BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

BSDs AS SERVERS

INSIDE

Vol.3 No.2
Issue 2/2010(8)
1898-9144

Energy Efficient, Powerful Performance

With dual Intel® Xeon® 5500 series Quad-Core or Dual-Core processors and up to 144GB of DDR3 memory, the iX-Athena is designed for small businesses seeking a high performance computing solution.

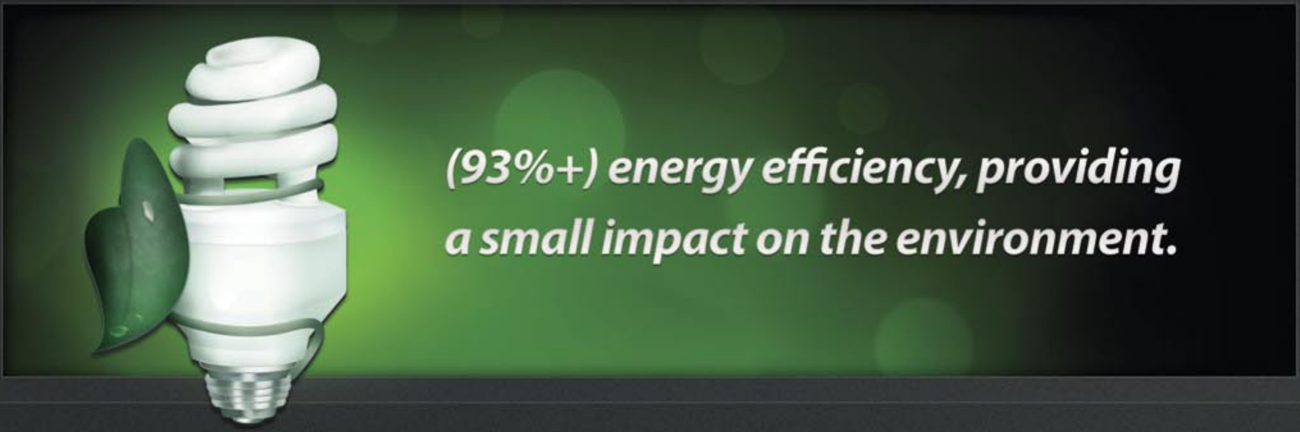
iX-Athena

Notable features include:

- Dual 64-Bit Socket 1366 Quad-Core or Dual-Core, Intel® Xeon® Processor 5500 Series
- Eight 3.5" Hot-swap SAS/SATA HDDs in a 4U/Tower Configuration (Optional 4U Rackmount Rail Kit Available)
- Dual Intel® 5520 Chipsets with Quick-Path Interconnect (QPI) up to 6.4 GT/s
- Up to 144GB DDR3 1333/1066/800MHz ECC Registered DIMM/24GB Unbuffered DIMM (18 DIMM Slots)
- Two (x16) PCI-E 2.0 slots, Four (x8) PCI-E 2.0 slots (1 in x 16 slot), and One (x4) PCI-E slot (in x8 slot)
- Intel® 82576 Dual-Port Gigabit Ethernet Controller
- Matrox G200eW Graphics Support
- Integrated IPMI 2.0 with Dedicated LAN
- Realtek ALC888 7.1 HD audio
- Two 5,000 RPM Hot-swap Cooling Fans
- Two 5,000 RPM Hot-swap Rear Exhaust Fans
- 1400W Redundant High Efficiency Power Supply (Gold Level 93%+ power efficiency)

Front View

Back View


(93%+) energy efficiency, providing
a small impact on the environment.

iXsystems Presents the iX-Athena Workstation

The iX-Athena showcases amazing computing performance and energy efficiency, while keeping noise levels to a minimum.

The iX-Athena delivers the most powerful performance available on the market today. Dual Intel® Xeon® 5500 series Quad-Core or Dual-Core processors boost performance for specific workloads by increasing processor frequency. Next-generation Intel® Virtualization Technology enhances performance in virtualized environments by up to 2.1x with new hardware-assist capabilities. Up to 144GB of DDR3 memory with eighteen DIMM sockets supports higher performance for data-intensive applications and makes it easy for the iX-Athena to handle any workload.

In terms of energy efficiency, the iX-Athena also leads the pack. The automated low-power states of the Intel® Xeon® 5500 series processors intelligently save power during low-use periods and increase performance when the system requires it. The iX-Athena also features an FCC Class B certified power supply with gold level (93%+) energy efficiency to provide 1400W of power and minimize impact on the environment.

The Super-Quiet operation of the iX-Athena allows users to spend less time distracted by a loud machine, and more time focusing on its powerful computing capabilities. At normal operation levels, the iX-Athena workstation's 5,000 RPM cooling and exhaust fans perform at a hushed 38 decibels to make this an ideal machine for any office or lab environment.

With eight 3.5" hot-swappable SAS/SATA hard drive bays, the iX-Athena also offers ample storage for all conceivable technical computing and graphics applications. The iX-Athena even includes four dedicated power connectors for high-end graphics cards, all contained in a stylishly sleek, high-end quality, dark gray chassis.

To order today call:
1-800-820-BSDi

For more information about the iX-Athena visit:
http://www.iXsystems.com/Athena

iX systems™

intel Xeon inside™
Powerful. Intelligent.

Dear Readers!

Please let me introduce to you the first online issue of BSD Magazine. I would like to take this opportunity to thank everyone who made this transformation possible and who supported us in our mission to provide practical knowledge about BSD systems to everyone across the world.

BSD Magazine is now becoming a free monthly publication and this could never be possible without your continuous support. Hopefully, our ever-growing readership will help promote BSD systems and contribute towards their recognition and acclaim worldwide.

This issue is devoted to the subject of servers, the field where BSD systems are amongst the most powerful. We are certain that whichever BSD flavor you're fond of and whatever profession you currently hold you will find something here that might just make your day.

Most of our authors had already been introduced before but this is the first time I had the pleasure of working with them personally and I need to say – it has been a fantastic experience for me. We are absolutely delighted that they decided to stay with us after our transition to the electronic format.

Please keep the emails coming. We appreciate your comments, suggestions and ideas.

Michał Gładecki
Editor in Chief

MAGAZINE BSD

Editor in Chief:
Michał Gładecki
michal.gladecki@software.com.pl

Contributing:
Jan Stedehouder, Rob Somerville, Marko Milenovic, Petr Topiarz, Paul McMath, Eric Vintimilla, Matthias Pfeifer, Theodore Tereshchenko, Mikel King, Machtelt Garrels, Jesse Smith

Special thanks to:
Marko Milenovic, Worth Bishop and Mike Bybee

Art Director:
Agnieszka Marchocka

DTP:
Ireneusz Pogroszewski

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

National Sales Manager:
Ewa Łozowicka
ewa.lozowicka@software.com.pl

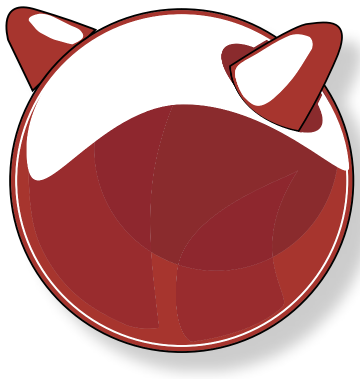
Marketing Director:
Ewa Łozowicka
ewa.lozowicka@software.com.pl

Executive Ad Consultant:
Karolina Lesińska
karolina.lesinska@bsdmag.org

Advertising Sales:
Michał Gładecki
michal.gladecki@software.com.pl

Publisher :
Software Press Sp. z o.o. SK
ul. Bokserska 1, 02-682 Warszawa
Poland
worldwide publishing
tel: 1 917 338 36 31
www.bsdmag.org

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



A first look at PC-BSD 8 release

Jan Stedehouder

With the release of FreeBSD 8.0 it's only a matter of time before we can expect the next version of PC-BSD. At the time of writing plenty of work was being done to prepare PC-BSD 8.0.

