
"postgres".
This user must also own the server process.

The database cluster will be initialized with locale C.
The default database encoding has accordingly been set to SQL_ASCII.
The default text search configuration will be set to "english".

creating directory /home/postgres/DB ... ok
creating subdirectories ... ok
selecting default max_connections ... 40
selecting default shared_buffers/max_fsm_pages ... 28MB/179200
creating configuration files ... ok
creating template1 database in /home/postgres/DB/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    /usr/local/pgsql/bin/postgres -D /home/postgres/DB
or
    /usr/local/pgsql/bin/pg_ctl -D /home/postgres/DB -l logfile start
```

Listing 2. Starting Postgresql

```
$ /usr/local/pgsql/bin/postgres -D /home/postgres/DB -i &
$ LOG:  database system was shut down at 2009-05-04 12:39:17 EDT
LOG:  database system is ready to accept connections
LOG:  autovacuum launcher started
```

Some applications though require a lot of shared memory. A DBMS is one of them. The speed killer for any DBMS is disk access. A large table will require many disk accesses in order to run a sequential scan over it (even if it is doing an index scan, we are still in the same boat, millions and billions of rows result in very large indices on the columns). Once all the pages of a table are read, they will be in memory. Now if another query is run that accesses the same table, it would be nice if the pages of the table that had just been read into memory could be reused by this new process. Shared memory allows this to happen. It allows a database system cache of disk pages to be accessed by any future query, thus minimizing the number of disk accesses. Of course, there is not infinite memory, so the cache can be filled up, so there is an algorithm that will remove pages from the cache, and they may need to be reloaded from disk (which in turn will slow down a query compared to if all the pages are in memory). This is the motivation for having a large pool of shared memory (and physical memory!) available on a system running a large database.

Listing 3. Creating a table, and loading in test data

```
$ /usr/local/pgsql/bin/psql test
Welcome to psql 8.3.7, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

test=# create table data1 (id int,c1 int,c2 int,c3 int,c4 int);
CREATE TABLE

test=# copy data1 from '/home/postgres/data1';
LOG:  checkpoints are occurring too frequently (8 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
LOG:  checkpoints are occurring too frequently (6 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
LOG:  checkpoints are occurring too frequently (5 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
LOG:  checkpoints are occurring too frequently (6 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
LOG:  checkpoints are occurring too frequently (6 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
LOG:  checkpoints are occurring too frequently (5 seconds apart)
HINT:  Consider increasing the configuration parameter "checkpoint_
segments".
COPY 4895928
test=#
```

Listing 4. Running a SQL query, and seeing what happens

```
test=# explain analyze select count(*) from data1;
                                                        QUERY PLAN

-------------------------------------------------------------------------------
-----------------------------------------------
 Aggregate  (cost=90000.00..90000.01 rows=1 width=0) (actual time=19814.63
3..19814.635 rows=1 loops=1)
   ->  Seq Scan on data1  (cost=0.00..77760.00 rows=4896000 width=0)
(actual time=3.548..9943.798 rows=4895928 loops=1)
 Total runtime: 19815.852 ms
(3 rows)

test=#
```

The `postgresql.conf` file defines all the engine settings; shared memory, port, logging etc. This is the main file that you need to know and love. For the purposes of this article, the shared memory parameters are most relevant, and only a couple will actually be changed. The full list of parameters can be found at: *http://www.postgresql.org/docs/8.3/static/kernel-resources.html* (see Table 1).

However, a key component are the operating system settings for shared memory. These are the key elements at the OS level (*http://www.postgresql.org/docs/8.3/static/kernel-resources.html*): see Table 2.

The default settings for FreeBSD are low, and only practical on a small database. The impact of changing these settings on FreeBSDwill provide us with a baseline to compare with NetBSD and OpenBSD in the next article.

The settings on the FreeBSD system while the above queries were run was: see Listing 8

```
$ sysctl kern.ipc.shmall
kern.ipc.shmall: 8192
$ sysctl kern.ipc.shmmax
kern.ipc.shmmax: 33554432
$ sysctl kern.ipc.semmap
kern.ipc.semmap: 30
$
```

They were changed as follows:

```
kern.ipc.shm_use_phys=1
kern.ipc.shmmax=1073741824
```

```
kern.ipc.shmall=262144
kern.ipc.semmsl=512
kern.ipc.semmap=256
```

The `shm_use_phys` will attempt to lock in the shared memory, to avoid it from being swapped to disk. This will ideally reduce the swapping of the database page cache, and improve performance.

When setting the system level settings, it is important to evaluate what else the system is doing.If is it primarily a database server, then allowing for the majority of the memory to be used by the database engine makes sense. However if it is running a variety of important tasks, in addition to the database engine,

Listing 5. A join

```
test=# explain analyze select a.id from data1 a,data2
b where a.id=b.id;

                QUERY PLAN

-----------------------------------------------------
-----------------------------------------------------
 Merge Join  (cost=1594911.78..81866227.29
rows=5349790523 width=4) (actual time=61963.732..18002
935.774 rows=2978820010 loops=1)
   Merge Cond: (a.id = b.id)
   ->  Sort  (cost=797500.75..809740.75 rows=4896000
width=4) (actual time=28694.640..38499.480
rows=4895928 loops=1)
         Sort Key: a.id
         Sort Method:  external merge  Disk: 76528kB
         ->  Seq Scan on data1 a  (cost=0.00..77760.00
rows=4896000 width=4) (actual time=10.984..15228.646
rows=4895928 loops=1)
   ->  Materialize  (cost=797391.08..858582.44
rows=4895309 width=4) (actual time=33269.086..6020020.
575 rows=2978819821 loops=1)
         ->  Sort  (cost=797391.08..809629.35
rows=4895309 width=4) (actual time=33269.082..43071.65
6 rows=4895928 loops=1)
               Sort Key: b.id
               Sort Method:  external merge  Disk:
76528kB
               ->  Seq Scan on data2 b
(cost=0.00..77753.09 rows=4895309 width=4) (actual
time=0.803..17650.072 rows=4895928 loops=1)
 Total runtime: 23971003.015 ms
(12 rows)

test=#
```

Listing 6. A join, without the explain

```
test=# select current_time;select count(a.id) from
data1 a,data2 b where a.id=b.id;select current_time;
      timetz
------------------
 21:00:25.4649-04
(1 row)

   count
------------
 2978820010
(1 row)

       timetz
--------------------
 21:14:33.841233-04
(1 row)
```

Listing 7. A join with database indices added to the join columns

```
test=# create index data1_id_index on data1(id);
CREATE INDEX
test=# create index data2_id_index on data2(id);
CREATE INDEX

test=# explain analyze select a.id from data1 a,data2
b where a.id=b.id;

        QUERY PLAN

-----------------------------------------------------
-----------------------------------------------------
 Merge Join  (cost=1595002.97..81875284.90
rows=5350388308 width=4) (actual time=52353.445..17993
383.348 rows=2978820010 loops=1)
   Merge Cond: (a.id = b.id)
   ->  Sort  (cost=797491.51..809731.33 rows=4895928
width=4) (actual time=24912.966..34717.616
rows=4895928 loops=1)
         Sort Key: a.id
         Sort Method:  external merge  Disk: 76528kB
         ->  Seq Scan on data1 a  (cost=0.00..77759.28
rows=4895928 width=4) (actual time=10.255..13085.619
rows=4895928 loops=1)
   ->  Materialize  (cost=797491.51..858690.61
rows=4895928 width=4) (actual time=27440.473..6014191.
429 rows=2978819821 loops=1)
         ->  Sort  (cost=797491.51..809731.33
rows=4895928 width=4) (actual time=27440.470..37244.00
7 rows=4895928 loops=1)
               Sort Key: b.id
               Sort Method:  external merge  Disk:
76528kB
               ->  Seq Scan on data2 b
(cost=0.00..77759.28 rows=4895928 width=4) (actual
time=3.964..13975.471 rows=4895928 loops=1)
 Total runtime: 23961505.737 ms
(12 rows)

test=#
```

then careful planning will be required to ensure that one set of tasks does not negatively affect any other.

For the next series of SQL queries, the Postgresql shared memory settings were changed to be:

```
shared_buffers = 756MB
temp_buffers = 128MB
work_mem = 128MB
max_fsm_pages = 17920000
```

Table 2. FreeBSD shared memory parameters

Name	Description
SHMMAX	Maximum size of shared memory segment (bytes)
SHMALL	Total amount of shared memory available (bytes or pages)

Listing 8. Postgresql requests too much shared memory

```
$ /usr/local/pgsql/bin/postgres -D /home/postgres/DB -i
FATAL:  could not create shared memory segment: Invalid argument
DETAIL:  Failed system call was shmget(key=5432001, size=1154555904,
03600).
HINT:  This error usually means that PostgreSQL's request for a shared
memory segment exceeded your kernel's SHMMAX parameter.  You can either
reduce the request size or reconfigure the kernel with larger SHMMAX.  To
reduce the request size (currently 1154555904 bytes), reduce PostgreSQL's
shared_buffers parameter (currently 96768) and/or its max_connections
parameter (currently 43).
        If the request size is already small, it's possible that it
is less than your kernel's SHMMIN parameter, in which case raising the
request size or reconfiguring SHMMIN is called for.
        The PostgreSQL documentation contains more information about
shared memory configuration.
$
```

Listing 9. A join with more hared memory

```
test=# explain analyze select a.id from data1 a,data2 b where a.id=b.id;
                                                                 QUERY
PLAN
------------------------------------------------------------------------
------------------------------------------------------------
 Merge Join  (cost=1360712.97..81640994.90 rows=5350388308 width=4)
(actual time=56038.472..17997549.197 rows=2978820010 loops=1)
   Merge Cond: (a.id = b.id)
   ->  Sort  (cost=680346.51..692586.33 rows=4895928 width=4) (actual
time=27685.239..37493.809 rows=4895928 loops=1)
         Sort Key: a.id
         Sort Method:  external merge  Disk: 76544kB
         ->  Seq Scan on data1 a  (cost=0.00..77759.28 rows=4895928
width=4) (actual time=7.031..12175.891 rows=4895928 loops=1)
   ->  Materialize  (cost=680346.51..741545.61 rows=4895928 width=4)
(actual time=28353.126..6015294.046 rows=2978819821 loops=1)
         ->  Sort  (cost=680346.51..692586.33 rows=4895928 width=4)
(actual time=28353.122..38160.727 rows=4895928 loops=1)
               Sort Key: b.id
               Sort Method:  external merge  Disk: 76544kB
               ->  Seq Scan on data2 b  (cost=0.00..77759.28 rows=4895928
width=4) (actual time=6.085..12125.387 rows=4895928 loops=1)
 Total runtime: 23965801.345 ms
(12 rows)
```

The database server was then restarted. Setting these parameters is part science, part art, and a touch of luck. The science portion of the equation stems from the fact that the total amount of memory is known, and the size of the *high access* tables in the database may also be known. This will provide some sense as to how much shared memory if required (assuming a fairly straight forward installation). The art stems from the fact that a database is rarely static, and your initial assumptions will soon be void. Thus you must use some judgment and experience to factor in the intangible factors involved.Luck also plays a role, especially on systems that have other important processes running, as you can never be 100% sure of what the other processes will be doing at any given time, nor can you tell for sure what might happen to the database as users get a hold of it. Revisiting the configuration parameters is likely on a real world system.

If these settings are way off, generally nothing too bad will happen. Things might run slower than you hoped, or, the engine might not even start. For example, if `max_fsm_pages` is too high: see Listing 8.

If this message appears, then your eyes were bigger than your stomach, and you will need to reduce the values a bit.

The settings that were chosen in the above configuration were chosen more ad hoc than by hard calculation, as this system is not a production system, but a test system being used to see how the lack of shared memory affects query performance.

Running our test join query: see Listing 9.

We see no improvement!What could be the reason for this? The answer lies in watching how the memory is being used while this query runs. Looking at the output of vmstat (every couple of seconds) through the majority of the query run time (ie. after the initial set of disks accesses that occurs to load in the table from disk), we see something like: see Listing 10.

So there is very little paging happening for the vast majority of the query run time (the pi and po columns are the number of pages read in/out

Listing 10. vmstat output while the query is running. Very little paging is occuring

```
 procs      memory      page                    disk   faults         cpu
 r b w     avm    fre   flt  re  pi  po    fr  sr ad0   in   sy   cs us sy id
 2 0 0   1778M   318M    36   0   0   0    31   0   0    2 21598  401  1 66 33
 1 0 0   1778M   318M     1   0   0   0     0   0   1    3 574530  494  1 99  0
 1 0 0   1778M   318M     0   0   0   0     0   0   0    2 558781  400  2 98  0
 1 0 0   1778M   318M     0   0   0   0     2   0   5    7 594514  415  1 99  0
 1 0 0   1778M   318M     0   0   0   0     1   0   0    2 586000  402  1 99  0
 1 0 0   1778M   318M     0   0   0   0     0   0   0    1 565347  414  1 99  0
 1 0 0   1778M   318M     0   0   0   0     0   0   0    3 561447  415  1 99  0
 1 0 0   1778M   318M     0   0   0   0     0   0   0    2 572229  415  2 98  0
 1 0 0   1778M   318M     0   0   0   0     2   0   2    3 594159  395  1 99  0
```

Listing 11. Running a count, without any pages cached

```
test=# select current_time;select count(*) from
datat;select current_time;

       timetz
--------------------
 20:57:13.459257-04
(1 row)

  count
----------
 29375568
(1 row)

       timetz
--------------------
 20:57:31.609562-04
(1 row)

test=#
```

Listing 12. Running another count, with pages cached

```
test=# select current_time;select count(*) from
datat;select current_time;
       timetz
--------------------
 20:57:39.897516-04
(1 row)

  count
----------
 29375568
(1 row)

       timetz
--------------------
 20:57:47.151574-04
(1 row)

test=#
```

Listing 13. Running an average, without any pages cached, compared to with pages cached

```
test=# select current_time;select avg(id) from
datat;select current_time;
       timetz
--------------------
 05:19:18.395715-04
(1 row)

         avg
---------------------
 127273.413337982095
(1 row)

       timetz
--------------------
 05:19:27.290473-04
(1 row)

test=# select current_time;select avg(id) from
datat;select current_time;
       timetz
--------------------
 05:19:33.635784-04
(1 row)

         avg
---------------------
 127273.413337982095
(1 row)

       timetz
--------------------
 05:19:42.425779-04
(1 row)
```

during the interval). This means that the tables are in memory, and that most of the query time is spend simply dealing with the join complexity (all of the cross product pairs). Meaning, a faster computer would help in the running time, but more shared memory will not.The astute observer will notice that most of the cpu time it spent on system time, and not user time. This is an issue that will be examined closer in a future article.

Examining all of the traces for each run, we see that they are very similar, in fact the database index is never used, even when one is available. The reasons for this get into the query optimizer for postgres, and that is a fairly complicated topic, however

Listing 14. Running a count on distinct entries in a row (no cache vs cached)

```
test=# select current_time;select count(distinct(id))
from datat;select current_time;
       timetz
--------------------
 05:20:06.933674-04
(1 row)

 count
--------
 150146
(1 row)

       timetz
--------------------
 05:20:58.923578-04
(1 row)

test=# select current_time;select count(distinct(id))
from datat;select current_time;
       timetz
--------------------
 05:21:15.381181-04
(1 row)

 count
--------
 150146
(1 row)

       timetz
--------------------
 05:22:06.505639-04
(1 row)

test=#
```

Listing 15. Running the join under higher shared memory settings (no pages cached)

```
test=# select current_time;select count(a.id) from
data1 a,data2 b where a.id=b.id;select current_time;
      timetz
------------------
 21:00:25.4649-04
(1 row)

   count
------------
 2978820010
(1 row)

       timetz
--------------------
 21:14:33.841233-04
(1 row)
test=#
```

Listing 16. Running the join under higher shared memory settings with pages cached

```
test=# select current_time;select count(a.id) from
data1 a,data2 b where a.id=b.id;select current_time;
       timetz
--------------------
 21:15:16.905288-04
(1 row)

   count
------------
 2978820010
(1 row)

       timetz
--------------------
 21:29:02.414675-04
(1 row)

test=#
```

Listing 17. Running a count with a very low shared memory configuration (no pages cached)

```
test=# select current_time;select count(*) from
datat;select current_time;
       timetz
--------------------
 21:54:46.095708-04
(1 row)

  count
----------
 29375568
(1 row)

       timetz
--------------------
 21:55:19.721834-04
(1 row)
test=#
```

in our case, as all the data is already in memory, there is no need to use the index, it will not speed anything up. If the table was very large, then it might make sense to use an index on the join column, because the index would likely be a lot smaller than the main table (however that is not always the case, there are parameters to adjust the optimizers behavior, however changing it is not always recommended; again this is a broader topic though).

So now that we have found out how adjusting the shared memory parameters can give us very little performance benefit, when would it give us a large benefit?

Lets look at an even larger table, and see how things are handled:

```
test=# select count(*) from datat;
count
```

Listing 18. Running a count with a very low shared memory configuration (with pages cached)

```
test=# select current_time;select count(*) from
datat;select current_time;
       timetz
--------------------
 21:55:48.775529-04
(1 row)

  count
----------
 29375568
(1 row)

      timetz
-------------------
 21:56:11.23004-04
(1 row)

test=#
```

Listing 19. Running an average, with a low shared memory configuration (uncached vs. cached)

```
test=# select current_time;select avg(id) from
datat;select current_time;
       timetz
--------------------
 05:08:20.228458-04
(1 row)

         avg
---------------------
 127273.413337982095
(1 row)

       timetz
--------------------
 05:08:35.549252-04
(1 row)

test=#
test=# select current_time;select avg(id) from
datat;select current_time;
       timetz
--------------------
 05:08:46.134212-04
(1 row)

         avg
---------------------
 127273.413337982095
(1 row)

       timetz
--------------------
 05:08:55.048271-04
(1 row)
```

Listing 20. Running an count on distinct entires, with a low shared memory configuration (uncached vs. cached)

```
test=# select current_time;select count(distinct(id))
from datat;select current_time;
       timetz
--------------------
 05:13:46.329221-04
(1 row)

 count
--------
 150146
(1 row)

       timetz
--------------------
 05:14:54.122289-04
(1 row)

test=#
test=# select current_time;select count(distinct(id))
from datat;select current_time;
       timetz
--------------------
 05:15:13.467694-04
(1 row)

 count
--------
 150146
(1 row)

       timetz
--------------------
 05:16:17.952668-04
(1 row)

test=#
```

Table 3. Summary of results

Query	Low shared mem	High shared mem
count	23 – 33 seconds	8 – 18 seconds
avg	9 – 15 seconds	9 seconds
count distinct id	53 – 64 seconds	53 seconds

```
----------
 29375568
(1 row)
```

After restarting the database engine (so nothing is cached), running a count gives us: see Listing 11.

So it takes about 18 seconds. Running the same query again immediately afterwards gives us: see Listing 12.

So the count now takes 8 seconds, so it is 10 seconds faster. The difference in speed is accounted for by the fact that the pages for the table are now in memory. If the more of a table that will fit into memory, the faster subsequent queries run on that table will run.

Examining how some other aggregate functions behave: see Listing 13.

The initial run took around 9 seconds, while the second run took also took around 9 seconds.

A more complicated query, which counts the number of distinct id entries: see Listing 14.

The initial run takes 52 seconds, and the second run takes a out the same time.

Returning to the join query, when we run it with the larger shared memory settings: see Listing 15.

The running time is pretty much in line with what we saw before, (with the default configuration) 14 min and 8 seconds.If we run it immediately after: see Listing 16.

We see no speed increase, this is because all the pages already fit into memory before the shared memory parameters were increased. So increasing them did not improve the performance.

Examining a low shared memory configuration

Adjusting the shared memory parameters to be very low will allow us to test the hypothesis that the execution time for aggregate queries will be much slower than any previous run, because the table will not fit into memory:

```
shared_buffers = 1MB
temp_buffers = 1MB
work_mem = 1MB
max_fsm_pages = 17000
```

With this `low` configuration, we see that the count query over table datat is much slower: see Listing 17.

So it took 33 seconds for the initial run, and then the subsequent try: see Listing 18.

Took 23 seconds. Which is 3 times slower than during the high shared memory configuration.

For the other aggregate functions: see Listing 19.

The run time for the initial run is 15 seconds, while the second run is 9 seconds. The initial run a 6 seconds slower than the initial run in the high shared memory configuration, but the second is equal.

For the count the number of distinct id rows: see Listing 20.

The first run takes 68 seconds and the second run takes 64 seconds.So they are both fairly close. The high shared memory took 52 seconds, so this query is 18-19% slower with the low shared memory settings.

The join query behaves exactly the same way as we saw previously: see Listing 21.

The join query took around 14 minutes.

Conclusion

Given the small survey of queries, with two different shared memory configurations, we can conclude that having higher shared memory settings will increase the speed of a large number of queries.

The join query did not see much performance change over the various shared memory configurations.The next article in this series will compare the same queries over a NetBSD and an OpenBSD installation to see what impact the operating system might have (on the same hardware configuration). Then the issue of the join query performance will be investigated in greater detail.

Listing 21. The join with a low shared memory configuration