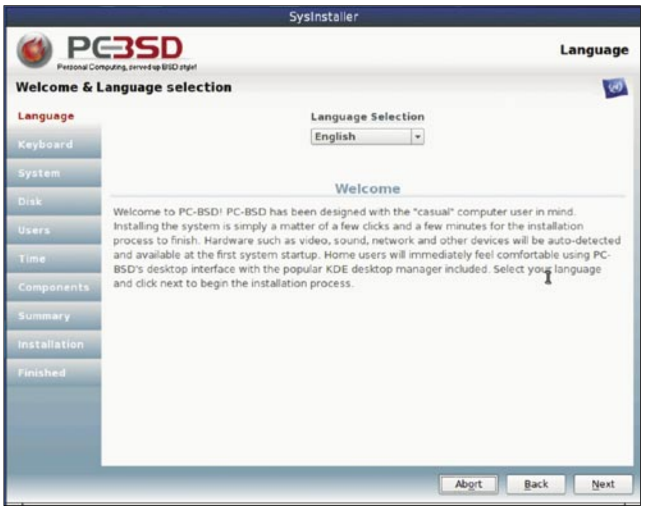
What can we expect from this new release? In this article we will look at the changes in FreeBSD 8.0 that are more of interest to end-users and glance over the novelties in the alpha-releases of PC-BSD 8.0. And we will ask the question what it takes to build an end-user friendly operating system.

What was new in FreeBSD 8.0

The FreeBSD 8.0 RELEASE followed the previous 7.0 RELEASE by somewhat more than a year and a half, which we might almost call a speedy development (compared to the two years and three months between 6.0 and 7.0). Personally I like to compare the development of FreeBSD with Debian GNU/Linux as they are comparable in size (software) and scope (supported platforms) and both geared towards the more technologically adept users. The jump from Debian 4.0 to the latest 5.0 release took from April 2007 to February 2009. So, yes, we can conclude that the FreeBSD team kept a strong pace without sacrificing stability.

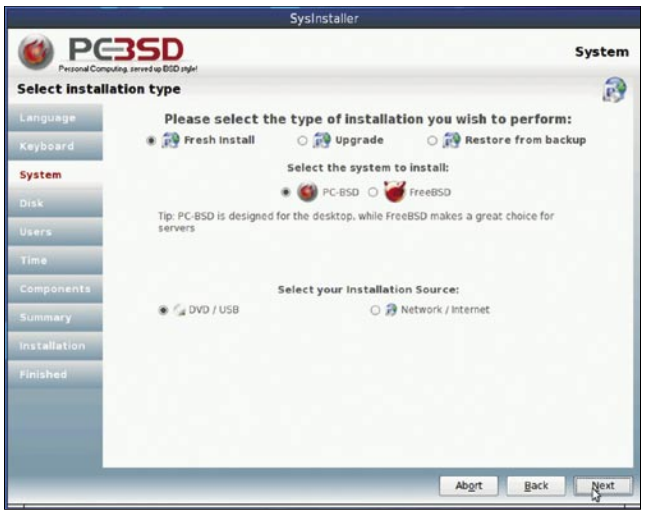
The list of improvements in FreeBSD 8.0 also shows that real ground was covered in pushing it forward. There are improvements in wireless networking, multiprocessing and in the Network File System implementation. The ZFS filesystem is no longer considered experimental. For end-users it is interesting to see that it supports GNOME 2.26.3 (officially released in March 2009, with 2.28 released in September 2009) and KDE 4.3.1 (officially released on 1 September 2009). In both cases FreeBSD managed to have the latest available versions of the desktop environments at the time.

Apart from this, the USB subsystem was rewritten and support for VirtualBox, both as host and guest, has been



Welcome to PC-BSD! PC-BSD has been designed with the "casual" computer user in mind. Installing the system is simply a matter of a few clicks and a few minutes for the installation process to finish. Hardware such as video, sound, network and other devices will be auto-detected and available at the first system startup. Home users will immediately feel comfortable using PC-BSD's desktop interface with the popular KDE desktop manager included. Select your language and click next to begin the installation process.

Figure 1. The new PC-BSD installer allows for more flexibility and control



Please select the type of installation you wish to perform:
[✓ Fresh Install] Upgrade Restore from backup
Select the system to install:
[✓ PC-BSD] FreeBSD
Tip: PC-BSD is designed for the desktop, while FreeBSD makes a great choice for servers
Select your Installation Source:
[✓ DVD / USB] Network / Internet

Figure 2. Installing FreeBSD is now an option with the PC-BSD disk

improved. This makes it easier to test out FreeBSD on your Windows or Linux box, or, run Windows in VirtualBox for those few Windows-based applications that you still need.

Towards a PC-BSD 8.0 release

PC-BSD is built on top of FreeBSD and the work towards the next PC-BSD release began on September 5 when Kris Moore, the lead developer, announced the availability of the first alpha release for version 8.0 (*ftp://ftp.pcbsd.org/pub/alpha-iso/*). Now, at the end of December, we can work and play with the 7th alpha (if I counted correctly), one that is considered feature complete. What can we expect from PC-BSD 8.0 knowing full well that at the time you are reading this article the release candidate or the final version might already be available for download?

As an aside, if you wish to follow along with the work that is being done I suggest you subscribe to PC-BSD testing mailing list (*http://lists.pcbsd.org/pipermail/testing/*).

Flexibility during installation

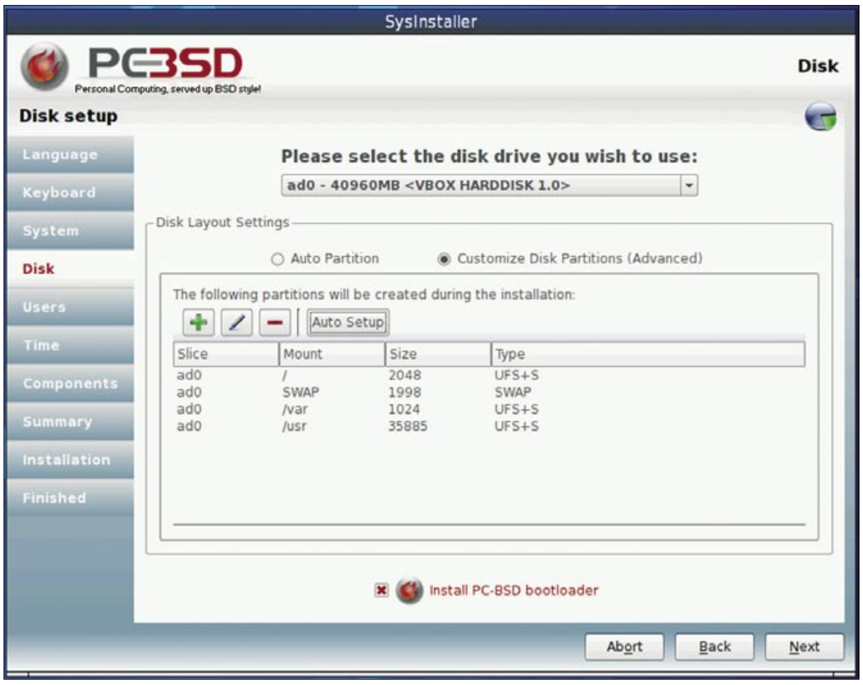
It's hard to miss the major overhaul of the graphical installation wizard (Figure 1). The installer now has a look-and-feel that is akin to that of Fedora, OpenSUSE and Mandriva. The changes aren't simply cosmetic though. The mailing list makes clear that the installer is now scriptable, which is interesting for more advanced users. The graphical wizard also has new features that give the user more control on what to install and where. For example, you can now opt to install FreeBSD instead of PC-BSD from the same medium (Figure 2). Once the FreeBSD is done and you reboot, you can use sysinstall to wrap up the FreeBSD installation.

Users also have more options to fine-tune the partitions during installation. The 7.x release created two partitions as default, root and swap, which you could change with some fiddling. The new installer suggests a partition table that should be more familiar to FreeBSD users (Figure 3). If you have no prior experience with FreeBSD, you won't have to bother yourself with it and simply use the Auto Partition option. With these two changes PC-BSD allows users to remain even closer to FreeBSD than in the 7.x release without sacrificing the easy install for novice users.

Geared more towards new users is the option to boot PC-BSD from the CD/DVD into a live environment (Figure 4). Why is this important? Well, when we look at Linux we see that live CD/DVD's played a major role in reducing prejudices against the open source desktop. Knoppix did a lot to show that an open source desktop was complete and ready to use for everyday computing. It has been a while since we had an up-to-date live disk based on FreeBSD and it is nice to again have a tool to promote PC-BSD/FreeBSD this way.

Tighter integration of PBI installs

PC-BSD 8.0 will sport the KDE 4.3.4 environment. One thing I like about the KDE 4 desktop is how it integrates the local desktop with online services. One example can be found in the options to change the look-and-feel of your desktop. Scattered throughout you can find buttons like *Get New Themes* of *Get New Icons* which open new screens that pull in information from the KDE-Look website. It's no longer necessary to go to the site and browse through the available collection, hoping to find a package that is suitable for your computer. Installing a new theme is a mouse-click away. PC-BSD is now applying the same mechanism to installing and managing



Please select the disk drive you wish to use: ad0 - 40960MB <VBOX HARDDISK 1.0>

Disk Layout Settings: ( ) Auto Partition [✓ Customize Disk Partitions (Advanced)]

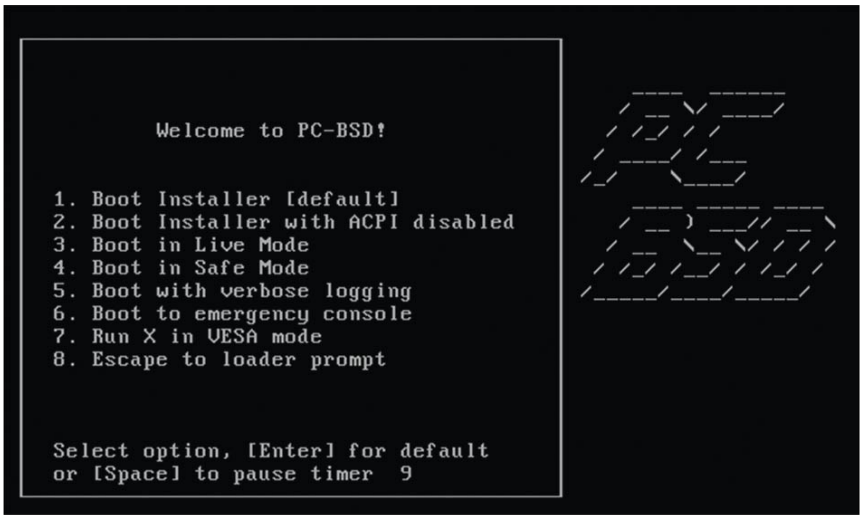
The following partitions will be created during the installation:

Slice	Mount	Size	Type
ad0	/	2048	UFS+S
ad0	SWAP	1998	SWAP
ad0	/var	1024	UFS+S
ad0	/usr	35885	UFS+S

Auto Setup

[✓] Install PC-BSD bootloader

Figure 3. PC-BSD remains close to the FreeBSD disk partitioning



```
Welcome to PC-BSD!

1. Boot Installer [default]
2. Boot Installer with ACPI disabled
3. Boot in Live Mode
4. Boot in Safe Mode
5. Boot with verbose logging
6. Boot to emergency console
7. Run X in VESA mode
8. Escape to loader prompt


Select option, [Enter] for default
or [Space] to pause timer  9
```

Figure 4. You can boot PC-BSD into a live environment

PBI's, themselves already an easy way to install software under PC-BSD.

Till now you had to go to the PBI website (*http://www.pbidir.org*) to locate and download the software you wanted. Downloading, double-clicking and following the steps in the wizards was the way to go. For casual users this is much simpler to do than installing software via packages or ports. The 8.0 release removes the need to go the PB website and allows for browsing available software via the new Software Manager.

The Software Manager holds three tabs, the first being the Software Browser. This is your window to the PBI website. If a PBI is available for your release a download icon is visible (Figure 5). When you click on it, you are asked whether you wish to install the program (Figure 6). If so, you are directed to the second tab, Installed Software, where you'll see the download in progress (Figure 7).

The Software Manager brings an *app store experience* to the desktop and that seems to be the way to go for the time being.

Other improvements

I guess most of us are familiar with Murphy's Law and thus we prepare for the time that our system doesn't work anymore. PC-BSD 8.0 offers a new tool for backup and restore, aptly named Life Preserver (*System›Life Preserver*). For now it is but a simple tool to create backups and send them to a remote server. You can't select specific folders (yet), but expect new features to be built upon the basic tool in the near future. (Figure 8)

As noted before, PC-BSD 8.0 brings improvements for both the general users and the more experienced/daring ones. Another example of this is the Ports Console. It runs in its own jail and allows you to install new software via packages and ports without inflicting harm on the rest of the system. This makes it a great sandbox to acquire FreeBSD-like skills. The downside of the portsjail (as it is called) is that you need to launch the programs from withing the jail. According to the mailing list something like:

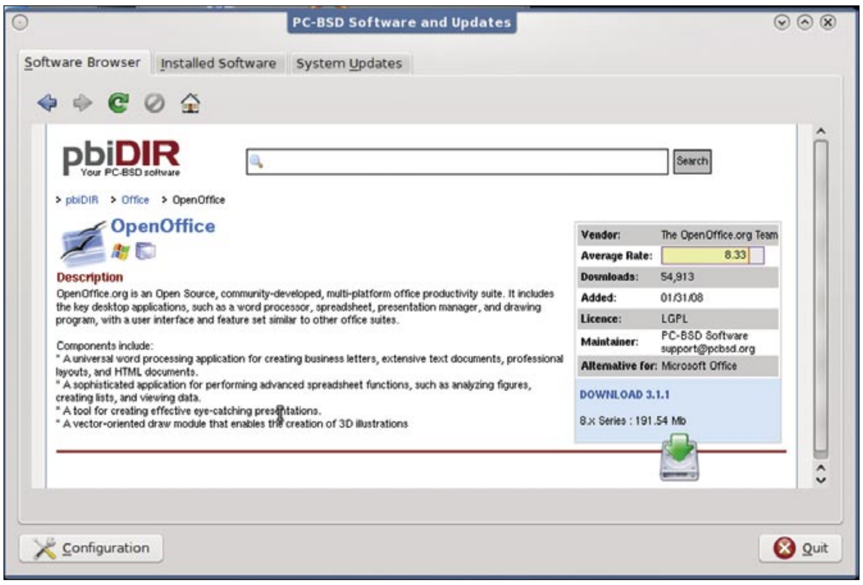
```
% portjail run /usr/local/bin/firefox
```

should do the trick when you wish to launch, for instance, Firefox (installed in the portsjail) from the *outside*.

End-user friendly easier said than done?

The focus of PC-BSD is to be an easy to install, easy to use and easy to maintain FreeBSD-based system. The next release is another step in that direction. It is no small task to create an operating system that is end-user friendly and in my opinion only a few open source desktops are fit to be used by companies and organizations. PC-BSD is one of them, the others being Ubuntu, SUSE Linux Enterprise Desktop. And – if Red Hat get it's head wrapped around it – Red Hat Enterprise Linux Desktop. This doesn't mean that all the other BSDs and Linux's are bad; heaven forbid, they serve a need for their userbase and if they have a strong enough userbase they will continue to exist and thrive.

Thus I was slightly amused when I heard of the TechRepublic article *Why the BSDs get no love* by Jack Waller (*http://blogs.techrepublic.com.com/opensource/?p=1123&tag=nl.e011*) and noticed the ensuing discussion on the FreeBSD advocacy mailing list. Jack's argument was that the BSDs lacked two things to attract the same attention as Linux: a graphical installer and live CD. Barring that, BSD would remain stuck in the 1990s. Well, he apparently didn't keep up with BSD Magazine, otherwise he wouldn't have written the article. Both PC-BSD and live BSDs have been discussed in various issues of this magazine.



OpenOffice

Description

OpenOffice.org is an Open Source, community-developed, multi-platform office productivity suite. It includes the key desktop applications, such as a word processor, spreadsheet, presentation manager, and drawing program, with a user interface and feature set similar to other office suites.