```
test=# select current_time;select count(a.id) from data1 a,data2 b where
a.id=b.id;select current_time;
       timetz
--------------------
 21:57:08.122083-04
(1 row)

   count
------------
 2978820010
(1 row)

       timetz
--------------------
 22:10:43.551956-04
(1 row)

test=#
```


About the Author

Barry Fox is a system administrator and software developer. He has been working with various flavors of BSD since the release of FreeBSD 2 in 1995.

EuroBSDCon 2009

Cambridge, UK. 18-20 September 2009

The eighth European BSD conference is a great opportunity to present new ideas to the community and to meet some of the developers behind the different BSDs.
The two day conference program (September 19 - 20) will be complemented by a tutorial day preceeding the conference (Sept 18).
The conference will take place at the University of Cambridge, England.

http://2009.euroBSDcon.org/

Triple booting

Windows 7, Ubuntu 9.04 and PC-BSD 7.1

Jan Stedehouder

In this article we will give a step-by-step guide to installing three different operating systems on the same hard drive.

Windows 7 is the upcoming new operating system of Microsoft, Ubuntu 9.04 and PC-BSD 7.1 have just been released. Getting a multi boot system is a great way to test out new operating systems on your hardware, while at the same time keeping a stable system to continue your regular work.

Disks, partitions and slices

Each of the three operating systems (OS) in this article provides an easy to use graphical installer, but some background knowledge about partitions and slices is needed to make proper use of them. It is possible to partition (or divide) a hard drive into four primary partitions maximum. We can create more partitions by setting up one primary partition as a extended partition, which in turn can have multiple logical partitions. Keep in mind that in FreeBSD-speak a partition is called a *slice*. FreeBSD, and thus PC-BSD, needs a primary partition (or slice), which in turn will be divided into logical partitions during installation.

Windows also requires a primary partition and it has a very strong preference for the first primary partition of the first hard drive (when using multiple hard drives). The new Windows 7 behaves somewhat different when installing on a clean hard drive. The OS creates two primary partitions (see Figure 1) as default, with the first primary partition amounting to 100 Mb and the second primary partition for the actual operating system (Apparently, this setup can be bypassed by not allowing Windows 7 to create additional partitions during the installation. The Windows Recovery Environment will then be installed in the root of the installation partition. However, we will have to wait for the final release to check this out).

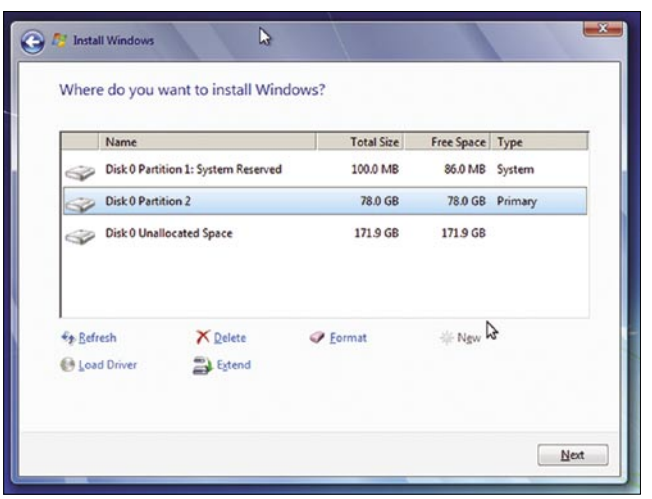


Figure 1. Windows 7 creates two primary partitions by default when installing on a blank hard drive

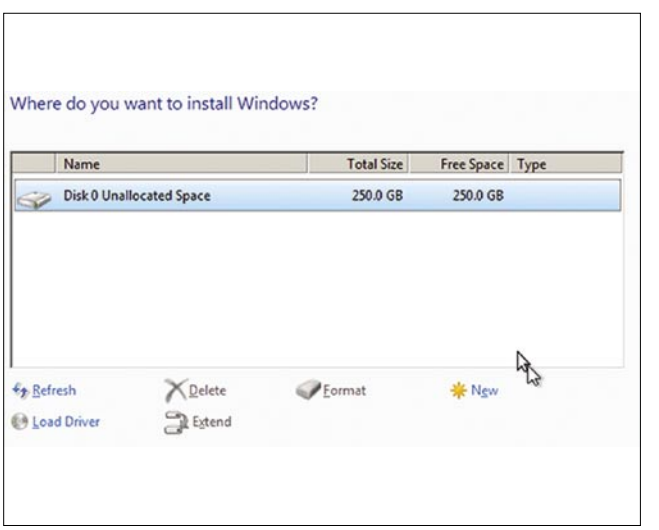


Figure 2. The partition manager of the Windows 7 installer

Ubuntu, and Linux in general, does not require a primary partition and can be installed on logical partitions without problem.

For this article I created a new virtual machine in VirtualBox with a 250 Gb hard drive. When we look at the various screenshots, you will notice the different conventions being used for this hard drive. Windows 7 uses *Disk 0* and refers to the partitions as *Partition 1* en *Partition 2*. PC-BSD uses `ad0`, `ad0s1` and `ad0s2` respectively, while Ubuntu refers to the disks and partitions as `sda`, `sda1` and `sda2`. And, while we are at it, `GParted`, to be used later on, goes for `hda`, `hda1` and `hda2`. When discussing the various steps we will use the conventions as they are used by each OS as they appear on-screen.

One last remark, for the sake of being complete I will describe the various steps for installing each OS, including Windows 7 and Ubuntu 9.04.

Step 1: Installing Windows 7

Windows 7 is the upcoming OS by Microsoft and at the time of writing the public release candidate was available. This release candidate will work until June 1st 2010 (which puts it way beyond the life span of any OS on my box). The Windows installer first asks you to select the language, the time and currency format and the layout of your keyboard. Click *Install Now* and then select which version of Windows 7 you want to install. In this article we use Windows 7 Professional. You need to agree with the license agreement. In the next step Windows 7 offers the choice between Upgrade and Custom (advanced). Select Custom for a clean install.

The question *Where do you want to install Windows?* appears with our new, clean hard drive in the box (Disk 0 Unallocated Space, see Figure 2). Click on *Drive options (advanced)*. This reveals the option to create a new partition (*New*). Click *New* and enter the size of the new partition for Windows 7 (in this case: 80 Gb). The installer notifies us that to ensure that all Windows features work properly, it will create additional partitions for systems files. Click *OK* and the new partition table reveals itself (see Figure 1).

Now simply follow the next steps in the wizard.

Wrong tangents

Before continuing with steps that will result in a working multi boot system, let us first show two possible routes that won't work. Linux users are familiar with installing their favorite OS alongside Windows and Ubuntu has made it very easy to do so. It even offers to migrate personal settings and documents from Windows to Ubuntu. Figure 3 shows a new partition table with three partitions for Ubuntu: sda5 for root, sda6 for */home* and sda7 for swap.

The next step would be to install PC-BSD, but this won't work. Figure 4 shows it's graphical installer and how it *sees* the partition table. There is no sign of the three logical partitions whatsoever.

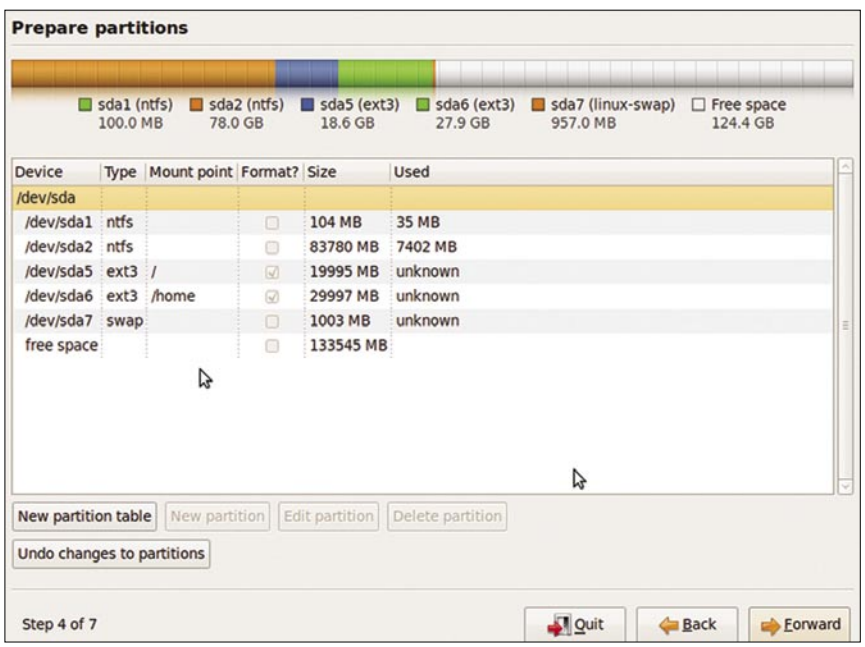


Figure 3. The partition table of a Windows- Ubuntu dual boot system...

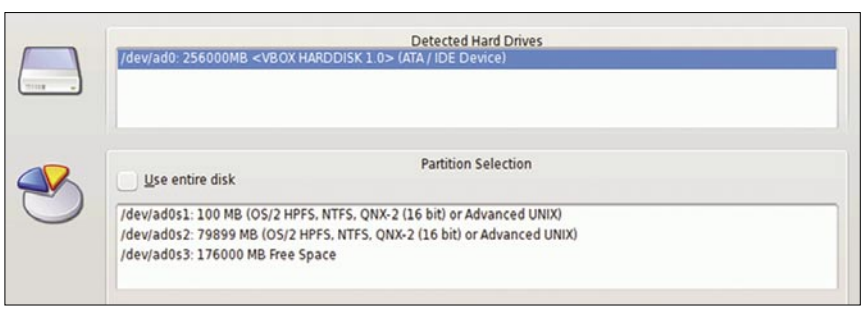


Figure 4. ... but the PC-BSD installer doesn't recognize the Linux partitions this way

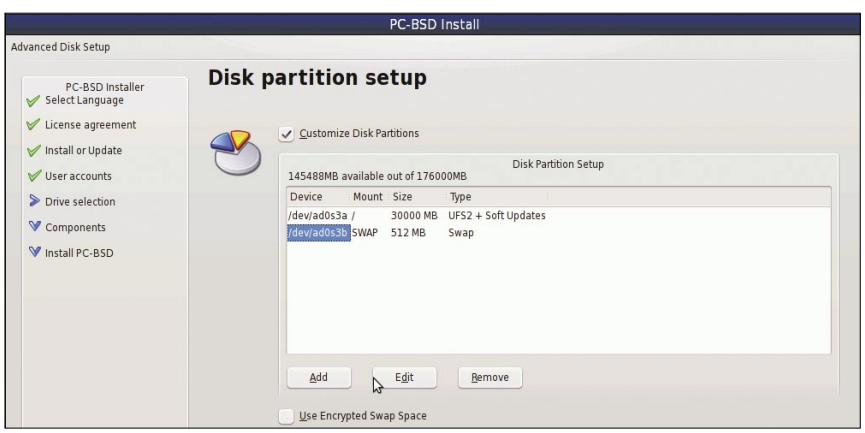


Figure 5. With the PC-BSD partitioner we can setup the disk partitions fine...

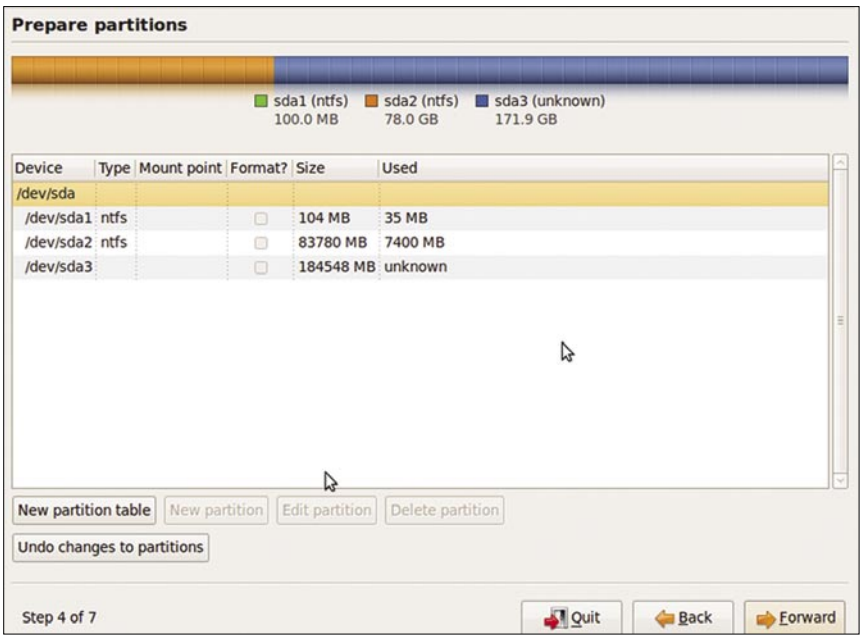


Figure 6. … but Ubuntu can't make heads or tails of them

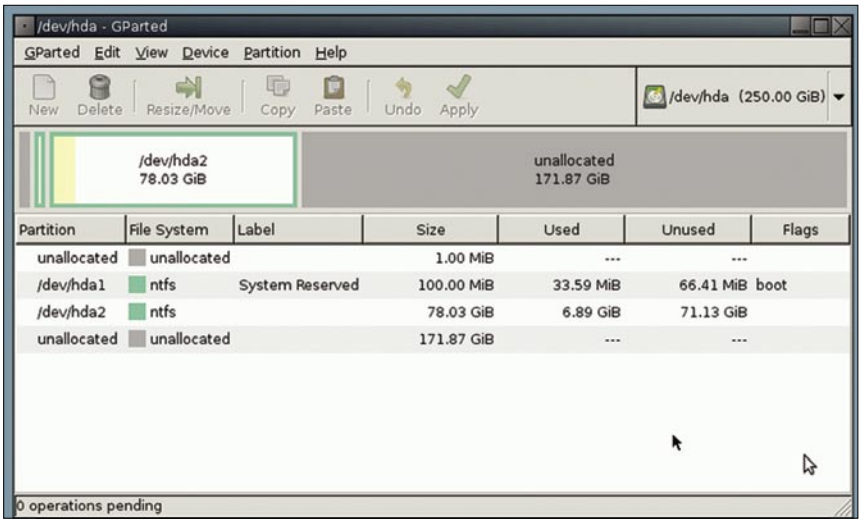


Figure 7. GParted scans the hard drive and shows the partitions used by Windows 7

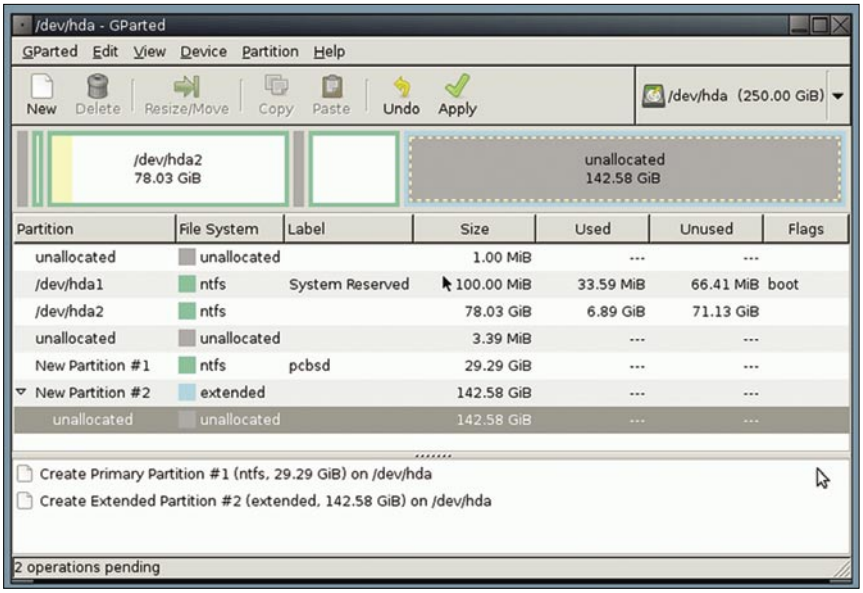


Figure 8. The new partition table as needed for installing first Ubuntu, and then PC-BSD

Changing the sequence and first installing PC-BSD won't work either. In Figure 5 we can see the slice and the logical partitions created via the PC-BSD installer. One root partition (`ad0s3a`) and a swap partition (`ad0s3b`), but as Figure 6 shows, the Ubuntu installer only sees one large space of an unknown file system.

The easiest solution is first to create a primary partition that will be used by PC-BSD. We will use GParted for this, but you can use your favorite partition manager as well.

Step 2: Preparing a primary partition with GParted

Gparted (*http://gparted.sourceforge.net/*) stands for Gnome Partition Editor and you can download a live cd of live usb image from the website. Reboot the computer (I used the cd) and select 'GParted Live' in the boot menu. The wizard asks questions about the keymap (keyboard) and whether you wish to boot into a graphical desktop (X) or the command line. GParted scans the hard drive (or drives when you have more than one disk in your computer) and shows the current partition table. We installed Windows 7 first, so GParted shows hda1 and hda2 with the NTFS file system (see Figure 7).

Now, click on *unallocated* and then on *New*. With this we create the new primary partition that will hold PC-BSD. For the article I use a 30 Gb partition, which should be large enough for extensive tests with PC-BSD. As file system I selected NTFS. It is only a dummy partition and will be reformatted in a later step. This will allow the Ubuntu installer to *see* this partition and install Ubuntu in the free space next to it. The rest of the unallocated space is converted into an extended partition. We now have a new partition table with three primary partitions and one primary partition used as an extended partition (see Figure 8).

Step 3: Installing Ubuntu 9.04

Change the GParted disk from the cd/dvd drive for the Ubuntu 9.04 cd and reboot the computer. You choose on the languages and then select *Install Ubuntu* which will launch the wizard instead of the complete live desktop. The wizard again asks you for the

language you would like to use. The next step deals with the time. Select the proper region and time zone here. The installer suggests a keyboard layout based on this choice, but you can alter it by ticking *Choose you own* and selecting one of the options. Click *Forward* to launch the partitioner.

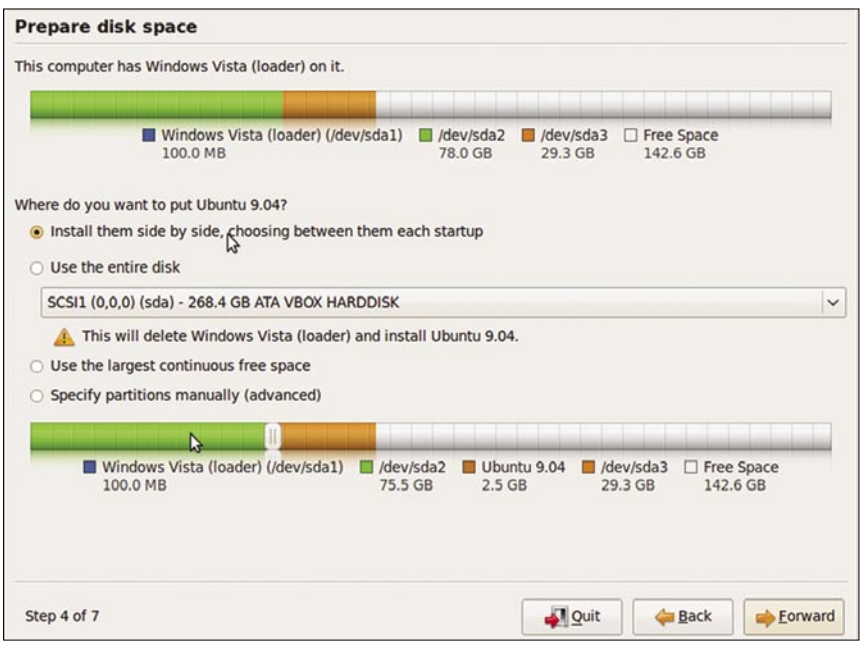


Figure 9. The partition table before installing Ubuntu

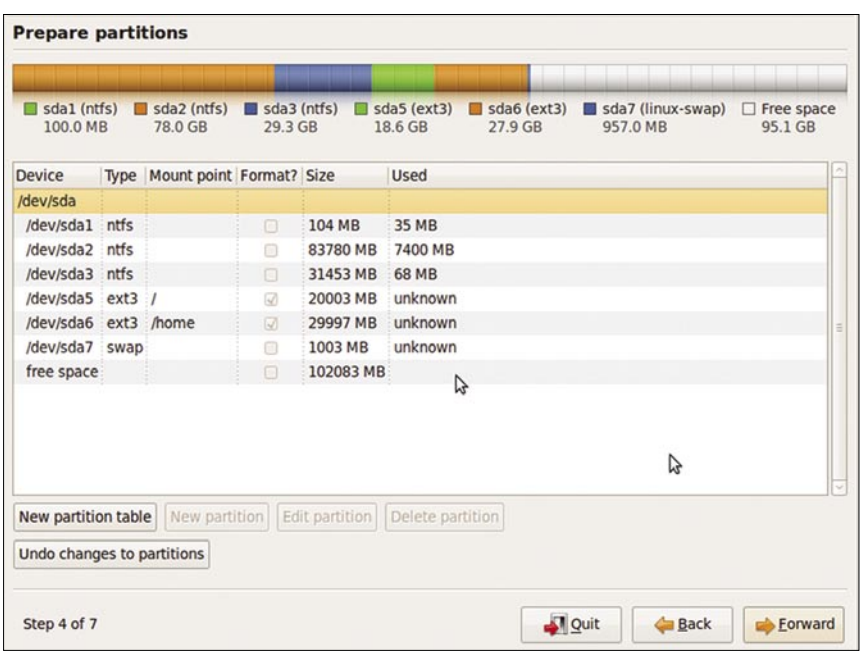


Figure 10. The partition table after adding the partitions for Ubuntu

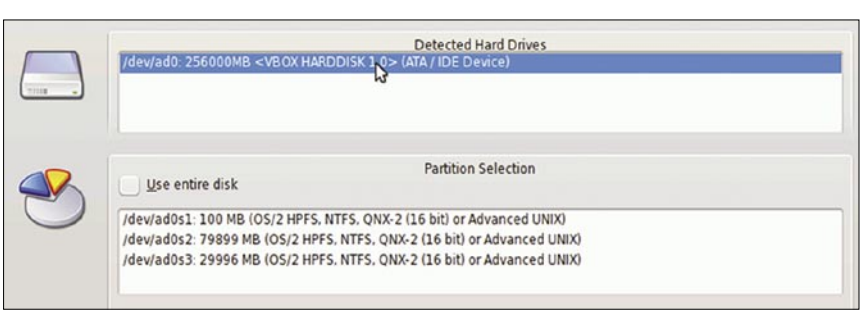


Figure 11. The new partition table as seen by the PC-BSD installer

The partitioner recognizes the various NTFS partitions we created and the empty space in the extended partition (see Figure 9).

We will create the needed partitions manually, so tick Specify partitions manually (advanced) and click *Forward*. Then click on *Free Space* and then *New partition*. We will create three new logical partitions: root (`/`), */home* (for user data) and swap. Root and */home* will use the `ext3` file system (see Figure 10).

As you can see we didn't use the entire hard drive. The current table, with 80 Gb for Windows 7, 30 Gb for PC-BSD, 20 Gb for Ubuntu root and 30 Gb for user data was sufficient for me. This way there is *some* room to install additional Linux distributions or other operating systems in the future. Of course, feel free to change the size of the partition to suit your own needs.

Why didn't we use this partitioner to create the primary partition for PC-BSD? It is possible to use it, but I find it horribly slow in recalculating the partition table after each change. And it doesn't clearly show that you have made an extended partition as container for the logical partitions, as does GParted.

From here click *Forward* to continue the installation. The wizard asks for you name, your username and password, the name for your computer and whether the user can log in automatically or not. We will allow Ubuntu to install GRUB, the Grand Unified Bootloader, the boot menu which can launch both Windows 7 and Ubuntu. Once Ubuntu finished installing, remove the cd and replace it by the PC-BSD cd or dvd.

Step 4: Installing PC-BSD 7.1

PC-BSD also has a graphical installer. The first step in the wizard deals with the system language, keyboard settings, the time zone and the question whether you wish to submit anonymous usage statistics to bsdstats.org, followed by the license agreement. In the *Selection installation choice* screen we opt for a fresh install. In step 4 you enter the root password and add the first user. Click *next* to prepare the hard drive for installing PC-BSD.

Figure 11 reveals that the PC-BSD installer sees the NTFS partition (`/dev/ad0s3`) that we prepared with GParted (and it is completely oblivious of the extended

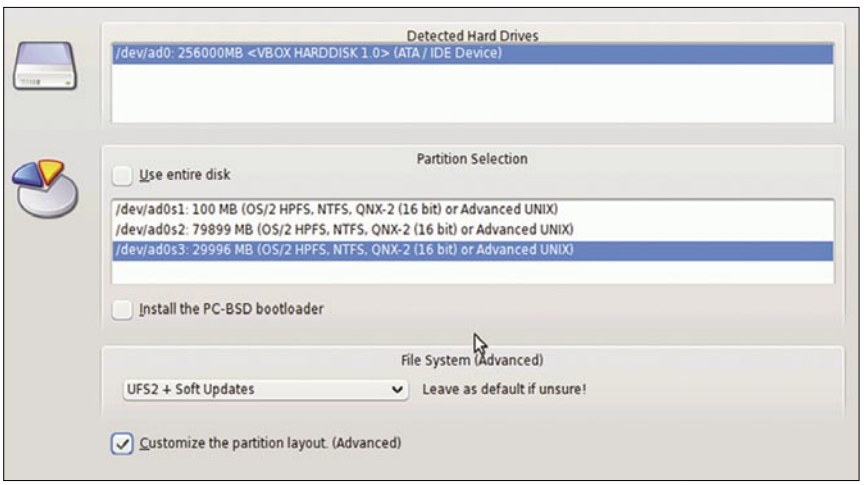


Figure 12. These are the proper selections for creating a working multiboot system during partitioning

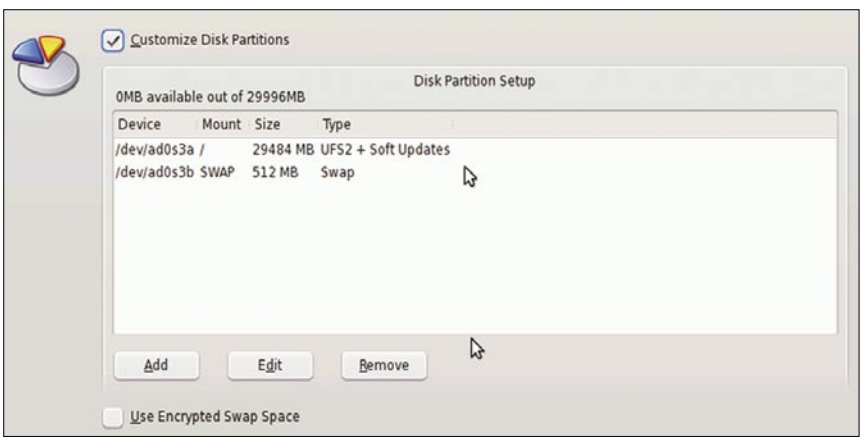


Figure 13. The logical partitions created by PC-BSD

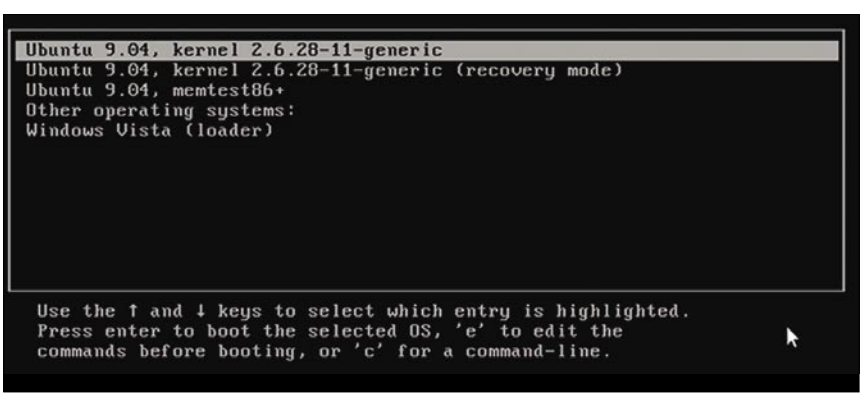


Figure 14. The bootloader GRUB doesn't have an entry for PC-BSD yet

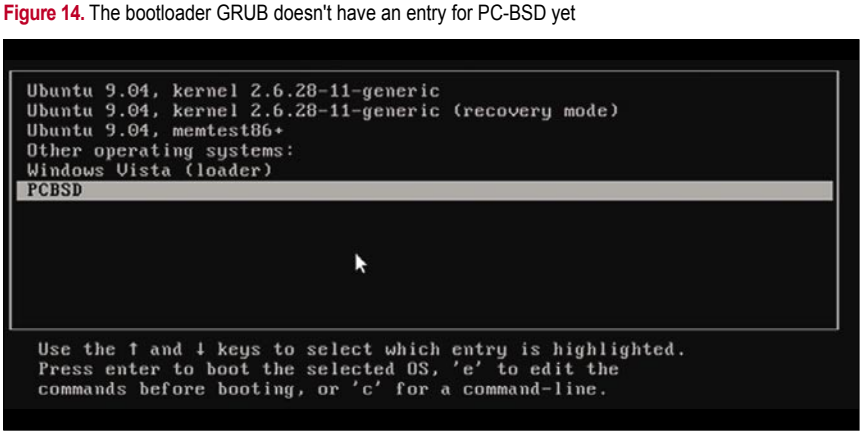


Figure 15. The 'new' GRUB with an entry for PC-BSD

partition and the logical partitions in use by Ubuntu). In order to install PC-BSD for a proper multi boot environment do the following (see Figure 12):

- click on */dec/ad0s3*;
- remove the mark in the tickbox *Install the PC-BSD bootloader*, and
- mark the box *Customize the partition layout. (Advanced)*.

We won't use the PC-BSD bootloader as it will not be able to boot Ubuntu.

Clicking *next* shows the logical partitions that PC-BSD has made in slice `ad0s3` (see Figure 13). Experienced users can alter the logical partitions at this point.

The next step deals with additional system components that we wish to install, after which we can install PC-BSD. When the installation is finished, remove the cd or dvd and reboot the system.

Step 5: Setting up the bootloader

At this point we have three operating systems and a bootloader, GRUB. When rebooting our computer we see that GRUB (see Figure 14) only has entries for two operating systems: Ubuntu 9.04 and Windows Vista (actually Windows 7).

We need to add PC-BSD to GRUB in order to be able to use it. Please launch Ubuntu 9.04 and then open a terminal. Enter the following command:

```
$ sudo gedit /boot/grub/menu.lst
```

This command will open the file *menu.list* in the editor *gedit* with administrator rights. Menu.lst holds the entries for GRUB and points the bootloader to the proper partitions with boot sectors. Go to the end of the document, where we see the entry:

```
# This entry automatically added by
the Debian installer for a non-linux
OS
# on /dev/sda
title                    Windows
Vista (loader)
rootnoverify      (hd0,0)
savedefault
makeactive
chainloader              +1
```

For GRUB to boot PC-BSD add the following lines:

```
# This entry was manually added by me
to boot PC-BSD
# on /dev/sda
title                          PC-BSD
rootnoverify       (hd0,2)
savedefault
makeactive
chainloader                    +1
```

GRUB points to (`hd 0,0`) to boot Windows 7, which would translate into `sda1` or `ad0s1`. (`hd 0,1`) (`sda2` or `ad0s2`) holds the actual Windows OS. We installed PC-BSD on primary partition sda3 or `ad0s3`, thus the use of (`hd0,2`). With this GRUB now has an entry capable of booting PC-BSD (see Figure 15).

Alternative bootloader

If this were only a one-time installation using GRUB would be fine. But, once you get into multiboot installations, chances are that you do want to change one or more operating systems, or add new ones to the mix. The Windows 7 release candidate is valid until June 2010, but the moment you install the final release (or revert back to Vista or XP) Windows will overwrite the master boot record (MBR) and GRUB will be gone. Thus, to create a more robust multiboot system we do well to use an alternative bootloader, one that doesn't rely on one of the underlying operating systems.

One alternative is GAG, short for Gestor de Arranque Grafico, which is Spanish for Graphical Boot Manager. GAG (*http://gag.sourceforge.net/*) is a third party tool that helps to set up a boot menu for each of our operating systems. It is quite versatile and allows a menu for up to nine different operating systems. When you download and extract the zip package from the website, you will find the file *cdrom.iso*. Burn it on a cd, put it in the cd/dvd drive and reboot the machine, then select the option *Install GAG*. The next steps ask for your keyboard type and language, after which you select the option *Setup GAG keys*.

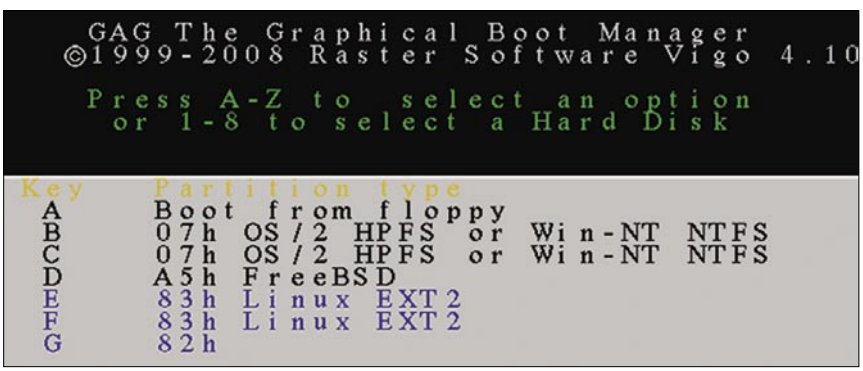


Figure 16. The partition table as recognized by GAG

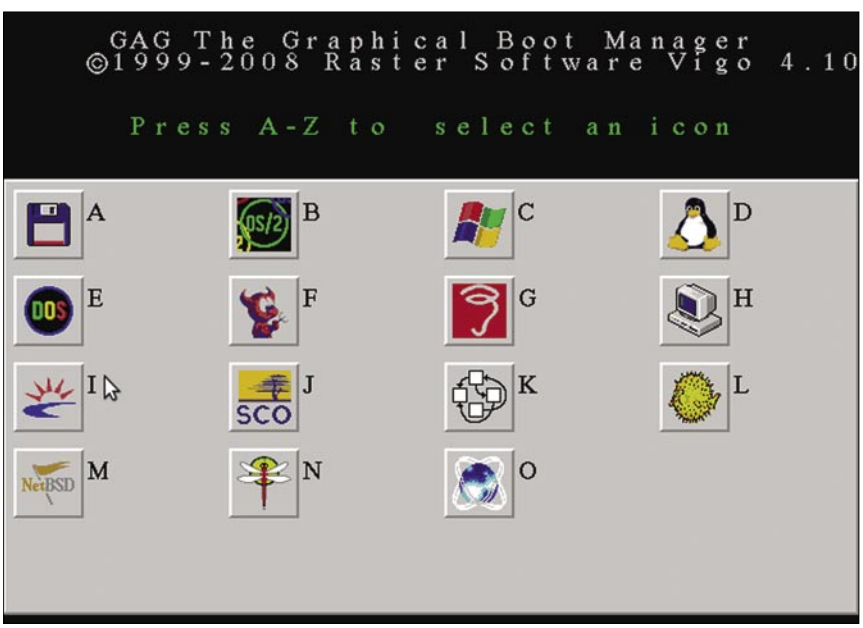


Figure 17. The GAG icons give an indication of the supported operating systems

GAG doesn't automagically scan the harddrive and add each operating system. We need to do that manually by selecting *Add a new operating system*, which reveals a list of recognized operating systems as can be seen in Figure 16.

The keys B and C are NTFS partitions. As we have noticed before, Windows 7 creates two primary partitions by default, with the first partition holding the boot sector. Press B and type your descriptive name for Windows 7. GAG will ask you to provide a password. When you don't need a password to prevent unwarranted booting into the operating system, simply press Enter. In the next step you can select one of the icons (see Figure 17).

Repeat this step for each operating system. In our example PC-BSD can be found under key [*D*] and Ubuntu under key [*F*]. However, booting Ubuntu this won't work now. If you plan to use GAG as a bootloader, you need to install the bootsector of Ubuntu on it's root partition. This option is available in the last step of Ubuntu's installation wizard. From here on you can add new Linux distributions, each booting from it's own root partition, via GAG.

To wrap it up

Creating a multiboot installation isn't complicated, simply a matter of keeping a few basic principles in mind,having some simple tools and clear mind. True, using virtual machines often does the trick, but multiboot installations allow you to use the full strength of your hardware for each of the operating systems.

About the Author

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

BuildaSearch a FreeBSD Web Service

Diego Montalvo

BuildaSearch is a web service which allows users to build a custom search engine or site search in less than five minutes. No coding skills are necessary when building a custom search. Users can customize their colors, backgrounds, logos, and search results.

BuildaSearch makes complex search services simple by providing users with a simplified interface and a copy and paste search for external websites.

The service provides search solutions for all skill-sets, from the Internet beginner to a business website looking for on-demand indexing search services.

Released in late April of 2009, *BuildaSearch Advanced Search* (BAS) is an on-demand indexing search service which creates the latest search content from a list of custom domains. Since BAS is an extremely resource intensive service always creating, updating, deleting, searching, and producing fresh results, FreeBSD is the perfect back-end for such an intensive web service. BAS is capable of searching anywhere from one to thousands of websites.

Technologies behind BuildaSearch services are the following: HTML, CSS, XML, Javascript, Shell programming, C++, AJAX, SQL, PHP all working together on BSD back-ends. BuildaSearch currently runs on FreeBSD versions 7.0 and 7.1.

In order to provide users with their choice of custom results the following APIs are currently used: Yahoo! BOSS, Microsoft Live, DuckDuckgo.com, Mnogosearch and Amazon Web Services.

Due out in late May *Search Clash* will become the future of BuildaSearch, opening up BuildaSearch to virtually any XML/ RSS based data source.

Taking it a step further data sources will be able to *Search Clash* meaning data sources once totally separate will be searchable together in synchrony in one cool search.

We are currently developing the future of our service which will allow virtually any distributed data source to be integrated into BuildaSearch services. Our *Search Clash* service will consist of three components: SDE (Search Development Environment), Third Party data sources, and BuildaSearch XML based API. Continuing the simplicity of BuildaSearch, our SDE will allow users

Figure 1. User Interface for designing and modifying search layout, search providers and cool colors

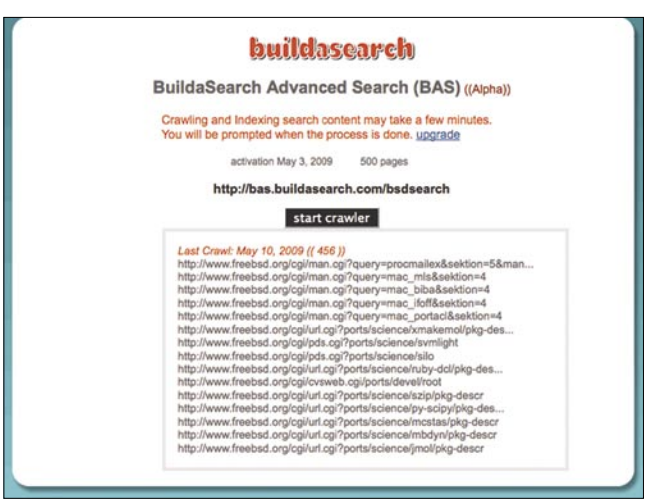


Figure 2. BuildaSearch Advanced Search (BAS) Crawling and Indexing BSD related sites

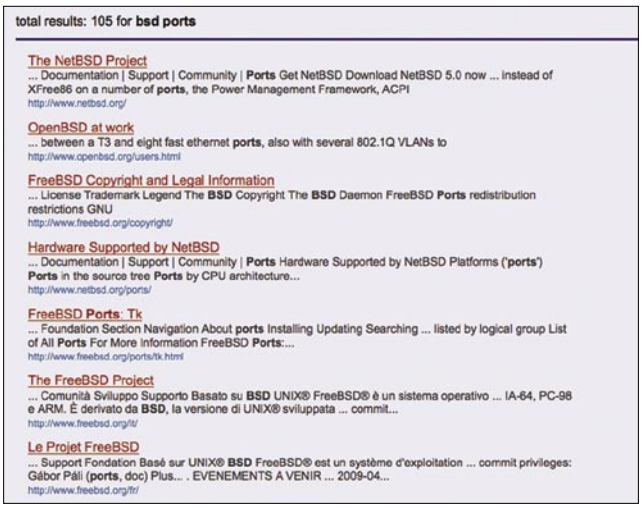


Figure 3. BAS search engine of BSD related sites

Figure 4. Above is an early example of two different data sources in a search clash. The screenshot illustrates a designer jeans search provided by the Yahoo! API and BuildaSearch (BAS) API

to build, test, and rollout custom *Search Clashes* without any coding.

Third-party data will provide unlimited content for building custom search engines. The BuildaSearch API will allow users to easily integrate their *Search Clash* into any blog or business website, regardless of programming language.

Have seen more XML in eight months since BuildaSearch was created than I have in 9 years of programming.

The concept behind BuildaSearch began with the decision to reuse already created and tested code for a search engine project. It was the reusing of code which began the idea of building a search service which would allow users to create a customized search engine with a few clicks of their mouse.

From the beginning BuildaSearch was to become a service which would make search engine creation and customization as simple as microwaveable popcorn.

BuildaSearch is tied into BSD at every step of development and deployment.

Initial creation and current development of BuildaSearch is all done using MacOS a derivative of FreeBSD. The great stability of FreeBSD has allowed BuildaSearch to provide users with the simplest user interface while taking care of mission critical tasks. The first version of BuildaSearch took a couple of months to create and was released in August of 2008. Whether you feel like creating a child-safe search engine or providing your business website with a fast-reliable search, BuildaSearch is the Occam solution.

All in all, as BuildaSearch provides more and more complex services, our goal is to always provide users with the simplest solution regardless of service complexity. The BuildaSearch motto is *Build a Custom Search in Less Than 5 Minutes*.

BuildaSearch was created by Diego Montalvo a Texas cowboy living in sunny San Diego. Stephanie Lester of *hot-summer-days* Arizona helps with everyday marketing and creative thinking.

How it all started...

I began to develop database driven web applications in 2000 and it all began with ASP and SQL Server. At the time I was working on a WAP search engine running on Windows NT, but soon realized that users manually entering data was never going to see potential. After a tip off of something called *BSDi* from Graham Toal co-creator of CDDB, I got online and found that BSDi had a brother *FreeBSD*. After ordering a FreeBSD CD, and installing the operating system, I realized never again would I use anything on a server which wasn't BSD. In late 2001 my first web service running on FreeBSD was released as one of the first WAP search engines in the world. Throughout the years FreeBSD has always been my primary server operating system. Have tried different distributions of Linux but still find FreeBSD to be my favorite by far.

DRAFT Notes

In the past it would have been quite difficult and, but with the inception of AJAX, BuildaSearch could be created:

- as a search project for a client, which soon became it's own search service.
- turned into as many companies Yahoo! And Microsoft where opening up their search APIs
- BuildaSearch is entire back-end is running on FreeBSD version 7.0 and 7.1.

buildasearch

Figure 5. Illustration above is our current business logo

Web Servers
for Embedded NetBSD

Donald T. Hayford

Web-based user interfaces have become ubiquitous for all sorts of electronic gear these days. If you are building a network-capable device, chances are you will want to add a web server to your device's software, as well – it's generally cheaper than a hardware interface and far easier to change or update.

These days, the use of analog controls (i.e., potentiometers and switches) has pretty much gone by the wayside and the entire device is controlled through a software interface, so why not go the next step and eliminate the front panel altogether? Network routers have been that way for years, and other companies are also making the change. The Agilent (*www.home.agilent.com*) 6000L series of rackmounted oscilloscopes, for example, have no display, relying mostly on a built-in web interface for user interaction and display of captured data (though, in truth, there is something contradictory about a 'scope without a scope). Note the important distinction here: this is not an instrument that plugs into a PC slot or USB port and has a graphical user interface. Rather, the device itself is the web server and the only function of the PC is to host the browser. Besides saving you money in production, a web server can also provide an easy window into your device during development at the cost of a few web pages and a little bit of code development.

According to Netcraft (*www.netcraft.com*), the most popular web server on the planet is Apache, running on 45% of all servers, followed by MS-IIS on an additional 30%. If you're setting up a desktop server with lots of RAM and disk space, you can certainly use either of these as a web server. But if you are building, say, a small device with an ARM processor and 16 to 32 MB of memory, what do you use? As it turns out, there are a number of choices for lightweight web servers that run on NetBSD, several of which are included as part of NetBSD's standard package system. In this article, we'll show you how to get, build, and run several web servers on your embedded NetBSD system.

The *light* part of lightweight servers can be defined in several ways – absolute size of the server application, the number of files that must be installed to make the server work, or the impact the application has on system performance through factors like memory and swap-space usage, speed of accessing peripherals, and the like. Built for the ARM processor, the Apache executable file is more than a megabyte in size. And installing Apache involves putting a lot of files into numerous locations – certainly a non-trivial task even with an installer. The five servers we will look at compare favorably to Apache in all of these aspects. Table 1 shows the names and websites for each server, along with the versions and release dates I looked at here. If you want to learn about even more small web servers, check out References 1 and 2.

Most modern web servers offer two features that are important to your system – chroot and CGI. The first stands for *change root*, and causes the program in question to act as if the specified directory is root, loosing the ability to access directories at a higher level. This very positive feature means that, should there be some security problem with the server, the damage will probably be limited to the directory in question. This may be particularly important when working with web servers that offer a CGI capability, as well. CGI stands for Common Gateway Interface and is a method by which you (the web server programmer) can allow the user to run certain types of scripts (programs) on the server machine to provide a more dynamic web interface. There are some security implications with allowing CGI and being able to *chroot* the server can help protect your machine. Of the web servers examined here, all offer CGI services, and all but one allow chrooting.

What you'll need:

- *An embedded processor running NetBSD.* This can be an old Pentium machine, a Linksys NSLU2 (my personal choice), or one of the many other embedded processors

that run NetBSD. It's up to you to get NetBSD running on this machine, but there are lot's of examples and tutorials to work from. See the NetBSD wiki (*wiki.NetBSD.se*) for more information, or earlier issues of this magazine. I assume that your embedded machine has a disk drive for compiling the necessary applications; if not, you'll need to use your development environment to build the web servers described below.

- *The latest version of pkgsrc.* See Listing 1 for the command line entries required to get the latest stable version of pkgsrc (at the time of this writing).
- *A favorite web page that you would like to use for testing.* If you're a web guru, you can make up your own. If not, you can use the web page contained in Listing 2. This page displays an image and a little text, then updates itself every second so you can see how your machine will perform with a modest load without having to continuously click the reload button. If you have your own image you'd like to display, then rename it test.jpg (assuming it is a jpeg image) and save it in the proper folder. I *borrowed* an image from the web page for BSD magazine. To do this with Firefox, go to the web site (*www.bsdmag.org*, for example) and select *Save Image As...* when you right-click your mouse on the image of the magazine cover. Save it as test.jpg. The browser view for this web page is shown in Figure 1.
- *A modern web browser.* Any browser based on Mozilla will work, as will Internet Explorer.

Table 1. Web servers examined in this article

Software Title	Version	Release Date	Web Page
nginx	0.5.35	01/08/08	*nginx.net*
boa	0.94.14-rc21	02/23/05	*www.boa.org*
thttpd	2.25b	12/01/03	*www.acme.com/software/thttpd*
lighttpd	1.4.20	09/30/08	*www.lighttpd.net*
Bozohttpd (httpd)	20080303	03/03/08	*www.eterna.com.au/bozohttpd* (www.netbsd.org)

Table 2. Web server code size and CPU usage

Software Title	C/C++ Source Files	Header files	Line count *(*.c + *.h)*	Compiled Size *(kB)*	Run Size (kB)	CPU Usage(150 kB/sec - *5 requests/sec)*
nginx	94 files *1247 kB*	76 files *428 kB*	51557	392	1916	5.1
boa	26 files *308 kB*	7 files *38 kB*	11068	211	1076	5.4
thttpd	8 files *226 kB*	10 files *54 kB*	9808	78	1260	6.6
lighttpd	91 files *1404 kB*	41 files *106 kB*	55115	210	1640	5.3
bozohttpd (httpd)	8 files *102 kB*	2 files *9 kB*	3768	55	1272	5.3*

*estimated (see text for more details)

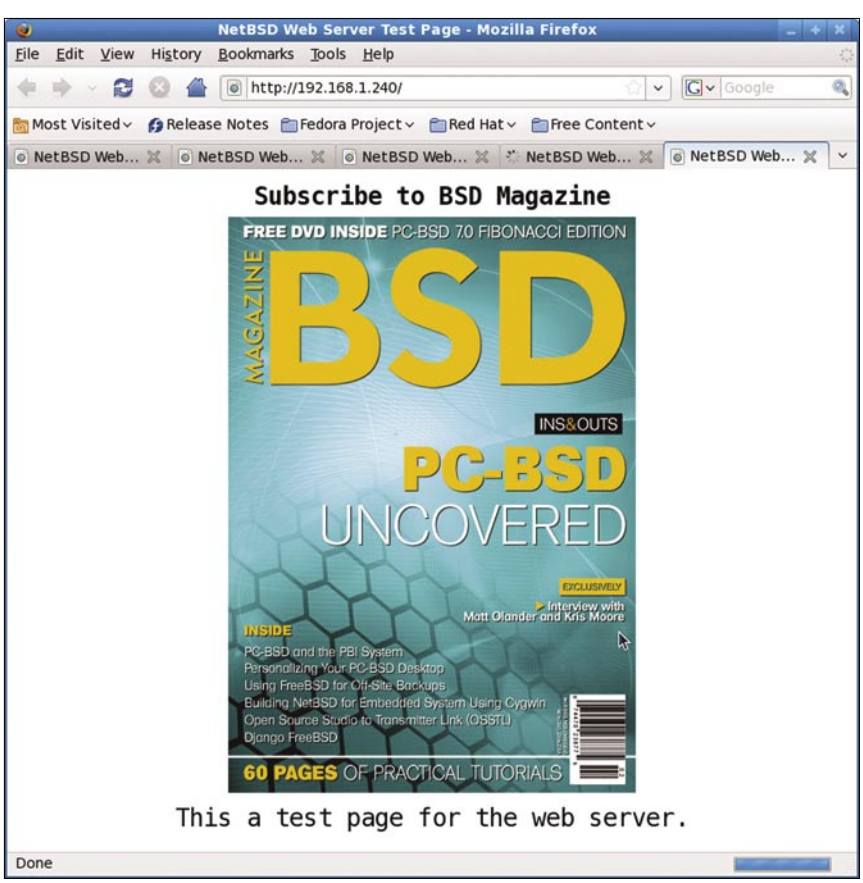


Figure 1. The output of the test web page sent by the Boa web server displayed using Firefox

I tested the web servers on a Linksys NSLU2, with a 200 MHz ARM processor, 32 MB of RAM, and a 100-Base-T ethernet port, running version 4.99.61 of NetBSD. Five different browser windows were used to request the test page in Listing 2. Since the page causes the browser to request an update from the server once a second, the web server ends up serving 18000 pages every hour. I used the program top to keep track of memory and total CPU usage over a 12 hour period, during which time the server delivered approximately 216,000 page requests. While five pages a second doesn't pose much of a burder for a mainframe web server that can handle hundreds or thousands of pages

per second, it iseems a reasonable expectation for a small processor to handle without significant performance degradation.

Nginx

According to Netcraft, Nginx (Figure 2.a) is the fifth most popular web server in the world, serving more than six million domains in April, 2009, including the popular sites *hulu.com* and *wordpress.com*. For all of it's power, it is surprisingly small, coming in at roughly 392 kB when built for the ARM processor (see Table 2). According to the wiki (*wiki.nginx.org/Main*):

Listing 1. Test web page source code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/
html4/strict.dtd">

<html><head><meta content="text/html; charset=ISO-8859-1" http-
equiv="content-type"><title>NetBSD Web Server Test Page</title></
head><body>
<table style="text-align: left; width: 100%;" border="0" cellpadding="2"
cellspacing="2">
  <tbody>
    <tr align="center">
      <td style="vertical-align: top;"><big style="font-weight:
bold;"><big>Subscribe to BSD Magazine</big></big><br>
      </td>
    </tr>
    <tr align="center">
      <td style="vertical-align: top;"><img style="width: 400px; height:
565px;" alt="BSD Magazine Cover" src="test.jpg"><br>
      </td>
    </tr>
    <tr align="center">
      <td style="vertical-align: top;"><big><big>This a test page for the
web server.</big></big><br>
      </td>
    </tr>
  </tbody>
</table>
<script language="javascript">setTimeout("window.location.reload(true)",10
00);</script>
</body></html>
```