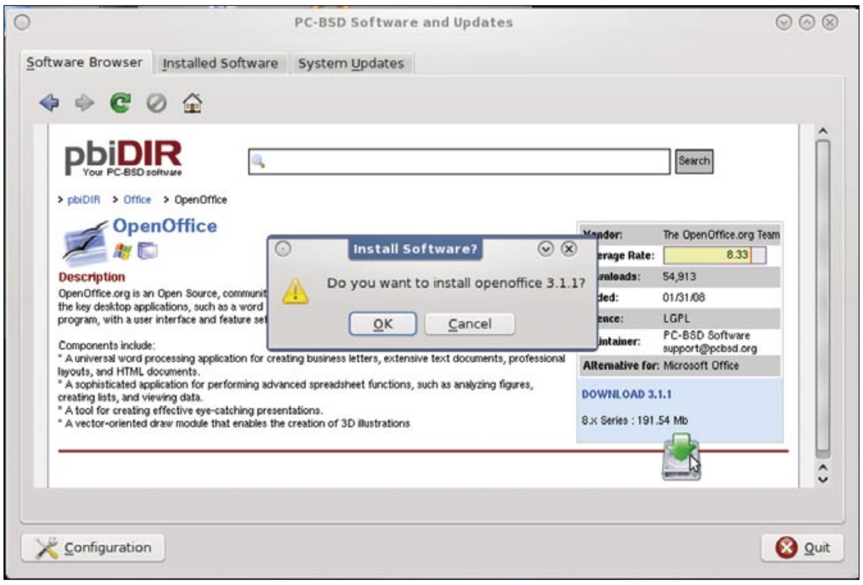
Components include:
* A universal word processing application for creating business letters, extensive text documents, professional layouts, and HTML documents.
* A sophisticated application for performing advanced spreadsheet functions, such as analyzing figures, creating lists, and viewing data.
* A tool for creating effective eye-catching presentations.
* A vector-oriented draw module that enables the creation of 3D illustrations

Vendor:	The OpenOffice.org Team
Average Rate:	8.33
Downloads:	54,913
Added:	01/31/08
Licence:	LGPL
Maintainer:	PC-BSD Software support@pcbsd.org
Alternative for:	Microsoft Office

DOWNLOAD 3.1.1

8.x Series : 191.54 Mb

Figure 5. The new Software Manager integrates the PBIdir website in your desktop



Do you want to install openoffice 3.1.1?

OpenOffice

Description

OpenOffice.org is an Open Source, communit
the key desktop applications, such as a word
program, with a user interface and feature se

Components include:
* A universal word processing application for creating business letters, extensive text documents, professional layouts, and HTML documents.
* A sophisticated application for performing advanced spreadsheet functions, such as analyzing figures, creating lists, and viewing data.
* A tool for creating effective eye-catching presentations.
* A vector-oriented draw module that enables the creation of 3D illustrations

ndor:	The OpenOffice.org Team
erage Rate:	8.33
wnloads:	54,913
ded:	01/31/08
ence:	LGPL
intainer:	PC-BSD Software support@pcbsd.org
Alternative for:	Microsoft Office

DOWNLOAD 3.1.1

8.x Series : 191.54 Mb

Figure 6. Installing new PBI's is easier than before

On the mailing list some people agreed with the statement that FreeBSD was lacking a graphical installer and pointed out that it would lower the threshold for new users to try their hand at FreeBSD. The opposite side of the discussion argued there was no need for a graphical installer and that the current sysinstall served its purpose. FreeBSD isn't used by desktop users only, which group is the target audience for a graphical installer. People working on embedded devices and preparing large scale deployments don't need a graphical installer and would only be bothered by it. Of course, there were more shades and nuances, so I suggest you go through the archives to read up on this discussion.

In relation to this article I'd like to repeat part of a post by Heidi Wyss. She explained how she struggled with installing FreeBSD 5.x some years ago. With the help of the FreeBSD Handbook and the man pages, coupled with strong motivation and perseverance she learned how to install and work with FreeBSD (*http://lists.freebsd.org/pipermail/freebsd-advocacy/2009-December/003966.html*). She ends her contribution with the following:

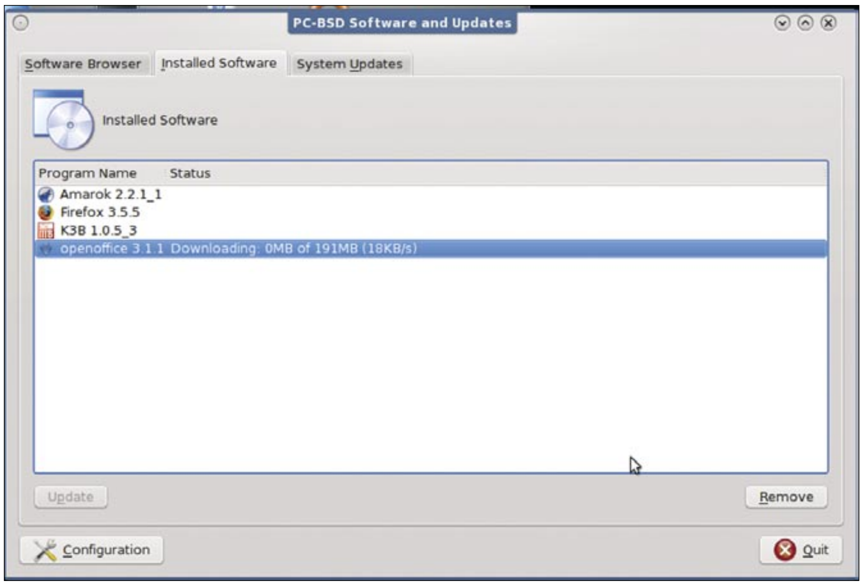
I'll end with this little tale, only to stir up thoughts: When I got the new versions of mplayer and vlc installed on FBSD8, I couldn't play most of my store-bought DVDs. Since I knew there had to be an easy fix, five minutes of searching on the Internet brought the easy fix (FreeBSD is so stable and reliable, once configured, I'd wholly forgotten about the CD/DVD device permissions), but how many so-called mainstream desktop users would get through that kind of glitch? Not many, however much someone like me, who's already quite delighted with FreeBSD, might wish otherwise.

I write most of my articles and books from the perspective of *regular* users, albeit the more adventurous ones among them. Moving to FreeBSD does indeed require digging into the in-itself-magnificent FreeBSD Handbook and a few trips to the man pages (pretty stern stuff). But understanding what the instructions mean requires a minimum level of expertise that most regular users simply don't have.

To make FreeBSD more palatable to end-users (or Debian for that matter) requires more than adding a graphical installer. Installing the operating system is simply the first hurdle and one that most users in a business environment never have to take. To achieve *ease of use* means looking at the desktop environment and offer tools that fit the skill-set of regular user. Ubuntu is doing this and it uses Debian as a strong foundation. PC-BSD is doing it as well, on the solid FreeBSD bedrock. The next release, PC-BSD 8.0, brings with it new elements to cater to the needs and wants of desktop users. The project can do that because of that singular desktop focus.

More help is on the way

Around the time that PC-BSD 8.0 hits the servers you might also see some new books about PC-BSD in the bookstores. Dru Lavigne on the one hand and Jeremy Reed and yours truly on the other hand are busy writing books on how to work with PC-BSD. This is an indication of the level of confidence publishers have in this operating system. If experience is an indicator both books will also help in getting new users to move to the PC-BSD desktop.



```
Program Name    Status
Amarok 2.2.1_1
Firefox 3.5.5
K3B 1.0.5_3
openoffice 3.1.1 Downloading: 0MB of 191MB (18KB/s)
```

Figure 7. New PBI's are downloaded in the background

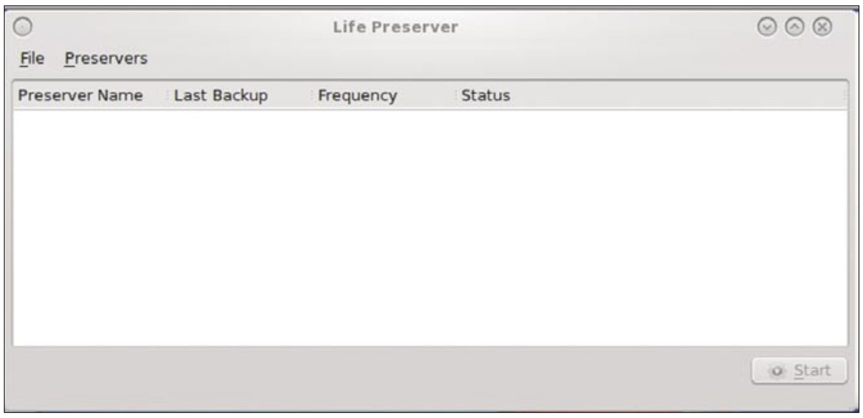


Figure 8. Life Preserver is the first step towards a better backup solution for PC-BSD

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.

Installing and securing an Apache Jail with SSL on FreeBSD

Rob Somerville

The Apache HTTP server developed by the Apache Software Foundation [1] is the most popular webserver software in use today installed at over 100 Million [2] sites worldwide.

Renown for stability, flexibility and speed, Apache is the web-server of choice in demanding environments. This article will walk you through performing some basic security measures, performing an Apache installation in a secure FreeBSD jail, generating an SSL certificate and setting up a password protected area on the server.