Listing 2. Getting the latest pkgsrc

```
-bash-3.2$ cd /usr
-bash-3.2$ su
-bash-3.2# export CVSROOT="anoncvs@anoncvs.netbsd.org:/cvsroot"
-bash-3.2# cvs co -r pkgsrc-2008Q4 -P pkgsrc
```

Listing 3. Getting and Building the Boa Web Server

```
-bash-3.2$ mkdir /home/hayford/boa
-bash-3.2$ cd /home/hayford/boa
-bash-3.2$ wget http://www.boa.org/boa-0.94.14rc21.tar.gz
-bash-3.2$ tar -xzvf boa-0.94.14rc21.tar.gz
-bash-3.2$ cd boa-0.94.14rc21/
-bash-3.2$ ./configure
-bash-3.2$ make
```

Unlike traditional servers, Nginx doesn't rely on threads to handle requests. Instead it uses a much more scalable event-driven (asynchronous) architecture. This architecture uses small, but most importantly, predictable amounts of memory under load.

Even if you don't expect to handle thousands of simultaneous requests, you can still benefit from Nginx's high-performance and small memory footprint. Nginx scales in all directions: from the smallest VPS [note: virtual private server] all the way up to clusters of servers.

Best of all, *Nginx* is easy to build and install since it's supported by NetBSD's packages. Follow the instruction in Listing 2 to build *nginx*. The source code for Nginx comes in 170 files with almost 2 MB of code (including comments) – see Table 2 for more details. The build script will place the *nginx* binary in `/usr/pkg/sbin` and half a dozen other files, including the *nginx* configuration file *nginx.conf*, are put in `/usr/pkg/etc/nginx`. If you don't make any changes, *nginx* expects to find the start-up web page, *index.html*, at `/usr/pkg/share/examples/nginx/html`. Either change the configuration file or move your test web page and data to this location.

To start and stop nginx for testing, use:

```
-bash-3.2$ sudo /etc/rc.d/nginx
onestart
```

and

```
-bash-3.2$ sudo /etc/rc.d/nginx
onestop.
```

You need to use the control words *onestart* and *onestop*, instead of *start* and *stop*, because *nginx* hasn't been enabled in `/etc/rc.conf`. Once you're satisfied with how nginx is running, you can enable it permanently by adding the line

```
nginx=yes
```

to `/etc/rc.conf`. Now, *nginx* will start up every time you boot NetBSD.

Besides being the largest executable we will look at, Nginx also uses the most memory during execution. Memory usage will jump by

about 300 kB for the first web page request received, then by about 4 kB for each subsequent page. During a 12-hr run serving 5 pages/persecond, the amount of memory used by Nginx grew to just under 2 MB. Some significant part of this memory is due to the fact that Nginx will keep a connection to a browser *alive* for some period of time to speed up subsequent interaction. The default *keepalive timeout* for Nginx is 65 seconds, meaning that Nginx will keep the connection around for that long waiting for the browser to make another request (it saves having to keep reallocating memory, for one thing). For an embedded application where only a single user will connect, that's ok. If you're using your embedded machine to serve web pages in a more normal scenario, you might end up with a lot of only slightly used connections that hang around for a while before they are destroyed and the memory becomes available for reuse. Adjust the timeout period in the configuration file if this becomes an issue.

Serving up our test web page and image at a rate of 5 pages per minute takes about 5 percent of the CPU (see Table 2) for *Nginx*. As you can see, that's about the same for all of the servers, so at least for simple pages, CPU usage will not be the deciding factor for selecting among these web servers.

Thttpd

The leading *t* in thttpd (*www.acme.com*) stands for *tiny*, *turbo*, or *throttling*, depending on your preference. According to the author, thttpd was the only web server that could selectively control the rate at which data is sent to the browser as far back as 2003. Now, several servers have that capability but it still fits the bill quite well as a tiny web server. Two other servers are also listed on the web site – `mini_httpd` and `micro_httpd` – the latter written using only about 200 lines of code. Both are also available as part of the NetBSD packages, but were not evaluated here. One possible problem with thttpd is that development seems to have stopped on the software, since the thttpd web site and source code have not changed since 2003. One positive selling point is that thttpd definitely has the coolest logo, shown in Figure 2.b.

Thttpd is the second smallest of the five servers discussed here in terms of code size and compiled size (see Table 2), and the smallest executable after allocating the necessary memory. It was also the slowest, clocking in at 6.6% CPU usage for serving up 5 copies of the test page (but see the discussion on bozohttpd). Thttpd is easy to build and install. Switch to the appropriate directory (`/usr/pkgsrc/www/thttpd`) and do the standard *make install clean* dance. When finished, you'll find the executable (*thttpd*) in `/usr/pkg/sbin` and the very simple configuration file (`thttpd.conf`) in `/usr/pkg/etc`. As with the others, you can start the web server using the *onestart* option or you can enable the server in `rc.conf` and reboot.

Lighttpd

Like nginx, lighttpd (Figure 2.d) is used by a respectable number of web sites – according to Netcraft, nearly 3 million in January, 2009. Lighttpd has the largest code base of the servers examined here, hitting nearly 2 MB of code. Lighttpd is actively maintained. Executable size is about half of nginx at around 200 kB, and the maximum amount of memory used when serving five pages per second was 1640 kB. Building and installing is the same for lighttpd as for the rest of the packages available in the package source. The configuration file is found at `/usr/pkg/etc/lighttpd/lighttpd.conf` and the executable is located in `/usr/pkg/sbin`. Start lighttpd by using the *onestart*

Listing 4. Steps to build nginx

```
-bash-3.2$ cd /usr /pkgsrc/www/nginx
-bash-3.2$ sudo make install clean
```

Listing 5. Modifying the Boa configuration file boa.conf

```
# Boa v0.94 configuration file
<snip>
# Port: The port Boa runs on.  The default port for http servers is 80.
# If it is less than 1024, the server must be started as root.
Port 8080
#Port 80
<snip>
# ErrorLog: The location of the error log file. If this does not start
# with /, it is considered relative to the server root.
# Set to /dev/null if you don't want errors logged.
# If unset, defaults to /dev/stderr
ErrorLog /home/hayford/boa/error_log
#ErrorLog /var/log/boa/error_log
<snip>
# Access logging.
AccessLog /home/hayford/boa/access_log
#AccessLog /var/log/boa/access_log
<snip>
# DocumentRoot: The root directory of the HTML documents.
# Comment out to disable server non user files.
DocumentRoot /home/hayford/boa
#DocumentRoot /var/www
<snip>
# MimeTypes: This is the file that is used to generate mime type pairs
# and Content-Type fields for boa.
# Set to /dev/null if you do not want to load a mime types file.
# Do *not* comment out (better use AddType!)
MimeTypes /home/hayford/boa/mime.types
#MimeTypes /etc/mime.types
<snip>
```

option or enable it in `/etc/rc.conf` to start it on boot up.

Boa

Boa (*www.boa.org*, Figure 2.c) is a small, single-tasking HTTP server that operates within a single thread, internally multiplexing all incoming requests. According to documentation on the Boa website, the server can handle several thousand hits per second on slow 300 MHz Pentiums and dozens of hits per second on a 20 MHz 386. The strange selection of hardware in the quoted benchmarks point out one issue with Boa – though popular in the embedded Linux community because of its small size, Boa does not seem to be under active development. Version 0.94.13, the last stable release, came out in 2002 and is available from SourceForge but that version won't build under NetBSD. The more recent (2005) version, 0.94.14rc21, will build under NetBSD and is available from the Boa website. There are 26 source code files (`*.c`) and seven header files (`*.h`). To get and build Boa, follow the command line instructions in Listing 4.

The out-of-the-box build of Boa is 212 kBytes in size – much larger than what I expected given the amount of source code. Though I didn't check, Boa probably includes a largish chunk of memory that is initialized or static and thus included in the executable. The website reports that Boa has been built as a 32 kB executable on a uCLinux system, so presumably you can make it smaller if you work at it. You'll also need to install Boa manually, though that's not hard. There are several items in the configuration file that you'll need to fix, which you will find in `~/boa/boa-0.94.14rc21/boa.conf`. The lines I changed are shown in Listing 5, and are pretty self-explanatory other than the need for a file that Boa doesn't provide (but see below for `mime.types`). Boa doesn't support *chrooting* for security purposes, but relies on the user/group in the configuration file (if you run as root) or your user/group identity to limit the files that Boa can read and serve. This is probably acceptable for a server on a limited embedded processor, but almost certainly not ok for a more sophisticated system. Don't forget that if you don't run as root, you can't use port numbers less than 1024. Listing 5 assumes you are running Boa as yourself, so point the browser URL appropriately (i.e., 192.168.1.240:8080/, for my particular setup).

One difficulty in running Boa is that it needs a file mime.types, which in Linux systems is put in the /etc directory but doesn't seem to be hanging around in NetBSD unless you've installed certain other software. The file itself is just a listing of file extensions and the associated mime types so that Boa knows what to do when asked to send a certain file type – for example:

```
 application/octet-stream bin dms lha
lzh exe class so dll img iso
 text/html                  html htm
 text/parityfec
 text/plain                 asc txt
text pm el c h cc hh cxx hxx f90
```

If you don't have the file, Boa will assume everything is text and the web page will be displayed as a text file, not as a web page. You can get a copy of mime.types from `/usr/pkgsrc/mail/sylpheed/files`. As you can see from the boa.conf file, I put a copy of this file in the directory I used to build boa.

To run Boa, just put a copy somewhere in the path (or include the path in the invocation) and tell it where the configuration file is located. Boa moves itself to the background and continues to monitor the port/address combination you specified in the configuration file. To stop Boa, you'll need to find its process id (use ps waux) and then kill it. For example:

```
   -bash-3.2$ ./boa-0.94.14rc21/src/
boa -c /home/hayford/boa
      <web server is active>
   -bash-3.2$ ps -waux | grep boa
      hayford 27692  0.0  3.1 1008
1000 ttyp0 S
   bash-3.2$ kill 27692
      <web server stops>
```

bozohttpd (NetBSD httpd)

According to the author, the main feature of *bozohttpd* is its lack of

Listing 6. Modification to /etc/inetd.conf to run bozohttpd as a inetd resource instead of a daemon

```
#       $NetBSD: inetd.conf,v 1.58 2007/10/16 02:47:14 tls Exp $
#
# Internet server configuration database
#
#       @(#)inetd.conf  8.2 (Berkeley) 3/18/94
#
http    stream    tcp    nowait:600    _httpd  /usr/libexec/bozohttpd
bozohttpd   /var/www
http    stream    tcp6   nowait:600    _httpd  /usr/libexec/bozohttpd
bozohttpd   /var/www
#
<snip...>
```

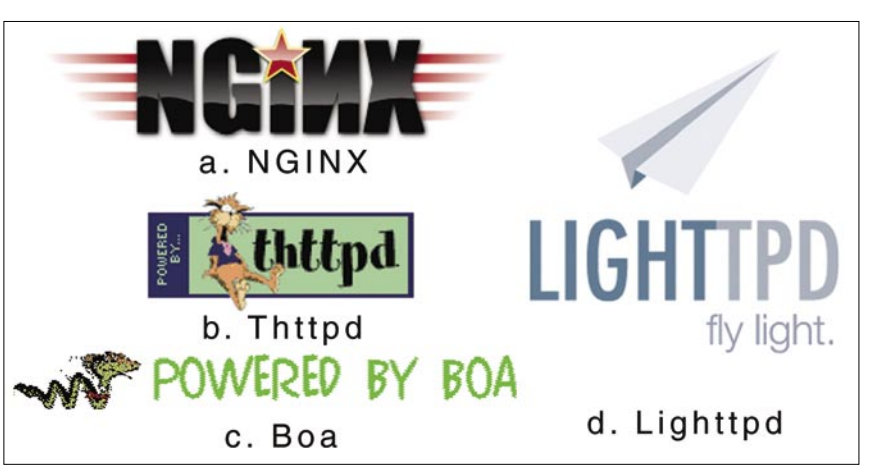


Figure 2. Web server logos. (Note: Bozohttpd doesn't have a logo)

features, thereby reducing code size and improving security. It is certainly the smallest of the web servers that we are examining here – version 20080303 checks in at 55 kB for the compiled code. It is also unique in that it was originally written to run on NetBSD and has now been ported to other *nixes. It, and it's author, should receive some kudos just for that. Bozohttpd supports CGI/1.1 and HTTP/1.1, HTTP/1.0, and HTTP/0.9. Added to NetBSD-5, meaning it appears in versions later than 4.99.1, bozohttpd is the default web server for NetBSD, appearing as /usr/libexec/httpd. Note that the name interferes with Apache, which also calls itself *httpd*, though Apache normally installs itself to /usr/pkg/sbin. It is, however, still offered as a standalone package in the source packages for those running earlier versions under /usr/pkgsrc/www/bozohttpd. To avoid confusion, we will continue to refer to the program as *bozohttpd*.

Another difference between *bozohttpd* and the other web servers examined here is that bozohttpd will run just as easily through the inetd server as it does a stand-alone daemon. To do this, you must add or modify a line (or two, if you are using IPV6) in the file /etc/inetd.conf as shown in Listing 6. The inetd server in your /etc/defaults/rc.conf file must also be enabled. In a nutshell, what this means is that *bozohttpd* is normally not running unless a web request is received on port 80 (as the default – change this in *inetd.conf* if you want a different port). The tradeoff is that bozohttpd isn't taking up room in the device's memory space if nobody is browsing your device, but the web response is slowed down by the need to load and run *bozohttpd* each time the device receives a request for a web page. You can, however, run it as a background daemon by invoking it on the command line as

```
-bash-3.2$ /usr/libexec/bozohttpd -b -I 8080 /home/hayford/bozohttpd
```

where -I specifies the port number that *bozohttpd* should bind to and the directory at the end is where *bozohttpd* should look for the web pages. The command line invocation illustrates a final difference; *bozohttpd* doesn't require a configuration file, using the command line to control the available settings. Note that port 8080 is used in this example, and not 80 – you can't use port values less than 1024 unless you run *bozohttpd* as root (as inetd does). For the comparison that we are doing here, *bozohttpd* would show very poorly if we use the inetd mechanism rather than running the program as a daemon – it can't, in fact, even keep up with the 5 requests per second that we're using for our test. But, it's a nice option if you know that it will work for your application.

Bozohttpd has a few issues. It appears to spawn a thread for every web request, so even as a daemon it runs perceptibly slower than the others. The data from *top*, however, doesn't show this since most of the work is done in the launched threads and not in the main process.

Consequently, the time recorded for *bozohttpd* is less than it would be if all of the CPU time was credited to the main process and I had to estimate the amount of time spent in the threads (mainly reading the file and sending it over the network).

My estimate of 5.3% is probably a little low, since *boa*, measured at 6.6%, appears to deliver the web page faster than *bozohttpd*; a better timing mechanism than *top* would be needed to measure the time more accurately. With *bozohttpd* as the server, the image display in the web browser would stutter on occasion, though the display was rock steady with the other four servers.

Conclusion

In this article, we've shown how to build, configure, and run five web servers on small embedded processors running NetBSD. All five work pretty well and can serve as the basis of a powerful user interface for your next embedded project. If you are using the web server to control your device, you'll want a server that allows CGI; all of the tested servers do. For embedded applications, memory size is probably the most important consideration. Here, *httpd* (the built-in server for NetBSD, also known as *bozohttpd*) and *boa* (popular with the embedded Linux crowd) are the clear winners. *Boa*, however, appears no longer to be under active development and thus may not be the best choice. *Boa* is also the only server that doesn't support chrooting to another directory, an important security feature for any device that will appear on the Internet instead of a local intranet. If a more powerful application is required, though that would be unusual for an embedded processor, *Nginx* and *Lighttpd* are good choices. Both are powerful and actively maintained and developed, so security enhancements and the addition of new capabilities for these are likely.

NetBSD is already famous for powering the only network-capable toaster in the world. Now, if they would just add a web server ...

References

- Comparison of lightweight web servers, *http://en.wikipedia.org/wiki/Tiny_web_servers*.
- Another good reference on lightweight web servers can be found at *http://www.ibm.com/developerworks/web/library/wa-ltwebserv*.
- Boa web server, *www.boa.org*.
- Thttpd web server, *www.acme.com*.
- Lighttpd web server, *www.lighttpd.net*.
- Nginx web server, *nginx.net*.
- Bozohttpd (httpd), *www.eterna.com.au/bozohttpd*.

About the Author

Don Hayford is a Research Leader at Battelle Memorial Institute, where he specializes in the development of data acquisition systems for customers. Don has been involved with microprocessors from the time they doubled in size from four bits to eight, and once knew how to boot up a PDP-11 using the front panel switches. In his career, he has written software for the CP/M, RT-11, MS-DOS, Apple DOS, Windows, Linux, and BSD operating systems using assembler, Basic, C, C++, C#, and Fortran. Married with three children, Don and his wife like to spend their free time cooking and travelling.

Out-of-the-box sshfs on NetBSD 5.0

Antti Kantee

Sshfs makes it possible to mount a remote directory tree onto the local machine. Only ssh access is required for this.

The mounted directory tree and the files it contains can then be accessed and modified by any application on the system as if the files were local. If only ssh access to a remote machine is possible, sshfs provides a much more practical way to access and manage files than scp'ing them back and forth.

Sshfs is special in that generally use requires no special setup or planning. If ssh access to a remote machine exists, any file system subtree the user has access to can be mounted. In contrast, mounting for example NFS over the internet requires complex access control and VPN tunnels in order to be secure.

Starting from the recently released NetBSD 5.0, sshfs support is present out-of-the-box on major architectures (i386, amd64, sparc64 and macppc). This means that no kernel compilation or installation of 3rd party packages is required for use; ssh access and running the sshfs command manually or via `/etc/fstab` is simply enough.

The *sshfs* implementation on NetBSD has been implemented on top of the NetBSD userspace file systems framework: puffs. The NetBSD implementation is called *puffs sshfs* or *psshfs* for short. For the remainder of this article we use the term *sshfs* to denote the general concept of mounting a file system over *ssh*, and *psshfs* to denote the NetBSD implementation of *sshfs*.

This article goes over the basics of using sshfs on NetBSD 5.0. This is followed by an in-depth discussion about tuning an sshfs mount for maximal performance. Some advanced use-cases are presented. The article ends by presenting some features of psshfs currently under development and appearing after NetBSD 5.0.

Mounting

A file system must be mounted before files on it can be accessed. For psshfs this is done with the `mount_psshfs` command. The syntax is similar to any mount command, and takes two arguments: the file system source and the directory the file system will be mounted to. In its simplest form, a psshfs mount requires running the command `mount_psshfs server.dns.name /mountpoint`. This will do ssh authentication and, if succesful, mount the user's home directory from the host `server.dns.name` under the local path `/mountpoint`. The remote server can be accessed until the directory is unmounted. This is done by running `umount /mountpoint`.

Next we will go over a subset of noteworthy issues in doing a psshfs mount. The full usage of the `mount_psshfs` command is described in the `mount_psshfs(8)` manual page.

Mounting as a regular user

Historically, file systems are mounted as root. It is possible to mount all puffs file systems and therefore psshfs as regular users. Mounting psshfs as a normal user has two implications:

- less daemons on the system run with root privileges
- the ssh keys of the user doing the mount are automatically used

To mount be able to mount as any user on NetBSD 5.0, the following two tasks must be done by root:

- The sysctl knob vfs.generic.usermount must be set to non-zero. This can be done either manually with the sysctl utility after every reboot, or by setting the variable in `/etc/sysctl.conf`.
- The user doing the mount must have read/write access to the file `/dev/putter`. This can be done by running the appropriate `chmod/chgrp/chown` commands. The device

node `/dev/putter` is used only by puffs on NetBSD 5.0, so changing its permissions affect only the ability to mount puffs file systems.

Understanding and tuning caching

To improve performance, psshfs caches results from the server locally. This cached data includes directory contents, file attributes and file contents. If all access to the server is done through the mounted file system, there is no issue with cache coherency. However, if access is done via another route, e.g. with scp, psshfs is unaware of it and the locally cached copy will no longer match the reality on the server. Future access to that data will be satisfied from the stale information in the cache.

The sftp protocol does not support notifications for modified files, so psshfs uses timeouts to address the caching issues. By default, directory contents and the attributes of the files in it are re-read from the server if over 30 seconds have elapsed since the data was cached.

The file content cache works differently. When file attributes are read from the server, the timestamps of the files are compared against previously read values. The cached local contents for a file are invalidated if the timestamps do not match. Therefore, it is possible to have large amounts of data in the local cache for long periods of time if the files do not change on the server.

All the caches implemented by psshfs may be flushed at any time by the kernel if the operating system is undergoing resource shortage. In case accessing data which is expensive to read from the server due to e.g. a slow link, it is worth considering making a local copy.

It is possible to set the timeout value for the attribute cache. This is done with the `-t <timeout>` command line parameter. There are three cases:

- `no timeout`. The attribute cache is always valid. This option is selected by setting timeout to -1.
- `immediate timeout`. The attribute cache is never valid. This option is selected by setting timeout to 0. It should be noted that this will severely degrade performance. Unless there is a strong reason, immediate timeout is not recommended.
- `n second timeout`. Timeout happens after the specified number of seconds. The default is 30.

In all cases, it is possible to force psshfs to invalidate all cached attributes by sending SIGHUP to the psshfs process.

When writing, psshfs always flushes caches immediately, so data will be available on the server as soon as it has been transmitted over the ssh connection. It is therefore possible to copy a file to a psshfs mount and access it on the server as soon as cp exits.

Read-ahead

To speed up sequential reading of files, the kernel performs read-ahead for file contents. This means that if sequential access is detected or suspected, 64kB blocks are read from subsequent offsets before applications actually access the data at those locations. If the read-ahead speculation was correct, in the best case the data will be in the local cache already when the application requests it.

However, since sftp operates over a single TCP connection and commands are processed in-order, a large amount of read-ahead over a slow channel will cause other operations to slow down. For example, if read-ahead requests for 8x64kB of data were just sent over a 512kbps link, interactive commands such as ls would have to wait approximately 10 seconds before all the read-ahead data was transferred and the results for the directory listing would start arriving.

To mitigate the above, psshfs provides the `-r` switch to limit the number of outstanding read-ahead requests. For example, in the above example if read-ahead was set to 2, the wait time would be reduced to 2.5 seconds. The general rule is that the value should be set higher as latency increases and lower as bandwidth decreases and should be tweaked only if problems are noticed. There should be no need for adjustment on a LAN.

Persistent connections

Occasionally, a TCP connection drops. This is usually due to transient network errors. The default action of psshfs is to unmount the file system and make future access to open files in the mountpoint fail. However, in some cases it is desirable to make access block while a reconnect is attempted. This, for example, saves from having to warm the local cache if reconnect is succesful. On the downside, if reconnect is not succesful within a small timeframe, the mountpoint may appear to hang. Notably, in case ssh password authentication is used, the password will have to be re-entered manually each time a reconnect happens.

A persistent mount may be specified with the `-p` flag. The default is non-persistent.

ssh options

It is possible to specify any ssh connection option to a psshfs mount. A useful example might be compression: `-O compression=yes`. All ssh options are specified using the `-O option=value`

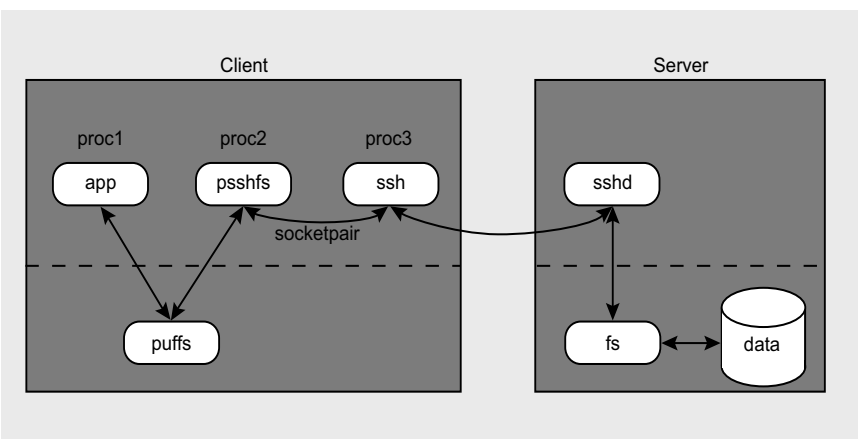


Figure 1. Simplified psshfs architecture

command line parameters. The possible ssh options are documented in the `ssh_config(5)` manual page.

fstab

Just like with any other file system, it is possible to specify frequently used mounts in `/etc/fstab`. The file system type to be used in the third column is `psshfs`. Other than that, standard fstab syntax applies. The first column is the first argument to the mount command, the second column is the mountpoint and the fourth column lists options to be passed to the mount command. Some examples along with their descriptions are presented in Listing 1. The examples have been adapted from the author's laptop. The common theme with all of them is `noauto`, since the laptop might not be connected to a network at boot-time.

Options in the fourth column are as excepted. In fstab the options are specified with `-flag=value`. In the case of an ssh option, *value* is actually `ssh_option=ssh_value`. This leads to the slightly odd-looking values with two equals signs. For example, the combo `-O=port=443` specifies that ssh should use 443 as the port when it connects to the remote server.

Use case examples

Next we go over some examples of how to use sshfs. It is not meant as an exhaustive list, but merely to give some ideas to the reader.

Sshfs is most useful in a single-user dynamic environment, such as a laptop or personal desktop. It is also a good alternative to nfs for single user systems. For example, the author has replaced his nfs mounts on his home system with sshfs. The advantage of using sshfs instead of nfs is that the home server can be mounted from virtually anywhere with a network connection without tunnel setups. Performance especially for the non-LAN case is the same or even slightly better (The performance of psshfs as compared to nfs was measured in the AsiaBSDCon 2008 paper *Send and Receive of File System Protocols: Userspace Approach With puffs*. The paper also goes into detail about the implementation of psshfs and is recommended reading for anyone with more interest on the subject.).

In case reading email the old-fashioned way on a shell server, it is benefitial to mount the home directory of the shell server. This way any attachments can be saved to the shell server home directory and viewed on the desktop machine that was used to ssh to the mail server without any extra copying. However, accessing the mail spool directly over psshfs is not recommended, as mailbox locking has not been verified to work.

A common case for the author is to include source code diffs in sent email. If the mail server is mounted on the workstation, the output of the `vcs diff` command can be redirected onto the mountpoint without a need for extra local copies. After the diff has been attached to the email and sent from another terminal, it can be removed with `rm !$` (on any csh-type shell).

Listing 1. psshfs examples for /etc/fstab

```
# my home directory from the CS lab's shell server
theserver.hut.fi           /m/school        psshfs   rw,noauto

# home directories on ftp.netbsd.org.  Use compression
ftp.netbsd.org:/home  /m/nbftp  psshfs  rw,noauto,-O=compression=yes


# home server from ssh port 443.  No timeouts.
dungeonmaster:/m/dm  /m/dm   psshfs         rw,noauto,-O=port=443,-t=-1
```

Listing 2. Enabling puffs in the NetBSD kernel configuration

```
file-system        PUFFS
pseudo-device      putter
```

Listing 3. Enabling sftp in OpenSSH sshd on NetBSD

```
Subsystem       sftp    /usr/libexec/sftp-server
```

Listing 4. Free space reporting in NetBSD 5.0 and NetBSD-current

```
netbsd-5> df -h /m/dm
Filesystem             Size       Used      Avail %Cap Mounted on
dungeonmaster:/m/dm        0B         0B         0B 100% /m/dm


netbsd-current> df -h /m/dm
Filesystem               Size       Used      Avail %Cap Mounted on
dungeonmaster:/m/dm      456G       337G       120G  73% /m/dm
```

Special case setups

As mentioned in the introduction, the default case requires no prior setup before. However, since it is possible to explicitly disable sshfs support either at the server side or the client side, we will go over what needs to be enabled.

Client

The client side, i.e. the host doing the mount, depends on puffs support. While the puffs kernel driver is enabled by default on major architectures on NetBSD 5.0, it is possible to compile a custom kernel that lacks support, such as one derived from pre-5.0 sources. Enabling puffs on NetBSD 5.0 is a matter of two lines in the kernel configuration file, as illustrated in Listing 2. After this, the kernel is normally recompiled and installed. Early in the NetBSD 5 release cycle it was necessary to add a configuration knob do a complete rebuild of the base

system, but this is no longer required for NetBSD 5.0 – the userspace binaries necessary for psshfs are now installed by default.

Server

Sshfs uses the ssh sftp subprotocol for file access. This is usually on by default and available for use. An easy way to test if an ssh server has sftp enabled is to connect to the remote server using the sftp command. If sftp has been disabled, it can be enabled on the server side in `sshd_config` as illustrated in Listing 3. This requires an intervention from the server admin.

Beyond NetBSD 5.0

While psshfs is fully usable on NetBSD 5.0, there are a couple of interesting features currently available in the development branch of NetBSD and will be making their debut with NetBSD 6.0.

Free disk space

The standard sftp protocol does not provide a method for querying free disk space on the server. However, OpenSSH provides a protocol extension which allows to request this information. The development version of psshfs uses the OpenSSH protocol extension and reports the free space at the remote mountpoint.The difference between the old and new output is illustrated in Listing 4. It should be noted that since sftp does not know about the server's mountpoints, the figures might be valid only at the remote mountpoint and not in any subdirectories, since they might be in a different file system. Also, free disk space reporting works only on OpenSSH servers.

Better interactive performance

As mentioned in the section on read-ahead, heavy data reads can starve out directory operations. On NetBSD-current there is an option to open separate sftp connections for bulk data transfers and directory operations. This means that directory operations are now processed in parallel with reads and writes and do not have to wait for them to complete before being processed. On a low-bandwidth link this can be a significant win and has been shown to speed response time by a factor of 30. Multiple connection mode is enabled with the `-c 2` flag. The default is still one sftp connection per mount.

About the Author

The author is a NetBSD developer and the author of psshfs and puffs.

FreeBSD Security
Event Auditing

Christian Brueffer

Security is increasingly a hot topic in systems administration. Vulnerable systems get patches, firewalls get set up and password policies are enforced. But in the end, all these measures cannot eliminate the risk of a system break-in. They can only reduce it.

Thus, care has to be taken that break-ins do not go unnoticed and the vulnerable point where the break-in occured can be found. This is where the security event auditing system, or short audit, can be of help.

Security Event Auditing

Audit was first introduced in FreeBSD 6.2 and is part of the default kernel configuration since FreeBSD 7.0. Basically, audit is a kernel-based logging mechanism. It records interesting kernel events like program executions and file openings, as well as events like user logins submitted to the kernel by privileged userland programs. They are logged into binary files, so-called audit records, in Sun's BSM file format. These trail files can then be converted to human-readable output using the praudit(1) utility. Among audit's key features are the ability to reliably map events to users, and the fine grained configurability of which events to log.

Configuration

On systems running at least FreeBSD 7.0, audit is supported out of the box. On older systems, the kernel has to be built with *options AUDIT*. Given that, all it takes to enable audit is adding

```
auditd_enable="YES"
```

to `/etc/rc.conf` and rebooting, or alternatively running

```
# /etc/rc.d/auditd start
```

to ensure that the audit daemon is running. It is responsible for managing the trail files and to pass the configuration of the audit system on to the kernel.

The system can be configured through files residing in the `/etc/security` directory. The main configuration file is `audit_control`:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

The most interesting directives in the above default configuration are flags and naflags. They specify which classes of events should be audited for all users (flags), and which of the events that cannot be attributed to a specific user should be audited (naflags). These event classes consist of events that fall into the same broad domain.

For example, events like mounting a file system, setting the hostname and rebooting the system are all part of the class for administrative events, ad. Available event classes are listed in the `etc/security/audit_class` file.

The events these classes consist of are listed in `/etc/security/audit_event`. In the configuration above, the `lo` class that stands for login and logout events (`lo`) is audited in both cases. Here's an example of a successful login event by user chris via SSH from IP address 192.168.1.23:

```
header,94,10,OpenSSH login,0,Thu May 22 08:48:43 2009, +
226 msec
subject,chris,chris,staff,chris,staff,947,947,54084,192.
168.1.23
text,successful login chris
return,success,0
trailer,94
```

However, the subsequent attempt to switch to the root user failed:

```
header,106,10,su(1),0,Thu May 22 08:
48:46 2009, + 66 msec
subject,chris,root,staff,chris,staff,
955,955,54084,192.168.1.23
text,bad su chris to root on /dev/
ttyp0
return,failure : Operation not
permitted,1
trailer,106
```

Let's modify the configuration in `/etc/security/audit_control` to also audit program execution, using the ex event class.

Further, we instruct the audit system to log the command line arguments, `argv`, that are passed on to programs.

```
flags:lo,ex
policy:cnt,argv
```

For these changes to take effect, we need to instruct the system to reload its configuration. This is done by running the `# audit -s` command. It sends a trigger to the audit daemon to synchronize its configuration, close the current trail file and open a new one.

Meanwhile, user chris entered the correct su(1) password and thus switch to the root user. In the following log he uses cat(1) to look at the contents of the `/etc/group` file:

```
header,129,10,execve(2),0,Thu May 22
08:57:26 2009, + 487 msec
exec arg,cat,/etc/group
path,/bin/cat
attribute,555,root,wheel,92,12,1784
subject,chris,root,wheel,root,wheel,1
026,1013,55821,192.168.1.23
return,success
trailer,129
```

This highlights one of the great features of the audit system: It can track a user even after he switches to the root user. Looking at the line beginning with subject, user chris can still be clearly identified.

If, e.g., the root password of a system is known to several people, this can be used to reconstruct the actions of everyone of them.

Also, since the audit system directly logs the invocation of the execve(2) system call, program execution is recorded reliably and cannot be circumvented. If arge is added to the policy options in `audit_control`, even the environmental variable argument to the execve(2) call logged.

Sometimes we only care about specific users, or want different policies for different users. In this case, the `/etc/security/audit_user` file is of help. It lets us specify, which event classes always to audit, and which events never to audit on a per user basis:

```
root:lo,+fr:no
```

This entry tells the audit system to always record logins and logout, as well as successful file reads. The `never to audit` field just contains the invalid class `no`. The `+` modifier in front of an event class means, only log successful actions. Similarly, – means only log failed attempts.

Analyzing Audit Trails

As already mentioned, audit trails are stored in a binary format and can be converted to plain text with praudit(1).

It is important to understand, that audit trails store information about UIDs, GIDs and audit in raw format. They are only converted to human readable form when run through praudit(1), which relies on information contained in the `/etc/passwd` and `/etc/group files`, `/etc/security/audit_event` and others. This also means that care has to be taken, in case UIDs, GIDs or IP address are reused, as it could change the meaning of the audit trails later on.

There is a second important that reads the binary audit trails, auditreduce(1). It takes one or more audit files as input, applies the user-supplied filters and outputs a new binary trail file. This means we usually want to pipe the output though praudit:

```
# auditreduce -a 20090519 -c -ex -u
chris /var/audit/20090604064647.20090
604065434 | praudit
```

This command line prints all of user chris's failed attempts to execute commands after or on May 19, 2009.

The auditreduce command is also useful without the combination with praudit, e.g. for reducing trail files before archiving them.

Managing Audit Trails

Depending on how audit is configured and how busy the system is, the requirements on available disk space and CPU time can be significant. For this reason, the management options for audit trails are important.

All of these options can be modified in the `/etc/security/audit_control` file:

```
dir:/var/audit
flags:lo
minfree:20
naflags:lo
policy:cnt
filesz:0
```

By default, the kernel writes audit trail files to `/var/audit`. This can be influenced with the `dir` directive.

The minimum free space required on the file system this directory resides on is enforced with the `minfree` directive. In the example above, at least 20% of the file system have to be free. If this restriction is violated, the `/etc/security/audit_warn` shell script is executed. By default it only sends a warning message to syslog, but it could easily be modified to reduce and compress trail files and send them to an archive server or burn them on a CD.

The filesz option can be set to set an approximate maximum size for tail files. Once the maximum is exceeded, the audit daemon closes the current trail file and opens a new one.

Conclusion

This article gave a short introduction to security event auditing on FreeBSD, a powerful tool for system monitoring, intrusion detection and post-mortem analysis. Since the FreeBSD audit facility implements the Sun BSM API and file formats, it is even possible to utilize already available third party software written towards these interfaces. When configured properly, the security event auditing can be an important part of a security architecture.

Further Information

FreeBSD handbook chapter: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/audit.html.*

Manpages: audit(1), auditreduce(1), praudit(1), bsm(3), audit(4), auditpipe(4), audit_control(5)

Securing OpenSSH server

Marko Milenovic

OpenSSH is free implementation of SSH suite. Many of us use it on a daily basis and got so used to it we couldn't imagine our lives without it.

There are quite a few people who don't know there were other, not so secure, systems for remote access in the past – `telnet`, `rlogin`, `rsh`, and `rcp`. So, what makes OpenSSH so special? The keyword is security. Unlike suites used in the past, OpenSSH implements very advanced security for data protection. Both data and passwords are encrypted all the time and in a transparent way. The OpenSSH team has worked really hard to make this tool even better. We'll be using the latest OpenSSH version in this text – `5.2/5.2p1`.

Securing the server itself – sshd_config

Depending on which BSD you use your default OpenSSH configuration file may differ. In order to avoid confusion, this text will cover all the configuration tips, and may include some that are already set. Config file that interests us the most is called `sshd_config` and it's usually located at `/etc/ssh/sshd_config` but this may differ. For example, if you use OpenSSH-portable on FreeBSD your config file will be located at `/usr/local/etc/ssh/sshd_config`.

```
Port 9999
Protocol 2
LoginGraceTime 1m
PermitRootLogin no
StrictModes yes
MaxAuthTries 4
MaxSessions 5
AllowUsers johndoe
PermitEmptyPasswords no
```

Let's see what has been achieved by the changes made above. First of all listening port has been changed from `22` to `9999`. A few people will argue that there is no real point in changing this since a skilled attacker may conduct a port scan and find our SSH service running on port `9999`. While true, why not make things more complicated for our attacker? On the other hand changing this port will prevent some brute force attempts done by certain scripts which go for port 22 by default.

The second line ensures that only SSHv2 is used. SSHv1 doesn't offer all the security of SSHv2 thus there is no reason to use it.

By default server will drop the connection if the user hasn't logged in within 2 minutes. `LoginGraceTime` tells our server for howlong to wait. Putting 0 here makes no time limit.

We definitely don't want our root user to be able to login via SSH. Though OpenBSD allows root user login by default that is just so that we may access newly installed system over the network, add limited user account and then change this. Keep this in mind.

Putting StrictModes to *yes* tells out SSH server to check file modes and ownership of the user's files and home directory before accepting login. This may be useful because new and inexperienced users may leave their data world writable.

The next line tells our server after how many failed login attempts to drop the connection. Once the number of failures reaches half this value, additional failures are logged. In our case user may try and login twice, after that his attempts will be logged and after the fourth attempt the connection will be broken.

MaxSessions tells our server how many sessions user may open per network connection.

We now state the exact list of users that may login via SSH. You may also use *AllowGroups* if you have a lot of users using SSH. If that number is low using AllowUsers is just fine.

The last line simply disallows paswordless accounts to access the system.

Our SSH server is now even more secure. Yet, this is not the end. Let's take a look at a few more options we want to change.

```
X11Forwarding no
UsePrivilegeSeparation yes
VersionAddendum
```

X11Forwarding allows our users to run X11 applications over SSH connection. If there is no good reason this option should be set to *no*.

UsePrivilegeSeparation tells sshd server to create an unprivileged child process to deal with incoming network traffic. After the user has successfuly authenticated another proces is created that has the privilege of the authenticated user. The sole purpose of this option is to prevent privilege escalation by containing any corruption within the unprivileged processes.

And finally, let's not say to the world which OS are we running our SSH server on. For example, by default *VersionAddendum* on FreeBSD says something like *FreeBSD-20080901*. Let's just leave this empty.

Changing configuration of our SSH server to the values above will make it more secure but this is not enough.

Changing options per user

Let's asume you might want to change some options on per user basis. Let some users have some of the options turned on or off. To override setting on per user basis use the following:

```
Match User johndoe
  X11Forwarding yes
```

This will allow using X11 for user johndoe. Note that some of the options can't be overridden here.

Using keys instead of passwords

One should try and avoid using system passwords to login whenever possible. Thou OpenSSH sends all the data through an encrypted channel using system passwords may be dangeorus since anyone who gets them may access the system. That is why there is a much better solution – using SSH keys.

When we edited our sshd_config file we ommited three lines which you most probably noticed:

```
#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile     .ssh/
authorized_keys
```

Since we do want to use key based access do uncomment these lines. Our second step will be to generate the keys on our local machine. This may be done by doing the following: see Listing 1 and 2.

Figure 1. OpenBSD logo

Listing 1. Generating RSA key

```
# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/johndoe/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/johndoe/.ssh/id_rsa.
Your public key has been saved in /home/johndoe/.ssh/id_rsa.pub.
The key fingerprint is:
92:1c:a3:14:d2:ff:ea:3b:fe:47:f0:f7:42:6d:20:e2 johndoe@localhost
```

Listing 2. The key's randomart image

```
+--[ RSA 2048]----+
|  ...            |
|   ...           |
|    ..o          |
|   . o.+o . .    |
|    . +oS+ . o   |
|       .E o o o  |
|       . . o o   |
|      o   . . .  |
|     o++..   .   |
+-----------------+
```

Listing 3. Chrooting CSH

```
# cp /bin/csh /chroot/bin/
# ldd /bin/csh
/bin/csh:
        libncurses.so.7 => /lib/libncurses.so.7 (0x80067d000)
        libcrypt.so.4 => /lib/libcrypt.so.4 (0x8007c9000)
        libc.so.7 => /lib/libc.so.7 (0x8008e2000)
```

We have now generated our RSA keys which we'll be using for accessing our remote SSH server. Notice the part that says *Enter passphrase (empty for no passphrase)*. You may use keys without a password but one should be used for a better security.

Our next step is to copy file called id_rsa.pub to our remote server. This may be done either by using SCP or SFTP. Once there just move it to your *.ssh* directory on the server and issue the following commands:

```
# cat id_rsa.pub >> authorized_keys
# chmod 600 authorized_keys
```

It is very important to remember to configure *chmod* part. If this file has any other access rights our key based authentication won't work. It is time to test our settings.

```
# ssh -p 9999 johndoe@remoteserver
Enter passphrase for key '/home/
johndoe/.ssh/id_rsa':
```

It works. If you wish you may disable all other ways of authentication:

- PasswordAuthentication no
- UsePAM no

This will make sure that without a key no one will be able to access our remote server.

Chrooted environment

Wouldn't it be nice if we could *lock* our users to certain directory so that they won't be able to sniff around our system? That is called chrooting. First of all we would need to create chrooted environment for our user. Note that once the user logs in he will be kept in an enclosed surroundings thus without any commands and programmes we don't give him. Let's say our chrooted directory is `/chroot/`. When user logs in he will be dropped out since that directory is empty – no shell or any programme at all. For him there is nothing beyond this `/chroot/` directory. This means we would need to give him some set of commands so that he may login and work. Let's assume that our user will be using CSH as his shell. We find out that CSH is located at `/bin/csh`. This means we must create `/chroot/bin/` and copy csh executable into it. Once we do that we need to check what libraries are needed so that csh actually works: see Listing 3.