Security in a production environment

Whilst Apache is more secure than many applications, any software is vulnerable to attack so it is prudent to take as many steps as possible to minimise the risk. It is advisable that the systems administrator ensures that only the required services are running on the server, user accounts are locked down and consider raising the security level of the kernel if appropriate. With a standard software install, if an intruder managed to gain root access via an exploit all applications and the host server environment itself would be open to abuse. Running an application in a jail restricts any troublemaker to the jail environment – see Figure 1.

This approach has a major advantage in that an attacker cannot gain access to the host file-system. However, it takes more configuration as Apache SSL etc. cannot access applications on the host.

Post-install, whether a jail is used or not it is important that the administrator takes steps to audit changes on the system and prevent insecure ports etc. being installed – this can be achieved by installing utilities such as `tripwire` and `portaudit`. See `man security` for further details.

For full functionality, both domain names will need to be added to your own DNS server or workstation hosts file. Changes to the IP address of the servers above will need appropriate changes to the netmask, routing entries, hosts file etc.

Prerequisites

A running test FreeBSD 7.2 installation with access to the internet will be required. It is recommended that a *Custom* clean install is performed comprising of the base, kernels,

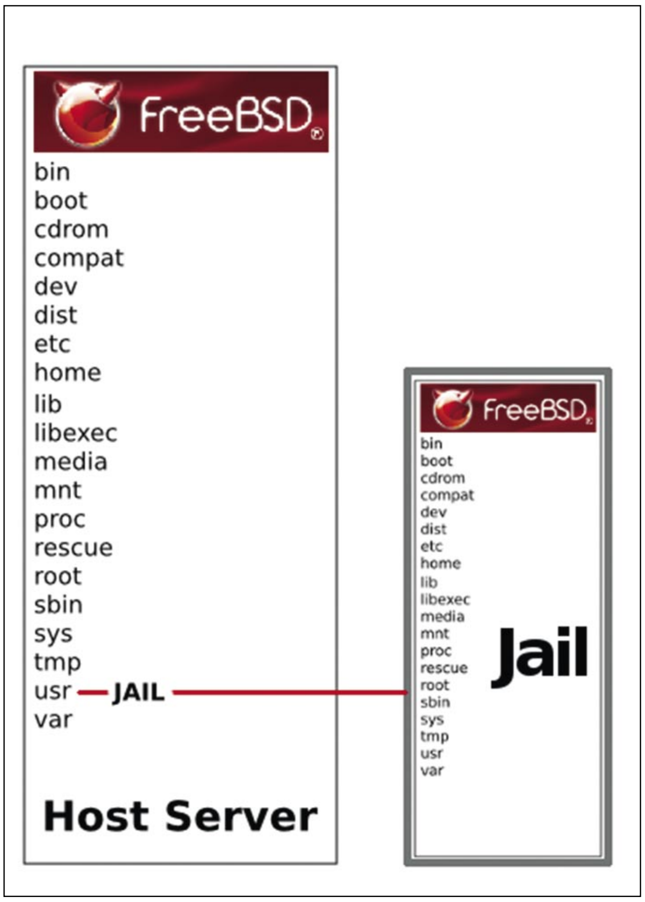


Figure 1. Jail filesystem architecture

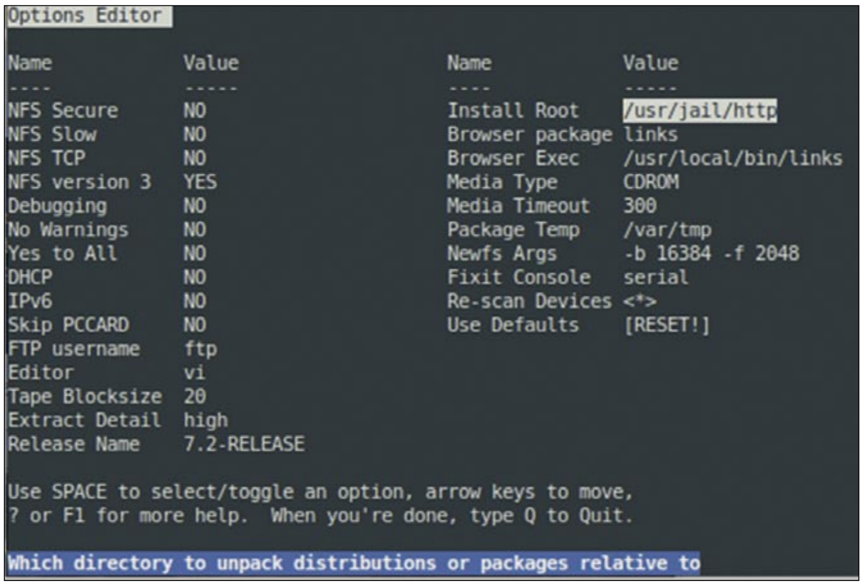
man and kernel sources to limit server bloat. The server should have NFS, Linux compatibility and use as a network gateway disabled. If desired, a separate partition could be used for the jail and the Apache content. The SSH daemon may be run on the host for remote access (on a non-standard port using the Version 2 protocol as Version 1 has security issues) if desired.

Caution should be exercised particularly running the patching and lockdown commands on a production server as these changes may cause adverse effects or result in unexpected results in other scripts or applications.

It is recommended that these changes are not applied in a live environment without an adequate testing regime in place with a non-production server.

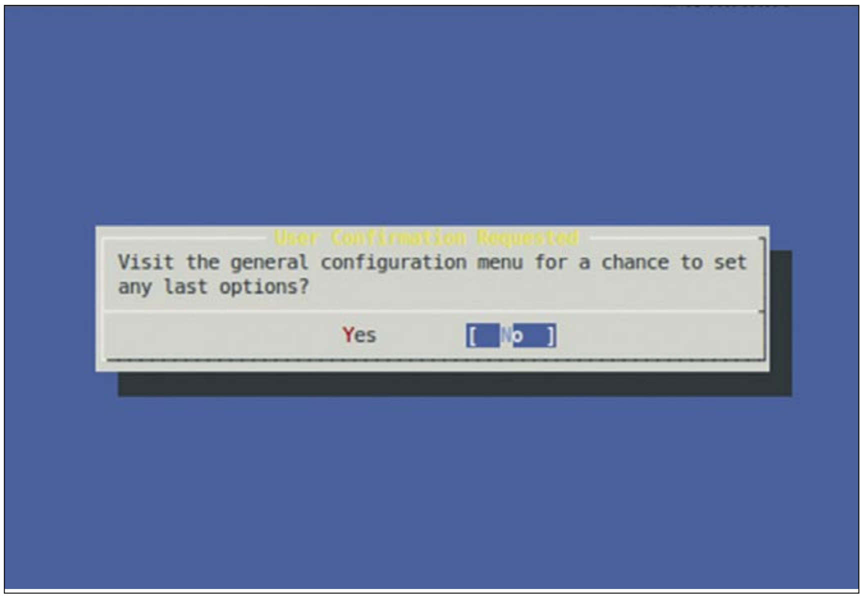
Table 1. Default IP addresses used

Value	Comments
192.168.0.110	IP address if host system – replace as appropriate
192.168.0.111	IP address of jail system – replace as appropriate
jailbird.merville.intranet	Domain name of host server – replace as appropriate
apache.merville.intranet	Domain name of jailed server – replace as appropriate



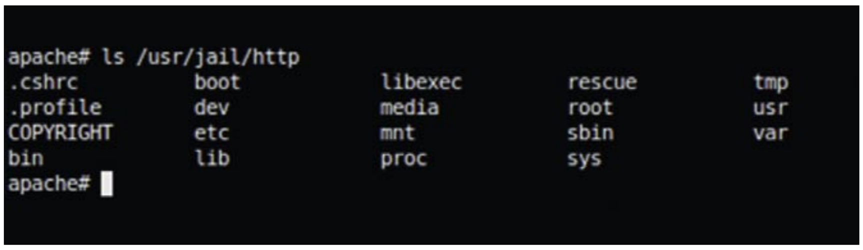
```
Options Editor

Name            Value                   Name            Value
----            -----                   ----            -----
NFS Secure      NO                      Install Root    /usr/jail/http
NFS Slow        NO                      Browser package links
NFS TCP         NO                      Browser Exec    /usr/local/bin/links
NFS version 3   YES                     Media Type      CDROM
Debugging       NO                      Media Timeout   300
No Warnings     NO                      Package Temp    /var/tmp
Yes to All      NO                      Newfs Args      -b 16384 -f 2048
DHCP            NO                      Fixit Console   serial
IPv6            NO                      Re-scan Devices <*>
Skip PCCARD     NO                      Use Defaults    [RESET!]
FTP username    ftp
Editor          vi
Tape Blocksize  20
Extract Detail  high
Release Name    7.2-RELEASE

Use SPACE to select/toggle an option, arrow keys to move,
? or F1 for more help.  When you're done, type Q to Quit.

Which directory to unpack distributions or packages relative to
```

Figure 2. Change Release Name and Install Root



```
User Confirmation Requested
Visit the general configuration menu for a chance to set
any last options?

            Yes            [ No ]
```

Figure 3. Ensure you choose NO here



```
apache# ls /usr/jail/http
.cshrc          boot            libexec         rescue          tmp
.profile        dev             media           root            usr
COPYRIGHT       etc             mnt             sbin            var
bin             lib             proc            sys
apache# █
```

Figure 4. Jail file structure tree [Identical to Figure 1]

Initial patching and lockdown

Please refer to the FreeBSD [3] website and *bsdguides.org* [4] for further information, in particular the Hardening FreeBSD section.