So, we also need three libraries copied from /lib into `/chroot/lib/`. Once we do this our user will be able to login and... do nothing. Giving him just csh is not enough. He will need something more in order to work normally. Which system commands will you provide him depends only on you. It is usually normal to copy: `ls`, `cd`, `mv`, `cp`... and similar to `/chroot/`. Make sure you check each and every of them using LDD and to copy all needed libraries into chroot directory. You will also need to create / chroot/home/username as his new home directory. User will be able to work but won't be able to access our system. When he issues something like:

```
# cd /
```

he will be dropped into `/chroot/` dir meaning no access to real root directory of our system.

The last thing we should do is modify our SFTP instance. The original line holds:

```
Subsystem sftp /usr/local/libexec/
sftp-server
```

We want this changed to:

```
Subsystem sftp internal-sftp
```

Conclusion

The OpenSSH team is doing a really great job in making sure our favorite remote access system works good and stays secure. With just a little good practice on our side and by using the tools development team created for us we may tighten this already secure system. One may argue that some of the tips given in this text are a bit too much. When it comes to security little paranoia nevers, right?

So, is all this enough to feel secure? Yes, in most cases you should feel rather secure. You should keep in mind a few facts. Though OpenSSH is developed by people devoted to security it is still just a software. A sofware has bugs and issues. In order to really be secure it is suggested you keep an eye on the development of OpenSSH and do regular updates. The development team is doing a great job keeping OpenSSH up to date and fixing all the issues and security problems as soon as possible. You may keep an eye on this page: *http://www.openssh.com/security.html* It is a list of all know security issues within all versions of OpenSSH. This way you'll know when is the time to either look for a patch or a new version of OpenSSH. Always keep in mind that your server will be as secure as you are careful and responsible about the things you do on it.

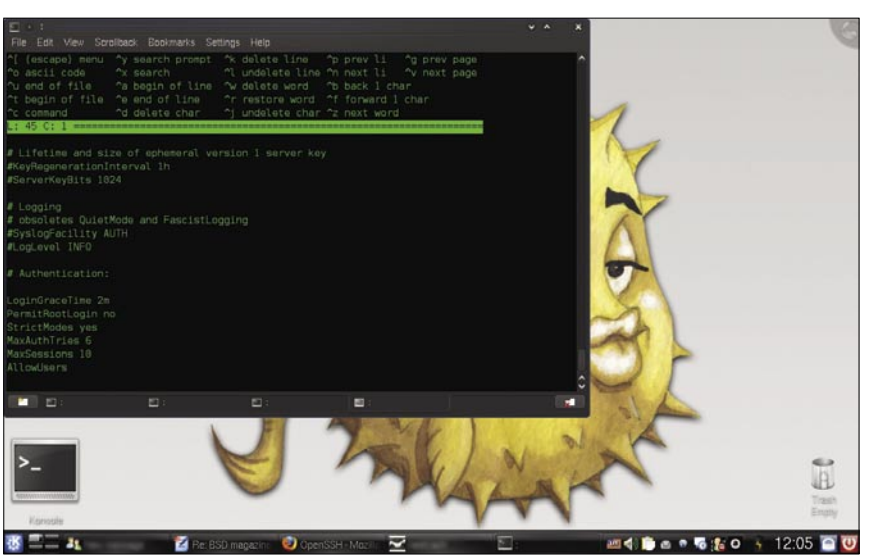


Figure 2. Modifying your sshd_conf file

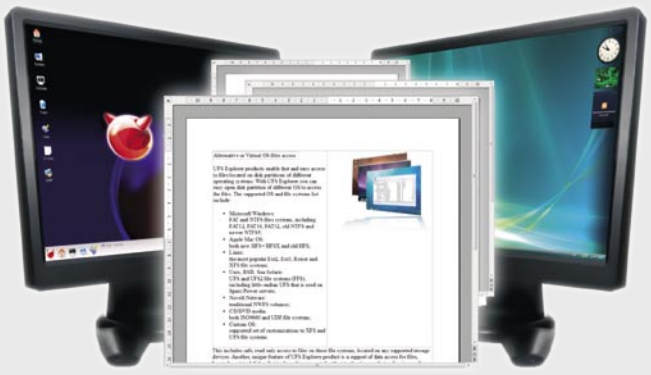
About the Author

Marko Milenovic works as a security consultant. He is coordinator of BSD Serbia – Serbian BSD community and Core team member of Free Software Foundation Europe. He works and lives in Belgrade. You may find his regular security rants at *http://www.sekuritatea.com/*

Increase your work efficiency on BSD systems with UFS Explorer

UFS Explorer

Imagine the scene. You have a PC running BSD-based system and Windows. Files access on BSD certainly takes you few seconds and clicks. Switching between BSD and Windows does not trouble you any more. Finally, you easily copy necessary files from the former OS to the latter one.



Or, alternatively, you have installed BSD derivate on your virtual machine to make experiments with new software. Each time whenever you need to access BSD file you do not launch or configure your virtual machine. Totally enjoying speed you gain access to any interested file on it within a moment. It is fantastic how quickly and easy you drag required files from the virtual machine running BSD to host Windows!

Does that sound unbelievable? Well, not with UFS Explorer.

The lion's share of the top qualitative desktop software available in the world computer market is developed predominantly for Windows application. As a result, users who already have BSD operating system are forced to install Windows as well. Yet, this neighborhood appears problematic. Continuous switching from BSD to Windows troubles data access. Moreover, it is impossible simply to copy files from BSD to Windows.

Additionally, many Windows users who want to change the operating system to BSD first install it on a virtual machine. It is accounted for their intention to test BSD avoiding any modification in a computer configuration. However, users face the need to start and configure a virtual machine so as to gain access to important BSD files. It is inevitable unless they have specialized software.

For UFS Explorer it is a usual practice to access necessary data on neighboring BSD-based systems along with files exchange from BSD to Windows. Therefore, it is an excellent alternative to Windows drivers when it goes about most winning access solution.

Besides, for UFS Explorer products it is a matter of few seconds and clicks to access necessary data on neighboring virtual BSD-based systems. Furthermore, it is possible to drag files from a virtual machine running BSD to Windows desktop.

UFS Explorer can serve as a helping hand for any corporate user as well. No matter where BSD is used (workstations or servers), with UFS Explorer one can gain fast access to necessary files on a neighboring BSD and any BSD formatted external storage medium.

However, the maximum benefit using UFS Explorer is achieved when working with back up copies on a virtual server. The software provides system administrators with a direct access to required information on virtual BSD servers. They can easily retrieve important data from virtual server backup files without launch and configuration of a virtual machine as well as reconfiguration of a guest operating system so as to copy BSD files to a target location.

In comparison with poor data access a loss of data is a more serious problem. It has particularly grave consequences for companies as they take risks of failure in the event of precious data loss. BSD data recovery is one of the major tasks accomplished by UFS Explorer. With a profound efficiency it enables disc data recovery on both workstations and servers running BSD. It yields highest quality results in case of data loss due to logical failures such as formating, software malfunction or accidental deletion and hardware failures in a RAID system with sufficient redundancy. UFS Explorer can be successfully applied to virtual machines for recovery goals as well.

For further information about the wide range of UFS Explorer features, please visit *www.ufsexplorer.com.*

About the developer of UFS Explorer

Utilizing the industry's finest Data Access&Recovery tools and technologies, LLC SysDev Laboratories works out qualitative software solutions for all the variety of operating systems, such as Windows, Linux, BSD, Mac OS, Novell. A rich experience in data recovery tasks allows the company to create reliable and highly efficient products which are able to crack crucial data loss problems within the shortest possible time.

Staying Secure using PC-BSD

James T. Nixon III

"Help! Pop-ups are destroying my computer!" I cannot count how many times I have heard those words come from my brother's mouth.

My first response is usually something like, *what have you downloaded lately?* to which he usually responds, *NOTHING*!... Of course. Well, he may not understand that something as trivial as a free screensaver can include malicious code turning his computer into a pop-up ad hell. I have warned him countless times about *bad sites*, but he can't tell good from bad, which really shouldn't matter. A user should be able to browse the net without the constant paranoia of being exploited. His excuse is usually something like, *the pop-up told me I had spyware, so I just installed this one program to help*. Aha! So you did install something. Virus and SpyWare removal tools only further condition the user into believing that this is the only safe way to compute, and that getting infected is normal, like the common cold.

There is the problem. Some users do not understand the difference between a pop-up ad and an actual Windows warning message. This may not apply to the people reading this magazine, but I am sure we all have a relative or two with similar issues. PC-BSD is geared towards these users, and aims to be an easy and secure solution to desktop computing. PC-BSD comes secure by default, using denyhosts to block all brute force SSH attacks, pf to filter packets based on the rules defined in `/etc/pf.conf`, and of course by requiring administrator rights to install software. The system is also kept secure by keeping PC-BSD up-to-date by issuing security updates with th System Updater utility. This utility notifies the user if an update is issued so that the user may double-click the Notifier icon in the system tray and then choose which updates to download and apply.

I have included the default `/etc/pf.conf` with comments to briefly explain what PC-BSD is doing by default. Of course you may ditch these rules for a set of your own, but at least you know your child's computer is safe out of the box (Listing 1).

As shown above, PC-BSD is pretty secure by default. In fact, you cannot SSH into your PC-BSD box unless you add an exception to the firewall. To add an exception to the firewall, go to *menu›System Settings›Firewall*, then click 'Run in Administrator Mode' and enter the administrator/root password. This should bring up a window titled *Firewall – KDE Control Module*. You may define whether the firewall should run at startup. You can also stop, start, or restart the firewall by clicking the corresponding buttons. To allow remote ssh into the box, click the Exceptions tab and then click the *Add Entry* button on the bottom left. From here you can select SSH from the drop-down menu, which should default to port 22. Make sure *Direction* is set to incoming, *Protocol* is set to TCP, and that you have the correct interface selected. Now click *Ok* and you should be able to SSH into your secure PC-BSD box. What did this do exactly? Well, let's take a look at /etc/pf.conf and find out.

```
# from /etc/pf.conf
pass in on em0 proto tcp from any to (em0) port 22 keep
state
```

So the Firewall tool in PC-BSD is a simple front-end to add rules to `/etc/pf.conf`. Good to know! In the above rule you can see how it is set to allow TCP connections from anyone to port 22. This is what makes running PC-BSD great. You get all the power, stability, and security of FreeBSD, without the hassle of teaching your mother about manpages. This is why I suggest PC-BSD to my entire family (and then quickly turn off my cell phone...). I hope this brief overview on how PC-BSD keeps itself safe from script-kiddies and Windows targeted malware brings more people to PC-BSD. Remember, submit those bug reports!

Listing 1. /etc/pf.conf

```
# from /etc/pf.conf
# skip all packets on localhost
set block-policy return

# drops packets with inconsistencies
scrub in all


# setting the policy to be default deny and log
anything that isn't accepted
block in log



# only allow packets on the interface from its network
antispoof quick for lo0 inet


# block non-routable IP addresses
block in from no-route to any


# everything outband is allowed
pass out keep state

        # everything in this table is loaded when pf
loads
        table <blacklist> persist file "/etc/
blacklist"


# Allow icmp out from anybody to anywhere
pass inet proto icmp from any to any


# Allowing anybody to connect to your machine with
either protocol from the specified port range
pass in proto {tcp,udp} from any to any port 49152:
65535 keep state


# block everyone in the table, could be IP addresses
or IP ranges, etc..
block from <blacklist> to any


# the following blocks can be written many ways. Here
we have each interface split up to pass  the specified
UDP/TCP ports.


# Filtering rules for the em0 interface
pass in on em0 proto udp from any to (em0) port 137
keep state
pass in on em0 proto udp from any to (em0) port 138
keep state
pass in on em0 proto tcp from any to (em0) port 445 keep
state
pass in on em0 proto tcp from any to (em0) port 137 keep
state
pass in on em0 proto tcp from any to (em0) port 139 keep
state


# Filtering rules for the dc0 interface
pass in on dc0 proto udp from any to (dc0) port 137 keep
state
pass in on dc0 proto udp from any to (dc0) port 138 keep
state
pass in on dc0 proto tcp from any to (dc0) port 445 keep
state
pass in on dc0 proto tcp from any to (dc0) port 137 keep
state
pass in on dc0 proto tcp from any to (dc0) port 139 keep
state


# Filtering rules for the fwe0 interface
pass in on fwe0 proto udp from any to (fwe0) port 137
keep state
pass in on fwe0 proto udp from any to (fwe0) port 138
keep state
pass in on fwe0 proto tcp from any to (fwe0) port 445
keep state
pass in on fwe0 proto tcp from any to (fwe0) port 137
keep state
pass in on fwe0 proto tcp from any to (fwe0) port 139
keep state


# Filtering rules for the fwip0 interface
pass in on fwip0 proto udp from any to (fwip0) port 137
keep state
pass in on fwip0 proto udp from any to (fwip0) port 138
keep state
pass in on fwip0 proto tcp from any to (fwip0) port 445
keep state
pass in on fwip0 proto tcp from any to (fwip0) port 137
keep state
pass in on fwip0 proto tcp from any to (fwip0) port 139
keep state


# Here is a simple example of a similar configuration,
but with less opportunity for commenting.
pass in on { em0, dc0, fwe0, fwip0 } proto { tcp, udp }
from any to  { em0, dc0, fwe0, fwip0 }  port { 137, 138,
445, 137, 139 } keep state
```

Stop Hackers
With Protection Script

Svetoslav P. Chukov

I suppose you have a border server that is freely accessible from the internet or you just want to have a secure machine. Whichever the case is, I will tell you my story.

I have a server that offers some services on the internet and I was happy to have it up and running for long time. But, one day, the server started crashing without a reason. Actually, there was a reason, and the reason was that the server was under attack. I saw many lines with messages in my logs that reported for a login attempts from different locations around the world. That was the end of my good time and the beginning of the fight against the hackers.

A server that is accessible from the internet is hard to be secure properly. Many people prefer to put the machine behind a router and firewall but even in this case there are many vulnerabilities and security holes that are available for exploitation by hackers. Let's take the real example of my server. The server offers SSH as a remote access service and you can imagine how many attempts for SSH login I had every day... Many attempts...

The accent of this article will be about using simple methods to protect your machine. One of those methods is just our console shell – SHELL. Probably you know that SHELL is just a shell. It can not improve the security or protect the machine from hack attempts, right? Or probably not, SHELL can be used to monitor certain events in the system and trigger actions depending on the event's results.

Actually, when I say SHELL I mean that this will be done via a SHELL script. SHELL can not do anything just by itself, you should instruct it what to do.

I have the following objectives in this article. To show you:

- A typical login attempt coming from a suspicious IP address.
- How to protect yourself from such an intrusion.
- Explanation of the used method.
- The result.

A typical login attempt coming from a suspicious IP address.

- Feb 10 18:21:37 view `sshd[20851]: input_userauth_request:` invalid user root
- Feb 10 18:21:37 view `unix_chkpwd[20852]`: password check failed for user (root)
- Feb 10 18:21:37 view `sshd[20850]: pam_unix(sshd:auth)`: authentication failure; `logname= uid=0 euid=0 tty=ssh ruser= rhost=xxx.xxx.xxx.xxx user=root`
- Feb 10 18:21:39 view `sshd[20851]`: Connection closed by xxx.xxx.xxx.xxx
- Feb 10 18:21:39 view `sshd[20850]`: Failed password for invalid user root from xxx.xxx.xxx.xxx port 55835 ssh2

If you have similar lines in your log files, then be sure that someone is trying to login to your machine without your permission. Most of the hackers regularly scan large pools of internet hosts for possible vulnerabilities and weaknesses. An open and accessible SSH is a weakness that should be solved if you wish your machine to survive on the internet. In this article you have the opportunity to see a real live example. From this example you can gain much knowledge and you can use it to protect your systems.

The following example is about the SSH but actually any server with login access is vulnerable for this kind of an attack. It is hard to classify it as an attack, because it is just login from an unknown host somewhere in the world. This is the most simplest case of an intrusion attempt but if successful this attempt is enough to compromise your server and security hole.

How to protect yourself from such intrusion

Here is the place where SHELL comes in help. For my security I created a SHELL script that handles those kind of

Listing 1a. Protection-script.sh

```
    This script will help you to stop and detect failed logins via SSH. It can be slightly modified to support many
other types of logins – FTP, and others.

    Copyright (C) May 4, 2009  Svetoslav Chukov <svetoslav.chukov@gmail.com>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

#!/bin/sh

allowed_ips="192.168.0.2"

while read line
do
        list='echo $line | grep "invalid user" | awk '//{print $13}''
        echo $line >> /var/log/auth0


        for i in $list
        do

                is_exists='cat /etc/pf.conf | grep $i | grep block'

                for a in $allowed_ips
                do
                        if expr "$a" : "\(.*$i.*\)";
                        then
                                echo "$i is allowed to connect to us. We forgive it!"
                                echo "$i is allowed to connect to us. We forgive it!" >> /var/log/auth1
                                   break 2;
                        fi
                done


                 length='${#is_exists}'
                 echo $length
                if expr "$length" "<" "1";
                then
                        echo "$i just crossed the line. This source IP has been blocked!"
                        echo "$i just crossed the line. This source IP has been blocked!" >> /var/log/auth2
                          #echo "block in from $i" >> /etc/pf.conf
                          echo "ipfw -q add deny all from $i to any" >> /etc/ipfw.rules
                          ipfw -q add deny all from $i to any
                 fi
        done
done < /var/log/auth.pipe
```

Listing 1b. Protection-script.sh

```
Please add or modify this line in your /etc/syslog.conf

auth.info;authpriv.info                                          /var/log/auth.pipe

This will instruct syslog daemon to report data in the named pipe. This pipe will be our bridge to   protection-
.sh script.

Then you have to create the named pipe with this command:
mkfifo /var/log/auth.pipe
```

Listing 2. Log files

```
/var/log/auth0

Jun 16 20:15:30 pcbsd sshd[29933]: Invalid user cesar from 192.168.1.6
Jun 16 20:15:33 pcbsd sshd[29933]: error: PAM: authentication error for illegal user cesar from freebsd1
Jun 16 20:15:33 pcbsd sshd[29933]: Failed keyboard-interactive/pam for invalid user cesar from 192.168.1.6 port
60758 ssh2

Jun 16 20:15:37 pcbsd sshd[30014]: Invalid user merlin from 192.168.1.7
Jun 16 20:15:48 pcbsd sshd[30014]: error: PAM: authentication error for illegal user merlin from freebsd2
Jun 16 20:15:48 pcbsd sshd[30014]: Failed keyboard-interactive/pam for invalid user merlin from 192.168.1.7 port
62605 ssh2

/var/log/auth2

192.168.1.6 just crossed the line. This source IP has been blocked!
192.168.1.7 just crossed the line. This source IP has been blocked!
```

issues. There are many options you can use, but basically the most useful way is to cut all connections to your machine immediately after an intrusion is detected. That is the best you can do and it is what I do. To protect my systems I need fast and reliable method to recognize such attacks, fast response, detailed log and report.

How can we recognize an attack? How can we recognize it, since the communication client <-> server is encrypted via a secure shell? Then, in this case we obviously can not use any of the known Intrusion Detection Systems. And also, we are not that advanced to have such a knowledge to use an IDS. An IDS also can not recognize a login attempt and can not work in our case.

My solution to solve this problem is simple and elegant. I based the entire protection of my server on a SHELL script. Let's explain the way how SHELL did the work in my case.

An explanation of the used method

When the machine registered a failed login attempt, a message for the issue is logged in the log file. So, this is our *key factor* to recognize and stop the attack.

The SSH daemon prompts usually 3 times for username and password and after each of the failed logins it logs a message. To protect my system on-the-fly I created a script that handles this messages via a named pipe directly from the syslog daemon. Then, this script just takes some actions and inserts an firewall rule.

That results in a direct drop of all the packets from this source IP to my machine. After the first failed login, the intruder is dropped and any further attempts to access my server are impossible. Simple and elegant!!!

Without using any complicated systems and methods, with this simple SHELL script I improved the security of my server. I hope it will be useful for you! Please, have a look at the following script source code here: see Listing 1.

The result

For almost 2 months while this script is working on my server it blocked more than 110 IP addresses from accessing my computer. So, you decide how effective it is!

There are many ways to build high security but it is interesting to know the best way, the easiest, and the most effective one.

Let's have a look at the following situation. One has a server and wants the maximum security for that server. Actually, there is no guarantee that this server will not be hacked in the future, everything is possible. So, one has to achieve 3 basic crucial points:

- As more effective as possible intrusion detection.

- Fast response to the attack.
- Make sure the hacker will not come again.

Now, let's have a look at our script. It detects attempts of logins with invalid user names. Immediately after the attempts are made, the script will make an action. It will add a drop rule to the firewall and it will also log the issue. With those actions the attacker will be banned any further attempts to reach our machine from this source IP will be impossible.

The defensive action will be triggered immediately after the first wrong login. Probably it is too drastic but the defense of the machine should be strong and it is better to stop the attacks in the beginning instead to recover broken systems later.

Let's see now the real results in the log files. There are 2 log files that will indicate an attack has been stopped. `/var/log/auth0` contains just a copy of authpriv.* log messages. More interesting is `/var/log/auth2`, it contains information about dropped IP addresses. Let's have a look: see Listing 2.

Voila! Any further attempts for our machine to be reached from this source IP are impossible.

So, we have 3 types of logs. A general log file with all the information about the authentication issues. A log file with the information about the banned IPs and a log file with the information about what IP is granted to reach our machine.

There are a lot of features that can be included in this script. It is a very simple and easy to use script but it is also very powerful. It protects the machine on-the-fly, without the need to scan log files or regularly update the firewall rules. It works right after the issue is reported to syslog. The attacker will have only 1 chance to connect to us. This will protect us from future attacks from this IP and also will prevent other types of attacks against us.

I called this script *Protection Script* because of the work it does. So, the name of the script is *Protection Script* but also it's work is to be a protection script. Protection Script can be modified to handle any type of a service. It is not intended to be only a protection for SSH. It can handle also FTP, SFTP or any other service that exists. To extend it to handle FTP, one has to modify just the matching parameter for the log.

That means one has to add support to match against this line of log:

```
Jun 16 20:15:48 pcbsd sshd[30014]:
Failed keyboard-interactive/pam for
invalid user merlin from 192.168.1.7
port 62605 ssh2
```

This line is log for an unsuccessful login attempt for SSH. If one wants to add support for FTP in Protection Script, he/she should match the criteria of detection against the same log line for FTP.

It can also be done for FTPS, telnet, and etc...

About the Author

Svetoslav Chukov is programmer and administrator of BSD and Linux systems. The interesting thing about him is that he is also an experimenter with BSD. He does interesting and crazy things that the other people don't even think about. Hi is very big fan of NetBSD and OpenBSD but actually he prefers NetBSD. He is also a big supporter of PC-BSD, and even one of his servers is running PC-BSD.

great subscription offer

SAVE $20!

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

Why subscribe?