- Run `freebsd-update fetch` and `freebsd-update install` to patch the binaries to the latest revision.
- Disable root access to the consoles:
  - `cp /etc/ttys /etc/ttys.000`
  - `sed 's/on  secure/on  insecure /g' /etc/ttys.000 > /etc/ttys`

There are two spaces between *on* and *secure* or *insecure* Ensure you have a user who can su to root on the system [or can ssh in] as you will not be able to login via a console as root!

Remove the backup ttys.000 file once you are satisfied that the system performs as expected. Alternatively, edit the ttys file using your favourite editor to change *secure* to *insecure* where appropriate.

- `Reboot`
- Check you can login as a standard user, then su to root
- Edit the `/etc/login.conf` file and replace

```
  passwd_format=md5 with passwd_
format=blf
```

This will change the default password encryption algorithm from md5 to blowfish which is more secure.

- Remove the line marked `:umask=022` and add the following in its place:

```
:umask=027:
:mixpasswordcase=true:\
:minpasswordlen=8:\
:passwordtime=30d:\
     :idletime=30:\
```

This will tighten up password security and change the default rights mask so that directories are created with 0750 permissions

- Run `cap_mkdb /etc/login.conf` to rebuild the login database
- Run the following to restrict access to cron to root only:

```
echo "root" > /var/cron/allow
echo "root" > /var/at/at.allow
```

- Stop unauthorised users viewing critical system files:

```
chmod o= /etc/fstab
chmod o= /etc/ftpusers
chmod o= /etc/group
chmod o= /etc/hosts
chmod o= /etc/hosts.allow
chmod o= /etc/hosts.equiv
chmod o= /etc/hosts.lpd
chmod o= /etc/inetd.conf
chmod o= /etc/login.access
chmod o= /etc/login.conf
chmod o= /etc/newsyslog.conf
chmod o= /etc/rc.conf
chmod o= /etc/ssh/sshd_config
chmod o= /etc/sysctl.conf
chmod o= /etc/syslog.conf
chmod o= /etc/ttys
```

- Restrict execution of programs to root and the wheel group only:

```
chmod o= /etc/crontab
chmod o= /usr/bin/crontab
chmod o= /usr/bin/at
chmod o= /usr/bin/atq
chmod o= /usr/bin/atrm
chmod o= /usr/bin/batch
chmod o= /usr/bin/users
chmod o= /usr/bin/w
chmod o= /usr/bin/who
chmod o= /usr/bin/lastcomm
chmod o= /usr/sbin/jls
chmod o= /usr/bin/last
chmod o= /usr/sbin/lastlogin
chmod o= /usr/bin/top
chmod o= /bin/ps
chmod o= /usr/bin/man
```

This list can be extended to other files, such as nmap as appropriate

- These daemons should be disabled for security reasons:

```
chmod ugo= /usr/bin/rlogin
chmod ugo= /usr/bin/rsh
```

- If desired, the logfile directory can be secured to prevent hackers flushing the log files or viewing them.

This will mean that certain applications will no longer able to manipulate files in the `/var/log` directory – e.g. logrotate

Listing 1. rc.conf for jailbird.merville.intranet

```
# Keyboard settings

keymap="uk.iso"

# IP settings

hostname="jailbird.merville.intranet"
ifconfig_em0="inet 192.168.0.110 netmask 255.255.255.0"
ifconfig_em0_alias_0="inet 192.168.0.111 netmask 255.255.255.255"
defaultrouter="192.168.0.254"

# Services enabled

moused_enable="NO"
inetd_enable="NO"
clear_tmp_enable="YES"
rpcbind_enable="NO"
sendmail_enable="NONE"
syslogd_enable="YES"
sshd_enable="YES"

# Syslog settings

syslogd_flags="-ss"
syslogd_flags="-a 192.168.0.110 "
syslogd_flags="-a 192.168.0.111"

# Jail settings

jail_set_hostname_allow="NO"
jail_enable="YES"
jail_list="http"
jail_interface="em0"
jail_devfs_enable="YES"
jail_procfs_enable="YES"

# Per jail settings

jail_http_rootdir="/usr/jail/http"
jail_http_hostname="apache.merville.intranet"
jail_http_ip="192.168.0.111"
jail_http_devfs_ruleset="devfsrules_jail"
```

```
chmod o= /var/log/
chflags sappnd /var/log
chflags sappnd /var/log/*
```

Building the jail

1. Ensure you have the FreeBSD Install CD loaded
2. `sysinstall`

- Select Custom Installation
- Select View/Set various installation options
- Change Install Root to `/usr/jail/http`
- Change Release Version to the pre-upgrade release [see figure 2]
- Press [q]
- Select Distributions
- Select Custom
  Select:
  - Base
  - Kernels›GENERIC
- Exit out of the menu to Custom Installation Options
- Select Media
- Select CD/DVD
- Select Commit

The installation will commence

When prompted to visit the general configuration menu choose No otherwise you will change the host system's settings (see Figure 3)

Exit from sysinstall

```
ls /usr/jail/http
```

This should display a minimal FreeBSD install tree (Figure 4)

3. Edit your `/etc/rc.conf` to reflect the following. Amend IP address, interface name hostname, router etc. to suit your configuration. The version I have used is shown Listing 1.
4. Edit `/usr/jail/http/etc/rc.conf` (see Listing 2)
5. I have secured SSH so it runs on port 10022. Edit `/usr/jail/http/etc/ssh/sshd_config` to force SSH to use the jails interface: see Listing 3.
6. cp `/etc/resolv.conf /usr/jail/http/etc/resolv.conf`

   Copy `resolv.conf` across so that the jailed server can resolve hostnames
7. `sh /etc/rc`

   Remove CDROM and Restart
8. `jls`

```
ping -c3 192.168.0.111
```

Jail should now be running (Figure 5)

9. `jexec 1 touch /etc/fstab`

```
jexec 1 passwd
jexec 1 adduser
jexec 1 tzsetup
```

Create a vanilla fstab for the jail, set the root password, add a default user (ensure they are invited to be a member of the wheel group) and set the timezone

10. Stop and start the jail:

```
/etc/rc.d/jail stop http
/etc/rc.d/jail start http
```

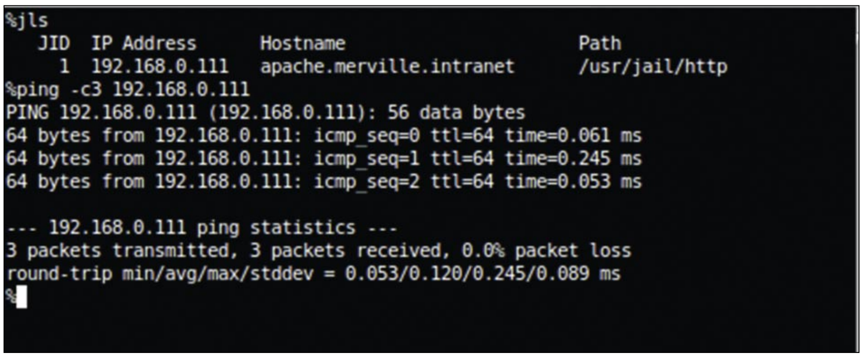
11. `sysctl security.jail.allow_raw_sockets=1` Temporarily allow the jail outgoing access to the internet

Listing 2. rc.conf for apache.merville.intranet

```
# Keyboard settings

keymap="uk.iso"

# IP settings

hostname="apache.merville.intranet"
ifconfig_em0="inet 192.168.0.111 netmask 255.255.255.0"
defaultrouter="192.168.0.254"
early_late_divider="NETWORKING"

# Services enabled

moused_enable="NO"
inetd_enable="NO"
clear_tmp_enable="YES"
rpcbind_enable="NO"
sendmail_enable="NONE"
sshd_enable="YES"
```

Listing 3. sshd_config for apache.merville.intranet

```
Port 10022
Protocol 2
ListenAddress 192.168.0.111
LoginGraceTime 1m
PermitRootLogin no
StrictModes yes
MaxAuthTries 3
MaxSessions 3
Subsystem       sftp    /usr/libexec/sftp-server
```



```
%jls
   JID  IP Address      Hostname                      Path
     1  192.168.0.111   apache.merville.intranet      /usr/jail/http
%ping -c3 192.168.0.111
PING 192.168.0.111 (192.168.0.111): 56 data bytes
64 bytes from 192.168.0.111: icmp_seq=0 ttl=64 time=0.061 ms
64 bytes from 192.168.0.111: icmp_seq=1 ttl=64 time=0.245 ms
64 bytes from 192.168.0.111: icmp_seq=2 ttl=64 time=0.053 ms

--- 192.168.0.111 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 0.053/0.120/0.245/0.089 ms
%
```

Figure 5. Jail up and running

12. `ssh user@192.168.0.111 -p10022`

Now ssh into the jail where the username/password combination is what you have configured in step 9 above

13. `su`

Enter the root password you configured in step 9 above.

Installing Apache [5]

Download and install the package, set it to run on jail start

1.
```
pkg_add -r apache22

echo "apache22_enable="YES"" >> /etc/rc.conf

rehash
```

2. Edit the `/etc/hosts` file in the jail and add the following to allow Apache to resolve the hostname

```
   192.168.0.111   apache apache.merville.intranet
```

3. `/usr/local/etc/rc.d/apache22 start`

Installing and configuring OpenSSL 6 and the SSL certificates

Rather than using a vendor provided certificate which is expensive, we will self-sign the certificate. The browser will interpret this as a potential security threat so this may not be suitable for a production environment. If you are using a 3rd party certificate, please consult the provided documentation on how to install this with Apache. Here we will configure OpenSSL. create the the directories for the CSR and server key, and generate the key and certificate.

Common Name should either be 192.168.0.111 or apache.merville.intranet if you intend to add the server to DNS.

1. See Listing 4.

You will now be prompted to enter a password – this will be needed in the next step

2. See Listing 5.

3. edit `/usr/local/etc/apache22/Includes/httpd-ssl.conf`

Change *ServerName http://www.example.com:443* to:

```
    192.168.0.111:443
```

(or `apache.merville.intranet` if using DNS)

Change SSLCertificateFile to point to:

```
   /usr/local/etc/apache22/ssl.crt/server.crt
```

Change SSLCertificateKeyFile to point to:

```
/usr/local/etc/apache22/ssl.key/server.key
```

If desired, remove the password from the certificate. If you do not do this, the server will hang waiting for the certificate password at *Starting Jails:* when the server reboots.

You can type the password at the console and the jail will resume, but no other process will prior to this (e.g. console login)

4.
```
cd /usr/local/etc/apache22/ssl.key

cp server.key server.key.000
openssl rsa -in server.key.000 -out server.key
```

Listing 4. Installing OpenSSL

```
pkg_add -r openssl
rehash
cp /usr/local/openssl/openssl.cnf.sample /usr/local/openssl/openssl.cnf
mkdir /usr/local/etc/apache22/ssl.key
mkdir /usr/local/etc/apache22/ssl.crt
chmod 0700 /usr/local/etc/apache22/ssl.key
chmod 0700 /usr/local/etc/apache22/ssl.crt
cd /root
openssl genrsa -des3 -out server.key 1024
```

Listing 5. Generating the SSL certificate

```
openssl req -new -key server.key -out server.csr
openssl x509 -req -days 365 -in /root/server.csr -signkey /root/server.key
-out /root/server.crt
cp /root/server.key /usr/local/etc/apache22/ssl.key/
cp /root/server.crt /usr/local/etc/apache22/ssl.crt/
chmod 0400 /usr/local/etc/apache22/ssl.key/server.key
chmod 0400 /usr/local/etc/apache22/ssl.crt/server.crt
cd /usr/local/etc/apache22/extra
cp httpd-ssl.conf /usr/local/etc/apache22/Includes
```

Securing and configuring Apache

In these steps we will create a secure area and password protect it using an encrypted password stored in the `.htpasswd` file.

1.
```
mkdir /usr/local/www/apache22/data/secure

htpasswd -c /usr/local/etc/apache22/.htpasswd apachelogin
```

edit `/usr/local/etc/apache22/Includes/www_secure.conf`

and add the following: see Listing 6.

This will lockdown the secure area so that the username apachelogin and the password you entered in step 1 will be required. If no content is found, access will be forbidden and the visitor will not be able to browse the directory tree.

edit `/usr/local/etc/apache22/Includes/default.conf` and add the following: see Listing 7.

This prevents Apache from announcing the version number it runs and locks down all other directories apart from the `../data` directory.

Now, copy a test page across to the secure area:

```
   cp /usr/local/www/apache22/data/index.html      /usr/local/www/apache22/data/secure/secure.html
```

2. Change the kernel securelevel

Add this to `/etc/rc.conf`:

```
   kern_securelevel="3"
   kern_securelevel_enable="YES"
```

Exit from the SSH jail session and restart:

```
exit
reboot
```

Listing 6. Additions to /usr/local/etc/apache22/Includes/www_secure.conf

```
<Directory /usr/local/www/apache22/data/secure/>

   AuthType Basic

   AuthName "Apache secure area"

   AuthUserFile /usr/local/etc/apache22/.htpasswd

   Require User apachelogin

   Allow from All

   Order Allow,Deny

   Options None
</Directory>
```

Listing 7. Additions to /usr/local/etc/apache22/Includes/default.conf

```
ServerSignature Off

ServerTokens Prod



<Directory />

  Order Deny,Allow

    Deny from all

  Options None

     AllowOverride None

</Directory>



<Directory /usr/local/www/apache22/data/>

  Order Allow,Deny

  Allow from all

  Options None

</Directory>
```

Testing it out

Use a browser from another computer on your network, If you enter *http://192.168.0.111* into your browser, the Apache home page should be shown.

http://192.168.0.111/secure/ should be forbidden after logging on as no *index.html* is present.

http://192.168.0.111/secure/secure.html should display *It Works* as we have copied the index file into the secure area. Repeat for *https://* and after accepting the certificate, the same results should occur.

A portscan using nmap or equivalent should be run against both IP addresses to ensure no services have slipped through the net.

Further reading and references

- *http://www.apache.org* – The Apache Software Foundation [1]
- *http://news.netcraft.com/archives/web_server_survey.html* – Netcraft [2]
- *http://www.freebsd.org* – FreeBSD [3]
- *http://www.bsdguides.org* – BSD guides [4]
- *http://httpd.apache.org* – Apache reference [5]

Further actions

- Harden Apache and tighten IP access, modules loaded etc.
- Install and tune a custom kernel rather than using GENERIC. Remove redundant kernel modules as appropriate. Use nextboot to check that the kernel works OK.
- Install portaudit which checks ports prior to installation against the on-line ports vulnerability database. Add a binary file auditing tool such as tripwire.
- Configure a firewall on the server and the jail.

About the Author

Rob Somerville has a keen passion for all things BSD / Open Source and has been working with technology since the early Eighties. His biggest claim to fame was designing an on-line search engine for a database company when 2400 Baud modems were cutting-edge. Married with 1 daughter, he shares the house with many computers, 2 cats, a dog and an extensive collection of O'Reilly books.

The gemstones for FreeBSD

Marko Milenovic

Building web applications has become so popular that you can't imagine Internet as a static system any more.

Anything and everything is web application that interacts with users and vice versa. In the age of web oriented programming languages and web applications one precious stone has been found in the mystical Japan. It brings zen philosophy to a modern programming language.

Why Ruby?

Ruby may be described as an absolutely pure object-oriented scripting language and a genuine attempt to combine the best of everything in the scripting world. It was written in C back in the nineties by Yukihiro Matsumoto (aka *matz*) and it combines syntax inspired by Perl with Smalltalk-like features. The main goal was *principle of least surprise*. Matz wanted to create a language that will help programmers *express themselves* and not fight it. And he succeded. Try it out and see for yourself.