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

3 EASY WAYS to subscribe:

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

BSD magazine ORDER FORM

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

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

Order information
(□ individual user/ □ company)

Title
Name and surname
address

postcode
tel no.
email
Date

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

Payment details:

□ USA $39.99
□ Europe 29.99€
□ World 39.99€

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

Contact Information:

Software Media LLC
1521 Concord Pike, Suite 301
Brandywine Executive Center
Wilmington, DE 19803
USA

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

OpenBSD
on the Sharp Zaurus

Michael Hernandez

If you look at the OpenBSD Platforms page (http://openbsd.org/plat.html) you'll see Zaurus down at the bottom. Is it there just so that the OpenBSD team can safely say that they support a multitude of platforms, from Alpha to Zaurus?

That may be, but the Zaurus isn't something the OpenBSD folks made up it's a brand of PDA created by Sharp, and you may be surprised to find that it could prove to be more than a novelty item in your pile of electronics.

Sharp released the SLC3000 in 2004 which had 416MHz Intel XScale CPU 4GB hard disk and USB Host support (prior to the SLC3000 you could use the Zaurus as a USB device but not plug USB devices into it), along with 64MB of RAM and 16MB Flash ROM. It's 4.5" x 3.25" when closed, and when opened it's 3.5" tall. The SLC3100 is the same except that it has 128MB Flash ROM. In 2006 Sharp Released the latest (and possibly last) version of the clamshell Zaurus, the SLC3200, which boasts a 6GB hard disk, but otherwise is the same as its predecessors. OpenBSD runs on all 3 of the SLC3x00 models.

I first took note of the Zaurus in around 2003 when a friend showed me his SLC860. It was one of the first clamshell models, and I was excited just to see a BASH prompt on a little handheld device. It was so exciting to see what basically was a baby laptop running Linux. I bought one almost immediately after seeing his and enjoyed it so much that I bought an SLC3000 shortly after it was released. To get my Zaurus I had to find an online reseller because the Zaurus clamshell models were (and still are) only sold in Japan.

When I got my Zaurus there were not too many OS options available. Basically, you could run the default Linux installation (which had PDA features and not much more) or you could run a custom ROM like pdaXrom (*http://www.pdaxrom.org/*) or OpenZaurus (now ngström, *http://www.angstromdistribution.org/*). Using the custom distributions was fun, but something didn't feel quite right... Even though I had much more freedom with a custom ROM than with the default Linux installation, I still felt somehow limited. I wanted my Zaurus to be more like the little laptop that it appeared to be, not just an overgrown PDA. Around the time I bought my Zaurus, I had also started experimenting with FreeBSD on my desktop and had also been using OpenBSD as a firewall OS at my office. When I learned that OpenBSD was released for the Zaurus I was overjoyed – unlike Linux, which can vary greatly from one distribution to another, I knew the OpenBSD would remain pretty much the same regardless of the platform I ran it on. What could the consistency and security of OpenBSD be like on my Zaurus? As far as I was concerned, it could only be good.

The installation of OpenBSD on the Zaurus is fairly straightforward. I found though, that to use the full disk for OpenBSD, some tricks were needed. Luckily I was able to find help on the OESF Forums, in a thread started by a a forum member named iamasmith (*http://www.oesf.org/forum/index.php?showtopic=17213*). In this thread he describes the process by which you can safely install OpenBSD on the entire disk of your Zaurus. It's not very difficult,

Once you have OpenBSD installed on your Zaurus, you basically can do anything that you could with a computer, only smaller. The operating system isn't stripped down or limited in any way. There are fewer packages for the Zaurus than for some other platforms, but you'll find that the package list is extensive, with over 4100 prebuilt packages available to be installed. The potential of this system doesn't really need explanation – there's an OpenBSD installation in your pocket! I think even Duncan Steele (the main character of Z4CK, a novel by Kevin Milne, which Kevin wrote on his Zaurus) would agree that OpenBSD on the Zaurus opens up a world of almost limitless possibilities. I use it as a mini web server with a full complement of PHP 5 extensions. I think it's great to be able to quickly test a code idea, or

read a manpage on the go. I keep a subversion checkout of projects I'm working on with me, so that if I get a flash of insight while I'm on the subway, I can try it out in moments, instead of waiting until I get home. Then, once I'm home, or at any wifihotspot, I can check my code back into the repository. A crowded subway is no place for a laptop, or even a netbook. The small size of the Zaurus makes it ideal – it's just large enough for me to read and type comfortably, but smaller than an average paperback novel. The Zaurus also makes a decent, although somewhat bulky, media player. If you think about it, I'm sure you could find very interesting ways to make use of the tiny OpenBSD machine.

However, I'm not advocating that you run out to get a Zaurus as soon as you finish this article. While a very useful tool (especially running OpenBSD) there are some caveats, some of which might be dealbreakers for you. First of all, the Zaurus is not a cheap device. The SLC3200 can cost nearly 600 Euro. That's almost $900 US, and that's not including tax and shipping. For the same money you can buy a 17" Dell laptop with 4GB of RAM, 320GB Hard drive, etc. In this time of a troubled economy, the Zaurus is definitely a splurge. The fact that you can't buy a Zaurus in a store outside of Japan is also concerning. If something goes wrong, you can't simply walk into the store you bought it from to ask for help. As the wikipedia article about the Zaurus states (*http://en.wikipedia.org/wiki/Sharp_Zaurus*):

Since there is no official export channel from Japan, Sharp offers no warranty or repair service outside Japan, so foreign buyers are dependent on their chosen reseller to handle repairs, usually by sending to their agent in Japan who acts as if the device was owned and used in Japan in order to have it repaired by Sharp, before sending it back to the owner. Whilst Zauruses are actually quite robust devices, due to their miniaturization they are not easily repairable by casual electronics hobbyists.

I suspect that the price and lack of easily accessible support might be enough of a deterrent for most readers. Unfortunately, the caveats don't end there. There are some issues related directly to the OpenBSD operating system for which you should watch out. What I have always found most frustrating about running OpenBSD, is that at times device support can be lacking, in terms of easily acquired products, anyway. This is not a fault of the operating system, rather it is the open philosophy to which the project strictly adheres that keeps some devices incompatible. Most often, it is the fault of the vendor, who refuses to provide documentation or release code that can be used under an open license. The real frustration for me comes when a device such as the Linksys USB200M is listed as supported, but I could only get the USB200M version 1 to work. No matter what I tried, I could not get a version 2 or 2.1 to work on any OpenBSD machine, my Zaurus included. The USB200M is among the easiest devices to find, they are sold at most stores that sell computers and accessories. It's always been frustrating that the most easily accessible devices are the ones least likely to work. Again, this is not fault of the operating system developers, it is due to a change in chipset that Linksys (Cisco) used in the device. It also doesn't affect only OpenBSD. In some cases devices won't work with any OS other than MS Windows. It's something that I got used to over the years I've been in the *nix world, but I never liked it. I wish all hardware vendors would

Another issue related to device support is that the Zaurus does not provide much power out of its USB port. This means that in order to run some devices that expect to get all of their power from the USB port, you need to have a powered USB hub. Plugging in a hub might not seem so bad, until you need to go and you're tethered to the electrical outlet. I found that using a monitor that had a built in USB hub was convenient, although unfortunately you cannot use the monitor's display capabilities with the Zaurus.

If you've ever tried to do a lot with a 400Mhz CPU and 64MB of RAM, you know how that experience can be. It's really not much in terms of resources, so when you look at the list of available packages, you should realize that some of them are just impractical for running on a Zaurus. Firefox, for example, will run, however after the base OS takes its RAM, and X and your window manager take their RAM, Firefox doesn't have much to work with, and it shows. It does work though, if you have the patience. As a matter of fact, I hardly use X on the Zaurus because the small amount of RAM makes it a bit sluggish and not very fun after a while. I'm a user that spends most of my time in a terminal anyway, even when I'm using OS X. I find that at the console I can get more accomplished with less resources. With that philosophy, I hardly start X at all, and get most of my work done via the command line, locally on the Zaurus keyboard or via SSH.

Lastly, you may think that a Zaurus running a full Unix installation might be a great PDA, but I find that its ability to be a PDA has almost been hampered. The large PIM style applications that run in X take far too many resources to be useful. On the command line you could attempt to duplicate PDA style features, but I have found it much easier to use my Blackberry.

With all of that said, I still love my Zaurus, and I'm glad to say that the OpenBSD team continues to work on it. I can say in all honesty that it's helped me out on a few occasions where I really needed to have a mini unix machine. As I mentioned, it makes a decent web server, and it's great for trying out shell scripts on the go. The other day some children on the train mistook my Zaurus for a Nintendo DS. *Look mom! He has a white DS!*, they said. That made me smile to myself. At the time, I was experimenting with the Zend PHP Framework, learning how to use it implement access control for a web project. The Zaurus looks like it's meant to play games, but I was getting work done. Times like that make me realize how useful it can be, even with all of its downsides. I wouldn't recommend a Zaurus for everyone, but I know that in the right hands, a Zaurus running OpenBSD can be exactly what it says on the right side of its screen a Personal Mobile Tool, and a quite powerful one at that.

Questions and Answer Session of the BSD Certification Group Community

Dru Lavigne and Mikel King

Recently, the BSD Certification Group (BSDCG) asked via their mailing lists for questions regarding the BSDCG or the BSDA exam, offering to answer them in this issue of BSD Mag.

Are there still two competing BSD certification groups? If so, what's the difference?

Around the same time the BSDCG formed in January, 2005, a separate commercial company started with a similar goal of assessing BSD system administration skills. The BSDCG approached the company to see if the two organizations should align. Since the company had already launched an exam and the BSDCG wanted to concentrate on creating a standard on which to base exams, the two organizations wished each other well and continued to pursue their goals separately. As a registered non-profit, the BSDCG established bsdcertification.org and the company continued to use bsdcertification.com.

In late July, 2008, the company re-contacted the BSDCG indicating that they were no longer providing an exam and offered to transfer the .com domain to the BSDCG. The BSDCG accepted the offer and now controls both the .org and the .com domains. Since its founding, the BSDCG has published the standard for assessing junior BSD system administration skills and launched the associated BSDA exam in February, 2008. The BSDCG is now creating the standard for advanced BSD system administration skills which will be used to create the upcoming BSDP exam.

Has anyone done a comparison of the BSDA and LPIC Certifications? I think that would be helpful for managers who are not familiar with the BSDA and what it covers.

The BSDA exam is scored for 100 questions covering the following 7 knowledge domains:

- Installing & Upgrading the OS and Software (13%)
- Securing the Operating System (11%)
- Files, Filesystems, and Disks (15%)
- Users and Accounts Management (16%)
- Basic System Administration (12%)
- Network Administration (15%)
- Basic Unix Skills (17%)

The LPI equivalent is the LPIC-1. Its knowledge domains cover:

- System Architecture (8%)
- Linux Installation and Package Management (11%)
- GNU and Unix Commands (26%)
- Devices, Linux Filesystems, Filesystem Hierarchy Standard (15%)
- Shells, Scripting, and Data Management (10%)
- User Interfaces and Desktops (5%)
- Administrative Tasks (12%)
- Essential System Services (10%)
- Networking Fundamentals (14%)
- Security (9%)

Both exams cover similar skills, target the same audience of junior system administrator, and provide detailed exam objectives. A major difference can be seen in the exam cost and delivery method. The BSDA costs $75 USD and is available in a paper-based format. The LPIC -1 costs $320 USD (as you have to take 2 exams) and is available through VUE and Prometric. The cost of the LPIC-1 is due to the high cost of using the proprietary services of VUE and Prometric. Three primary goals of the BSDCG are:

- to provide exams at a globally affordable price
- where possible, to use open source delivery solutions
- to work closely with the BSD community and employers who use BSD to offer and promote exams that assess BSD skills

How many people have taken the BSDA exam? Where has the exam been offered?

As of this writing:

- 1,114 have registered for a BSDCG ID (needed to take the exam)
- 102 have taken the exam
- 70 have passed the exam and hold a valid BSDA
- 9 exams are yet to be scored

Since its launch in February 2008, the BSDA exam has been hosted at various conferences around the world. Several user groups and employers have also hosted the exam. Past events and their locations include:

- SCALE: Los Angeles, California
- FOSDEM: Brussels, Belgium
- Linux-Tage Chemnitzer: Chemnitz, Germany
- Flourish: Chicago, Illinois
- IT360: Toronto, Ontario
- NLUUG: Ede, The Netherlands
- BSDCan: Ottawa, Canada
- CONFidence: Krakow, Poland
- LinuxTag: Berlin, Germany
- RMLL: Mont-de-Marsan, France
- OpenKyiv: Kiev, Ukraine
- LinuxWorld: San Francisco, CA
- FrOSCon: Sankt Augustin, Germany
- Augsburger Computer Forum: Augsburg, Germany
- Open Source Days: Copenhagen, Denmark
- NYCBSDCon: NYC, NY
- EuroBSDcon: Strasbourg, France
- CONISLI: Sao Paulo, Brazil
- Brandenburger Linux-Infotag: Potsdam, Germany
- MeetBSD: Mountain View, CA
- DCBSDCon: Washington, DC
- AsiaBSDCon: Tokyo, Japan
- Atualtec: Boa Vista, Roraima, Brazil
- LinuxFest Northwest: Bellingham, WA
- devGuide: Hamburg, London, and Amsterdam
- Certified Secure: The Hague, Netherlands

We have had requests for exams in other cities and currently have an outstanding request for an exam event near Morocco. Hosting an exam event is an excellent way to show your support for BSD and to connect with other BSD users.

My company or user group is interested in hosting an exam event. How do we do this?

Excellent! In order to host the exam, you need to:

- provide a quiet room with sufficient desks/tables and chairs for up to 10 people to sit comfortably and not too close to each other
- provide a contact for the proctor and for exam takers (e.g. to provide directions to the venue, provide access to washrooms, etc.)
- demonstrate that at least 4 people are willing to attend the event (we will help you find more)
- assist the BSDCG in promoting the event (e.g. on your website, mailing lists, etc.)
- give at least 6 weeks notice to provide ample time to promote the event, find a proctor, and ship the exams

The BSDCG will:

- handle registration and payment for the exam
- find the proctor nearest to the event location
- provide the exams and pencils
- contact those who take the exam regarding their results within 3-5 weeks

If you're interested, send an email to *chair@bsdcertification.org* to make the necessary arrangements. It should be noted that exam proctors are volunteers who donate their time and travel. Many take time off from work and pay travel and accomodation costs to another city or even another country in order to promote the BSDA. When sponsoring an event, let us know if you are able to *pass the hat* or wish to sponsor some or all of the proctor's travel costs. We'll put you in touch directly with the proctor to make the necessary arrangements.

Are there any success stories or any sightings of job listings "in the wild" that specifically call for any BSD certs?

We know of one job candidate who received a job interview because BSDA was on his resume. He was subsequently hired and the same company has paid for several of their existing employees to take the exam. We've also had several large companies express interest in hosting an exam session for their employees, for both the BSDA and the upcoming BSDP.

On the various BSD jobs mailing lists we have seen a few job postings mention the BSDA and have noticed that more of those looking for work are mentioning their BSDA. We encourage those holding a BSDA to mention it on their resume and social networking sites. This helps start the *snowball effect*, increases exposure to the BSDA, and adds value to the certification.

Earlier this year we started two LinkedIn groups. The BSD Certification group (*http://www.linkedin.com/groups?home=&gid=1600767*) is for anyone interested in BSD system administration and is a great way to network with other BSD system administrators. The BSDA Certified group (*http://www.linkedin.com/groups?gid=1600807*) is limited to those holding a valid BSDA and is an excellent resource for employers looking to hire someone with a BSDA. If you're a BSD sysadmin, we encourage you to join the group(s).

Interview with Albert Whale

ABS computer technology

Could you please briefly introduce yourself to our readers?

My name is Albert Whale, the President of ABS Computer Technology. I am a resident of Pittsburgh, PA, and I work with my company on Security and Consulting opportunities on an international basis.

I have been a Consultant ever since my first job out of college in 1985. So first and foremost I see myself as a solution provider, a problem solver. The same is true for my company, ABS Computer Technology as we are usually called upon by companies when others cannot resolve the problem, or do not know where to start to get their issues resolved.

I have always seen myself as the Maverick in the crowd, because I never followed the path of others, and I usually have an unconventional or totally controversial solution to the matter at hand.

If you continue to use the same tools techniques and processes as other companies, are you expecting better results?

What does ABS Computer Technology do?

Our specialty is security consulting, auditing, risk assessments and Network Design. We specialize in analyzing the needs of the organizations we are working for, and design a customized solution to meet their logical, physical and organizational needs.

While most of our work is technical consulting for network design, we also perform a great deal of consulting for Linux and Unix systems as well. We have recently partnered with both Mandriva and Red Hat for Linux solutions.

We also offer wireless security, secure web hosting and Spam elimination for our customers. We have leveraged our experience for our small to mid size customers, which help them as well.

Why did you decide to deliver solutions for Open Source?

We started offering solutions with Open Source, because we saw that all of the solutions in shrink wrap either came from Open Source beginnings, or were made better by Open Source software. I also think that Open Source software was more stable and had better problem resolution than software which came from the Big Software Giants.

Before I started ABS, I was always told that the Linux or Open Source software solutions were great ideas, but that few businesses would use these solutions because no one wanted to support (or offer support) for them.

The final straw was an interview I heard by Bill Gates. Bill stated that … *you should not upgrade from Windows 3.1 to Windows 95 if you wanted to fix problems in the software. Because Windows 95 would not have the same updates that Windows 3.1 had, you should continue to pay support for windows 3.1 OS.* Bill said, *the way that Microsoft worked on their software was to take a snapshot of what was working, and then split their team for develop of the new software, and keep the remainder working on the old software (bug fixes, updates …).* While it made sense for what they were doing, it also explained why nothing worked the way I expected.

This historic interview has stuck with me since I heard it. Most people upgrade their software to resolve problems, and get more features. If Microsoft was selling a newer product but not delivering a real solution, what good was it?

This type of interview never happened in the Open Source community, and if

the product didn't work, the issues were fixed. Functionality never took a backseat to the Marketing department in the Open Source community.

What security solution must a corporation use to fully protect its data?

There is no One security solution that a company must use to protect it's data. A true security professional will tell you that it takes multiple tools to protect your data, including the use of off-site storage, network scanners, Firewalls, monitoring and education.

The one factor that is the hardest one to predict and to protect is the human factor. However, educating your employees about their protecting your data can help to reduce the threat of Social Engineering.

There is no magic tool that does it all, especially for everyone. Security is hard. But then nothing good ever came easily. Anyone that tells you that security is easy doesn't know what they are talking about.

Is there a possibility to eliminate the risk at all or just reduce it to minimum?

Yes, but in doing so you still need to be able to use the equipment and the network. Risks can be minimized in many Layers. Such as, the Application Layer, the Network Layer, and the human layer. The more complicated or diverse the network, the more complex the solution can be.

Yes the risk can be mitigated to an acceptable level, but some effort is required to secure the applications, the network, provide redundancy, off-site data storage, and the most important part, educate the employees as to their role in security.

The more effort that is put into these elements, the better the security will be. If you look at the security of your home, the old saying is that locks are for honest people, the professionals are not going to use the front door.

After all, the efforts of the hackers are financially motivated. They spend countless hours developing exploits, attacks, and new techniques to deliver their Malware. (or in the case of employee theft or issues, more difficult for the security to be broken).

What security tools/measures do you need for secure hosting?

We use our Server Safe, Active Defense and SpamZapper® tools to provide secure hosting, and Email Security for our customers.

What is Active Defense? How it differs from traditional security?

Active Defense is our way to monitor the connections to the server to verify that they are used the way that they are supposed to be. Valid connections remain active, while connections which are being abused are terminated. The hacking performed on connections is sometimes so intense that the valid connections are either delayed or unable to communicate. Active Defense stops the hackers so that the servers can continue to communicate.

Do you have any advice for people to surf the Internet safely?

Be careful where you connect, what you download, and what you store on your computer. I use the No Script plug-in for Firefox to limit the JavaScript access to my connection.

I try to download software only from well-known sites, and I have strong encryption on my computers to encrypt the data stored on them.

What is Black Belt Security?

We coined our Security process Black Belt security, to represent the steps it takes to achieve a Black Belt in any Martial art (like Tae Kwon Do). Just as Tae Kwon Do requires multiple steps (or belt levels) to achieve a Black Belt, actual security requires multiple steps and processes to achieve your goal, namely security.

Black Belt security is our multi-layered approach to security for business. Security is not something that is easily achieved (you cannot buy it simply off of the shelf, and then you're done). We have heard of different process, Six Sigma, Lean Six Sigma, CoBIT, and others, Black Belt is our named approach to solving the security needs for business.

Do you have a recommended toolkit for

a. Diagnosing security problems with home and small business systems.

Any virus scanner will diagnose the security problem. The first virus detected is the best indicator on whether or not there are security problems. The problem with using a virus scanner to determine your security problems is that they are unreliable, and may not be able to detect all of the security issues. We use fairly sophisticated scanning tools when we investigate our customers' security needs, and unfortunately these are not easily used or understood by most non-professionals.

b. Repairing these found problems

We highly recommend Ad-aware and Spybot. We have found these two tools to be highly effective.

c. Preventing future invasions.

We have found an excellent Firewall for PCs which we recommend for our customers. Anyone can download it from our website, at *http://www.abs-comptech.com/free-offers/free-software-firewall-protects-you-better.*

I have read that some system altering programs are near impossible to detect. Do you have any way of investigating these problems and getting rid of them?

Using our SpamZapper® for email helps to minimize the potential for receiving these zero-day exploits, which probably contain the Malware you are referring to. The Free firewall will also detect the applications which are connecting to the internet from your computer, and will request permission to permit these connections. Additionally, the firewall also has a built-in scanner which will validate all files on access, if you want.

There are more advanced strategies which we employ, but for the non-professional these are exceptional tools which they can easily benefit from.

About Albert Whale

Albert Whale CHS CISA CISSP is located in Pittsburgh, PA where he manages ABS Computer Technology, Inc. and provides support for their customers. If you have questions you can reach Albert at aewhale@ABS-CompTech.com or at 1-412-635-7488 ext. 100

Interview with Matt Juszczak

Recently I had the opportunity to sit down with Matt Juszczak creator of BSDJobs.net and ask him some questions about the project. I would like to thank Matt for taking the time out of his busy schedule to talk with me about the project. The following is a synopsis of that Q&A session.

Why BSD? I mean what made you choose BSD as the focus of this project?

Many years ago, I would have likely given a fairly common and honest answer: it's what I found first, and it's what I'm used to. My skill sets have grown, and I've had the opportunity to work with other Unix-like operating systems, and throughout this experience I've remained a FreeBSD fan. I enjoy FreeBSD's centralized configuration, pf, out of box security, implied simplicity, and most importantly, the ports collection and ease of package creation. I also like the idea that FreeBSD is an operating system with additional packages, instead of a group of packages making up an Operating System. To be fair, if I had to pick a favorite Linux distribution, I would most likely pick CentOS.

With all of the other job listing and search sites already online such as Dice.com, Monster.com, TheLadder.com et cetera... What made you decide to focus on BSD?

These sites are too complex. They have actually overwhelmed me many times, enough for me not to post there (though I still do). I'm a big believer in the motto *simple and functional*. BSDJobs does just that: it provides a simple method to connect people with similar needs. It does only one thing, but it does it well. I can assure you that BSDJobs will never look prettier, but it will definitely become more functional.

Exactly what BSDs are supported by BSDJobs.net?

FreeBSD, OpenBSD, NetBSD, and Darwin (Mac OS X). There is also a non-specific category. I've made a note to add support for other BSDs, including DragonFly BSD.

How long has BSDJobs.net been operational?

The idea began in November, 2004. It was developed and launched in early 2007.

Do have of any success stories that you would like to share with the readers of BSD Magazine?

I've been working on many projects lately that I believe have started showing signs of success. However, I do have one BSD-specific success story which I hope will even further show the importance of BSDJobs. Around the time BSDJobs was being developed, I visited Chicago for the first time in quite a long while. As I was walking around downtown, I (literally) bumped into a local resident of the city. In the few minutes that I spoke with him, we stumbled upon an important common ground: he was desperately looking for a FreeBSD systems administrator to takeover his abandoned infrastructure, and I happened to be a FreeBSD systems administrator looking for side work. Two years later, I am still working on that project, and the BSD infrastructure has grown substantially.

I have heard and experienced similar stories from other employers and systems administrators. My hope is that BSDJobs will help reduce fear in those who make Operating System decisions, and increase the ability to easily find BSD systems administrators to provide endless support and replace vacant positions.

Is there a cost for using BSDJobs.net?

No, and there are no plans to implement any. The point of BSDJobs is to provide a free resource for anyone who wishes to use it, both job posters and job seekers alike. As the site grows, anti-spam measures (which has already become a problem) will need to be increased, as well as contributions from the community, but the site should always remain free.

How has the volunteerism in the BSD Community helped the site?

I've had press releases written. I've had help with the graphics of the site. BitVenue Networks, LLC (*http://www.bitvenue.com*) donates hosting for the project. There is much more work to be done, so volunteers are always welcome. The more, the merrier.

Do you have any major changes planned for the site in the near future?

At some point in the near future, a full overhaul of the site's code base will occur. The site will still look the same, but we'll be adding functionality, such as better integrated communication, increased captures to fight spam, listing edit features, and better searching.

So what version of BSD do you run the site on?

Right now, FreeBSD 6.3. We are migrating it to a FreeBSD 7.1 setup this Summer.

What made you choose FreeBSD over the others for this project?

The original web server hosting the project ran FreeBSD. FreeBSD is also my BSD of choice and the one I'm most familiar with. I also have experience with OpenBSD and NetBSD, and I believe either would have worked just as well for this project.

Do you have any other projects like this one that you would like to mention?

It amazes me how many web projects created this decade seem to focus on using as much technology as possible in the wrong ways instead of efficiently using available technology in the right ways. I've gotten to the point where I prefer to use my bank's mobile website in my normal web browser to get more accomplished in less time. This attitude has begun to influence my personal projects and ideas, and there are definitely a few in the works!

How have employers responded to BSDJobs.net?

I haven't had a lot of communication with users of the website. However, when I have replied to postings as a job seeker, I have received replies back, which makes me think people are seriously about the website. Furthermore, posts continue to come in reliably. The site gets anywhere between 800 and 1600 visits per month. Any feedback is welcome and can be submitted via the *About* page on *http://www.bsdjobs.net*.

Does BSDJobs.net have a search by specialty or certification feature?

You can search based on location and operating system. The BSD Certification (*http://www.bsdcertification.org/*) project is much more active now than it was when the site was being designed, so I think it would be wise to add this functionality.

Do you allow recruiters or head hunters to use the site?

MJ-Yes. The primary goal of BSDJobs is to fill open positions in the market. While it would be nice to increase the ability for employers to connect with job seekers directly, this isn't always possible or feasible. If recruiters and head hunters have a BSD-oriented position they are looking to fill, they are more than welcome to post.

Name your top 5 favorite Open Source projects in rank order.

Making a list of the most important open source projects is somewhat hard for me to do, considering I use so many on a day to day basis. As a Systems Administrator, I find myself working with FreeBSD and Linux quite often. As a big fan of standardization and centralized configuration, I often use OpenLDAP (*http://www.openldap.org*) to store system account information and other data. As a Database Administrator, MySQL (*http://www.mysql.org*) has played an important role in my career. Last but not least, Nagios (*http://www.nagios.org*) and Cacti (*http://www.cacti.net*) have allowed me to go to sleep at night knowing servers, databases, ldap directories, and any other critical component of a project are running smoothly.

Since I know that you are a user of MySql for some projects, what are your feelings about Oracles acquisition of Sun Microsystems?

As I understand it, these projects will now be housed under the same roof, so to speak. I worry that this acquisition may hurt MySQL as a product, but I trust that this will not be the case. I have used Oracle and MySQL in the work place. Both have their strengths and are fundamentally different. As always, continue to use the best tool for any given project, but these two should have no problem co-existing.

Do you have any advice for anyone wanting to launch an Open Source project of their own?

Getting started is the hardest part. Be pro-active, work with a small team, and do what you need to do to remain inspired, even if that means taking your laptop to a park with a solar-powered laptop charger and mobile-phone Internet connectivity until you've solved a problem. Work with people who share your passion and interests, but have different backgrounds and ways to contribute. Finally, attempt to keep conflict at a minimum.

What are your feelings if any about the Google Summer of Code events? Do you feel that Google has a hidden agenda? Do you feel that they are truly beneficial to the Open Source Community?

I haven't had enough time to form an opinion on whether or not there are any conspiracies going on. I've benefited a lot from GSOC projects. I believe there are many students in higher education who find it hard to remain challenged in their curriculum and gain „real world" experience, so perhaps this gives them a good opportunity to grow their skills. Furthermore, one major weakness in some open source projects is their lack of information on how to get involved, so GSOC can help provide that gateway.

So what is next for Matt Juszczak? Any personal achievements, plans or milestones you would like to share with the readers?

I am currently residing in Boston, MA after moving here in late 2007. I am involved in a large data center migration project with a company based in New York City. During that migration, about 75% of the infrastructure has been moved to FreeBSD. I am continuing to work on many web-based projects, including BSDJobs, as well as a venture that allows me to consult with other groups of individuals to help grow their ideas from the ground up, assisting them in implementing open source technologies, data center infrastructure, and a reliable and easily maintainable code-base.

by Mikel King

About the Author

Mikel King (http://twitter.com/mikelking) has been working in the Information Services field for over 20 years. He is currently the CEO of Olivent Technologies, a professional creative services partnership in NY. Additionally he is currently serving as the Secretary of the BSD Certification group as well as a Senior Editor for Daemon News.

Tips&tricks

FreeBSD Jails by Josh Paetzel

FreeBSD jails have always been an attractive way to isolate services from each other, and provide additional system security by compartmentalizing services from each other. In addition to compartmentalization FreeBSD jails can provide additional security to services.

Many services that are run on systems require the service run as the root user in order for the service to bind to a privileged port. Using jails, PF, and virtual ethernet interfaces it is possible to start services in a jail on a high numbered port, while still having it accessible on the expected port to the outside world. For the sake of this article apache will be used as the service. Things you will need:

- FreeBSD
- Apache
- PF
- A jail
- A tap interface

For starters, create a tap interface.

```
# ifconfig tap0 create
```

Then create a jail (Listing 1). Add the following to `/etc/rc.conf` (Listing 2).

Start pf

```
# /etc/rc.d/pf start
```

Start your jail

```
# /etc/rc.d/jail start
```

Move in to your jail

```
# jexec 1 csh
```

Install apache

```
# pkg_add -r apache22
```

Since we will be running apache as non root there are some modifications that need to be done. In particular we must bind to port 8080, and files that apache wants to write to at startup must be writeable by our prefered user.

In `/usr/local/etc/apache22/httpd.conf` set the Listen directive to port 8080, remove `mod_unique_id`, and uncomment out the line at the bottom that sources in `extra/httpd-mpm.conf`.

In `/usr/local/etc/apache22/extra/httpd-mpm.conf` change the PidFile to `/var/run/apache22/httpd.pid` and the LockFile to `/var/run/apache22/accept.lock`. Change the logs to be owned by our non root user

```
# chown www:www /var/log/httpd*
```

Create the needed directories for apache startup

```
# mkdir -m 700 /var/run/apache &&
chown www:www /var/run/apache
```

Change to the www user

```
# su -m www
```

Start apache

```
# apachectl start
```

Look for root processes running apache

```
# ps -waux | grep httpd
```

The rc script that is normally used to start apache will need some changing to avoid starting as root, but at this point you have apache running with no root privileges at all, listening on port 8080, which thanks to your firewall gets all traffic bound to port 80 on the external IP.

Listing 1. Create a jail

```
# csup -h cvsup9.freebsd.org /usr/share/examples/cvsup/standard-supfile
# cd /usr/src
# mkdir -p /usr/jails/www
# make buildworld installworld distribution DESTDIR=/usr/jails/www
# cp /etc/resolv.conf /usr/jails/www/etc/
# touch /etc/fstab.www
```

Listing 2. /etc/rc.conf

```
pf_enable="YES
jail_enable="YES"
jail_list="www"
jail_interface="tap0"
jail_www_rootdir="/usr/jails/www"
jail_www_hostname="www.example.org"
jail_www_ip="10.1.1.1"
jail_www_exec_start="/bin/sh /etc/rc"
jail_www_exec_stop="/bin/sh /etc/rc.shutdown"
jail_www_devfs_enable="YES"
jail_www_mount_enable="YES"
jail_www_flags="-l -U root"
jail_www_fstab=""

Create an /etc/ pf.conf

ext_if="em0" # will vary by system

nat on $ext_if from 10.1.1.1 to any -> ($ext_if)

rdr on $ext_if proto tcp from any to any port 80 -> 10.1.1.1 port 8080
```

Useful ssh tricks and tips. by Mikel King

Since this months' issue revolves around security I thought it a wise idea to discuss some tips and tricks that are security related. To that end I hope to explore some of the common useful options for ssh. First we will examine TCP port redirection using the ssh client, which can generally only be accomplished via root level privileges.

Since we are not going to alter the `sshd_config` to allow ALL users on the system the redirection privilege I am assuming that you have a working system where you hold the proverbial keys to the kingdom. If I am mistaken then perhaps you should download an ISO of your favorite BSD or even a live DVD like RoFreesbie so that you can play along.

First, I would like to discuss why one might consider creating a ssh TCP tunnel. Let us decide that you are visiting a new client for the first time and have not had a chance to setup your normal exclusionary firewall rules, and further that this client's network is one you do not entirely trust as of yet.

However, you need to access data on the intranet back at your office. This could be some files, or your client database, or even you jabber server. While there are numerous methods available to facilitate this sort of action we are going to tunnel some TCP via an ssh connection.

There for in this example let's expect that you need to access your MySql database securely form outside of your home network. As previously mentioned we will assume that you have root level access on the source system, which is most likely you personal laptop.

Reading the ssh man page you will note the `-L [bind_address:]port:host:hostport` which may seem cryptic at first, however, we will deconstruct the command one parameter at a time. First consideration is the `bind_address`, this is only an issue if your system has multiple address and you wish to specify which one to use for the outgoing connection.

This is the only optional parameter in the statement one that we can safely ignore. The port refers to the port on your local machine at this end of the tunnel, in other words the port that you wish to map the service on target machine to.

The host refers to the address of the host on the remote side of the tunnel. This host may the the target machine itself or another machine available on the same LAN as the the target. Finally the hostport is the TCP port that you wish to connect to.

In this exercise we will be connecting to our database server OSIRIS.olivent.com via another server *PTAH.olivent.com*.

These machines have appropriate DNS entries so as to ensure that I can always connect to them by their proper name. From here after I'll simplify things by only referring to them by their short names in all capital letters for clarity.

In the following example I will be opening a connection to the target machine ptah as the user sysmgr.

```
# ssh -N -f -L 4406:OSIRIS:3306
sysmgr@PTAH
```

As you can see that did not really do very much, now on my local machine I can direct my MySql client to connect as follows.

```
OSIRIS> mysql -h 120.0.0.1 -P 4406 -u
dbadmin -p
Enter password:
Welcome to the MySQL monitor. Commands
end with ; or \g.
Your MySQL connection id is 26621
Server version: 5.0.67-log Source
distribution
Type 'help;' or '\h' for help. Type
'\c' to clear the buffer.
mysql>
```

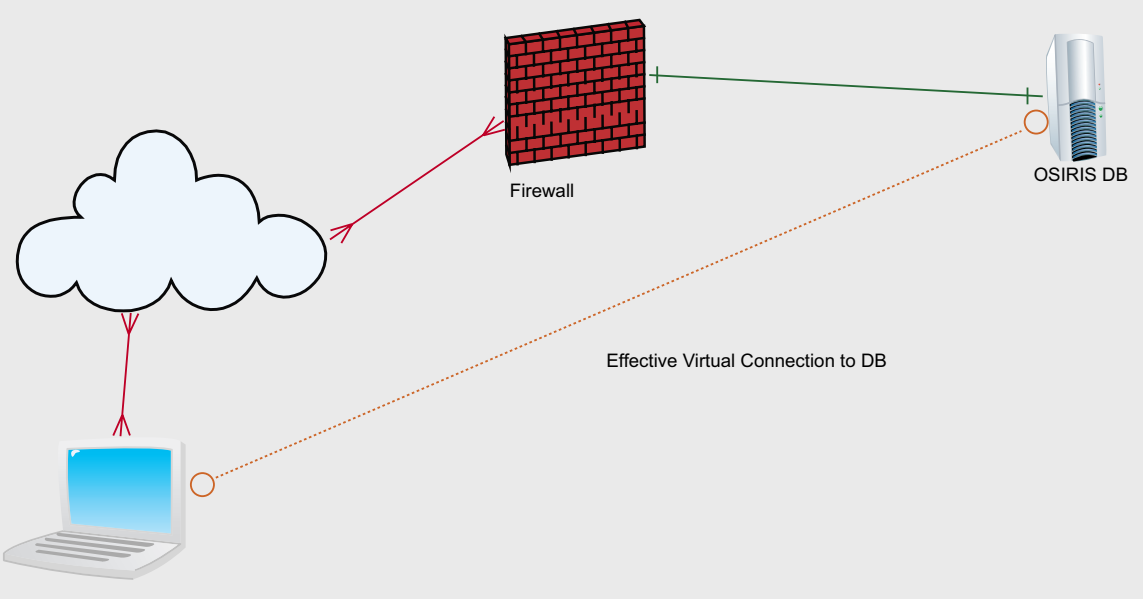


Figure 1

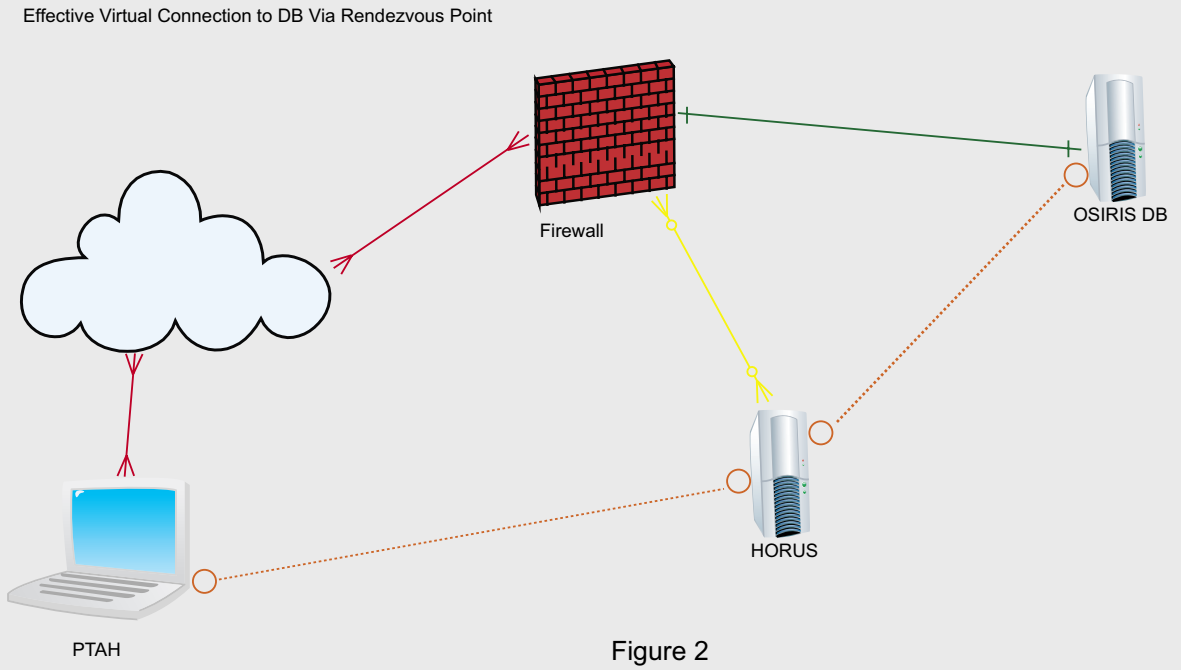


Figure 2

To summarize what thus far we have successfully established an ssh tunnel to our target and told ssh that no CLI access is needed as well as to send the connection to the background. We were then able to connect to the MySql database pretty much as we would if we were sitting at the console of the server in question, by simply adding the appropriate host and port switches as demonstrated above. Refer to Figure 1 below for more detail.

Suppose, however, you manage a site and need to allow a vendor to access and troubleshoot a server, but do not wish to grant this vendor full access to the entire network. How do you allow them to complete their work without being able to peruse your entire network? The answer is called a rendezvous point.

In order to facilitate rendezvous point you need three machines. The server, the client, and the way station. The server and client are fairly obvious, but the way station is the meeting point in this case we will call that machine HORUS. HORUS lives on the DMZ and exists solely for the purpose of facilitating these sorts of connections. It's firewall rule prohibit more external access excluding ssh of course.

In the following example first the database server OSIRIS is connected to the way station HORUS.

```
OSIRIS# ssh -N -f -R 4406:127.0.0.1:
3306 sysmgr@HORUS
```

Then the vendor on PTAH connects to the way station as shown.

```
PTAH# ssh -N -f -L 5506:127.0.0.1:
4406 sysmgr@HORUS
```

Meanwhile back on PTAH in another terminal we open the database utility connecting to the newly bound 5506 port on their local IP address. Refer to the following CLI excerpt as well as Figure 2 for more details.

```
PTAH> mysql -h 120.0.0.1 -P 5506 -u
dbadmin -p
Enter password:
Welcome to the MySQL monitor. Commands
end with ; or \g.
Your MySQL connection id is 26626
Server version: 5.0.67-log Source
distribution
Type 'help;' or '\h' for help. Type
'\c' to clear the buffer.
mysql>
```

As you can see from the demonstration above the vendor is able to access the database and perform what ever maintenance is required within the limitations of their database utilities. To further secure this method one could issue a ssh key pair so that no passwords need to be exchanged in the first place. What is nice about this later step is that once the maintenance has been completed simply revoke the vendor's key at the way point HORUS and terminate the tunnel from OSIRIS to HORUS.

In addition if the vendor's account is compromised in anyway the only access will be granted to HORUS which knows absolutely nothing about your internal network. In fact other than being a basic BSD server it should know nothing about databases, DNS, mail or anything other how to connect to the internet. Obviously it adds a layer of complexity to the whole process, as well as yet another server to maintain, but in the end is you have a large installation of vendor supported equipment and loath the idea of letting them run amuck about your network it certainly is viable option.

About the Author

Mikel King (http://twitter.com/mikelking) has been working in the Information Services field for over 20 years. He is currently the CEO of Olivent Technologies, a professional creative services partnership in NY. Additionally he is currently serving as the Secretary of the BSD Certification group as well as a Senior Editor for Daemon News.

Manage fast information access and high quality files recovery with

UFS EXPLORER

Data Recovery and Data Access solution for multiple Operating Systems

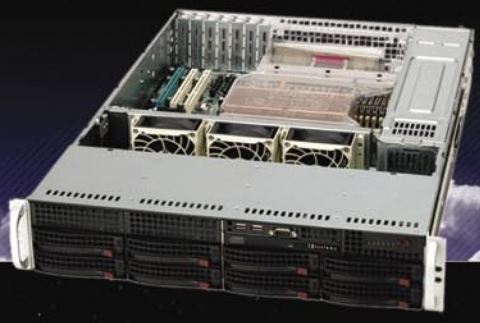
UFS Explorer software targets fast and easy access to files located on disk partitions of different operating systems. This includes access to neighbour disk partitions and removable storages formatted with file systems of alternative OS as well as file access on virtual machines of pupular virtualization software vendors.

The Recovery editions of UFS Explorer products meet industry standards and enable high quility solutions for lost and deleted files recovery from file systems, created with such major operating systems as Microsoft Windows, BSD, Sun Solaris, Apple MacOS, Linux and so on. The products support workstations, servers and embedded systems and cover data recovery from stand-alone fixed or removable storages as well as even complex NAS and RAID systems.

If you are interested in data access optimization or lost information recovery, just try UFS Explorer product for free.

Find more information at http://www.ufsexplorer.com/

iX-Neutron
A Star Among Servers



PROFESSIONAL SERVERS FOR YOUR BUSINESS

In striving to bring our customers faster, more reliable servers, iXsystems, Inc. introduces the new iX-Neutron server line. The iX-Neutron server series brings Intel's® newest chip technologies to your business to provide an astronomically fast family of machines. The Intel® Xeon® Processor 5500 Series utilizes these technologies to greatly increase speed, performance, and memory capacity, while saving energy simultaneously. The processor performance scales dynamically based on the requests and demands of the system. Visit us at http://www.iXsystems.com/neutron for more information and pricing.

iX-N1204

- 1U Form Factor with 4 Hot Swap SAS/SATA 3.5" Drive Bays
- Dual Intel® 64-Bit Socket 1366 Quad-Core or Dual-Core, Intel® Xeon® Processor 5500 Series
- Intel® 5520 Chipset with QuickPath Interconnect (QPI)
- Up to 96GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (12 DIMM Slots)
- 2 PCI-E 2.0 x8 or 1 PCI-E x16 Expansion Slots
- Intel® 82576 Dual Port Gigabit Ethernet Controller
- Matrox G200eW Graphics
- Remote Management-IPMI 2.0 + IP-KVM with dedicated LAN
- Slim DVD
- 650W Redundant 80%+ High Efficiency Power Supply

iX-N2280

- 2U Form Factor with 8 Hot Swap SAS/SATA 3.5" Drive Bays
- Dual Intel® 64-Bit Socket 1366 Quad-Core or Dual-Core, Intel® Xeon® Processor 5500 Series
- Dual Intel® 5520 Chipsets with QuickPath Interconnect (QPI)
- Up to 144GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (18 DIMM Slots)
- 2 PCI-E 2.0 x16, 4 PCI-E x8, (1 in x16 slot) and 1 PCI-E x4 Expansion Slots
- Intel® 82576 Dual Port Gigabit Ethernet Controller
- Matrox G200eW Graphics
- Remote Management-IPMI 2.0 + IP-KVM with dedicated LAN
- Slim DVD
- 700W Redundant 90%+ High Efficiency Power Supply

iX-N3216

- 3U Form Factor with 16 Hot Swap SAS/SATA 3.5" Drive Bays
- Dual Intel® 64-Bit Socket 1366 Quad-Core or Dual-Core, Intel® Xeon® Processor 5500 Series
- Dual Intel® 5520 Chipsets with QuickPath Interconnect (QPI)
- Up to 144GB DDR3 1333/1066/800 SDRAM ECC Registered Memory (18 DIMM Slots)
- 2 PCI-E 2.0 x16, 4 PCI-E x8, (1 in x16 slot) and 1 PCI-E x4 Expansion Slots
- Intel® 82576 Dual Port Gigabit Ethernet Controller
- Matrox G200eW Graphics
- Remote Management-IPMI 2.0 + IP-KVM with dedicated LAN
- Slim DVD
- 800W Redundant 80%+ High Efficiency Power Supply

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

iXsystems™

intel Xeon inside™
Powerful. Intelligent.