Ok, what's with Ruby on Rails?

Web applications usually work on top of some framework. Frameworks make development easy and fast. And Ruby wouldn't have become so popular on the web if Rails didn't come in as a boost. It was developed by David Heinemeier Hansson as a part of his work on Basecamp, a project management tool. Rails uses the *Model-View-Controller* (MVC) architecture pattern to organize application programming. It puts a powerful tool into web developers hands – a tool that brings quick and high quality development.

Why Lighttpd?

This is how developers of this great web server describe their product:

lighttpd is a secure, fast, compliant, and very flexible web-server that has been optimized for high-performance environments. It has a very low memory footprint compared to other webservers and takes care of cpu-load. Its advanced feature-set (FastCGI, CGI, Auth, Output-Compression, URL-Rewriting and many more) make lighttpd the perfect webserver-software for every server that suffers load problems.

You might think of Lighttpd as a lightweight web server with limited capabilities and you would be wrong. It's true, lighttpd is light and not just by it's name. It's very nice towards server when it comes to memory and CPU consuming. Yet, it lacks no advanced features that are present in other popular web serves such as Apache. And it works like a charm with Ruby on Rails.

Why FreeBSD?

Though this may not need an answer let's just give a few reasons why FreeBSD. FreeBSD is operating system designed with the idea that may be summed up in: The power to serve. It has shown it's power on some very busy systems that take high load such as Yahoo! Easy to setup, very easy to maintain and it works so good under great pressure.

Putting it all together

Let's put it all together and see what happens. First of all, we need Ruby installed. In order to do that use the following:

```
#cd /usr/ports/lang/ruby18 && make install clean
```

This will install all the essential parts of Ruby language. You are now ready to use Ruby on your FreeBSD machine as a substitute for shell scripts or Perl. Ruby has become a very popular substitute for Perl since it gives sysadmins a great tool for everyday tasks.

Let us move on. We got Ruby installed but we want more. Before moving to Rails there is another Ruby tool that we need to explore – gems. Gem is packaged Ruby application or a library. In order to work with gems we use command gem that comes with Ruby installation. Let's see what gem does: see Listing 1.

So, all we need in order to work with gems is this neat tool. Here is what gem does:

- Easy Installation and removal of RubyGems packages and their dependents;
- Management and control of local packages;
- Package dependency management;
- Query, search and list local and remote packages;
- Multiple version support for installed packages;
- much more...

So lets add Rails and a few other packages:

```
#gem install rails rails-app-installer
```

This will install Rails and all needed libraries. Once it's done check whether everything is in its place with:

- #gem list --local
- actionmailer (2.3.4)
- actionpack (2.3.4)
- activerecord (2.3.4)
- activeresource (2.3.4)
- activesupport (2.3.4)
- cgi_multipart_eof_fix (2.5.0)
- daemons (1.0.10)
- fastthread (1.0.7)
- gem_plugin (0.2.3)
- mongrel (1.1.5)
- mongrel_cluster (1.0.5)
- rack (1.0.0)
- rails (2.3.4)
- rails-app-installer (0.2.0)
- rake (0.8.7)
- sources (0.0.2)
- sqlite3-ruby (1.2.4)

Great! It all worked well.

Light it up

Ruby on Rails comes with web server called Mongrel. You may have seen it on the list of installed gems – *mongrel (1.1.5)*. Mongrel may work as a stand-alone server. It is capable of serving Ruby on Rails powered sites without requiring any other web servers. From the base directory of any typical Rails application, simply run:

```
#mongrel_rails start -p 80 -e
production -d
```

This will start an application on port 80 (`-p`) using production settings (`-e`) like a daemon in the background (`-d`).

This may be just fine for local development or if you are not running any other web sites on server. If you are then a solution must be found. And sometimes you simply want your application to be run by a fully featured web server. Lighttpd to the rescue!

Setting up Lighttpd is pretty easy on FreeBSD. First of all it needs to be installed:

```
#cd /usr/ports/www/lighttpd && make
install clean
```

Installation process will ask for some tweaks to be selected. You may always recompile Lighttpd and add more features if needed.

Once Lighttpd is installed two things need to be done. First of all we need to modify configuration file that may be found in `/usr/local/etc/lighttpd.conf`. You may notice that configuration *language* is pretty easy to understand. Here is sample configuration script stripped down to bare minimum. Users should play with various combinations and adapt this script to their needs (see Listing 2).

This configuration file is pretty easy to understand without any additional explanations. Notice that `server.modules` block has been stripped down to minimum required to run Lighttpd with RoR application. mimetype.assign has also been cut because of it's length. Take a look at the original `lighttpd.conf` file that comes with the installation for the whole list. It is suggested to use include option to keep this configuration file in better shape. So moving big blocks of configuration to separate files and linking them through *include* option may be a good idea.

Now you need to create directory where default web server will point to. In this case it's `/home/web/`. And add some simple HTML content there – you don't want your default page empty, do you? Lighttpd won't create log files on it's own so we need to do this:

```
#cd /var/log && touch
lighttpd.error.log lighttpd.access.log
```

Now add `lighttpd_enable="YES"` to `/etc/rc.conf` and light it up:

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

Visit *http://mydomain.org* and be amazed with your web site. If Lighttpd didn't fire up for some reason check `/var/log/messages` and `/var/log/lighttpd.error.log` to see the reason why. Sometimes log files need to be owned by the user that is running web server – in this case www. Log files will tell you what went wrong.

Put a Ruby in it

Great, our Lighttpd installation worked just fine. It's time to add some Ruby stones to it. As you may have noticed the last line of Lighttpd configuration files looked like this:

```
#include "rails_app.conf"
```

It's time to create rails_app.conf and comment this line off. Using your favorite text editor open this file:

Listing 1. gem usage

```
#gem
Usage:
    gem -h/--help
    gem -v/--version
    gem command [arguments...] [options...]

Examples:
    gem install rake
    gem list --local
    gem build package.gemspec
    gem help install
```

Listing 2. Lighttpd configuration

```
## modules to load
server.modules              = (
                               "mod_rewrite",
                               "mod_access",
                               "mod_fastcgi",
                               "mod_simple_vhost",
                               "mod_accesslog" )

## a static document-root, for virtual-hosting take
look at the
## server.virtual-* options
server.document-root        = "/home/web/"

## where to send error-messages to
server.errorlog             = "/var/log/
lighttpd.error.log"

# files to check for if .../ is requested
index-file.names            = ( "index.html",
"index.php",
                               "index.htm",
"default.htm" )

## set the event-handler (read the performance section
in the manual)
server.event-handler = "freebsd-kqueue" # needed on
OS X

# mimetype mapping
mimetype.assign             = (
  ".pdf"          =>      "application/pdf",
  ".sig"          =>      "application/pgp-signature",
  ".spl"          =>      "application/futuresplash",
  ".class"        =>      "application/octet-stream",
  ".ps"           =>      "application/postscript",

…take a look at lighttpd.conf file for the whole
list...

  ".wmv"          =>      "video/x-ms-wmv",
  ".bz2"          =>      "application/x-bzip",
  ".tbz"          =>      "application/x-bzip-
compressed-tar",
  ".tar.bz2"      =>      "application/x-bzip-
compressed-tar"
 )

## send a different Server: header
## be nice and keep it at lighttpd
server.tag                  = "lighttpd"

#### accesslog module
accesslog.filename          = "/var/log/
lighttpd.access.log"

## deny access the file-extensions
#
# ~    is for backupfiles from vi, emacs, joe, ...
# .inc is often used for code includes which should in
general not be part
#      of the document-root
url.access-deny             = ( "~", ".inc" )

$HTTP["url"] =~ "\.pdf$" {
  server.range-requests = "disable"
}

# which extensions should not be handle via static-file
transfer
# .php, .pl, .fcgi are most often handled by mod_
fastcgi or mod_cgi
static-file.exclude-extensions = ( ".php", ".pl",
".fcgi" )

######## Options that are good to be but not
neccesary to be changed #######

## bind to port (default: 80)
server.port                = 80

## bind to localhost (default: all interfaces)
server.bind                = "mydomain.org"

server.pid-file            = "/var/run/lighttpd.pid"

## virtual directory listings
dir-listing.activate       = "enable"

### only root can use these options
#
# chroot() to directory (default: no chroot() )
server.chroot              = "/"

## change uid to <uid> (default: don't care)
server.username            = "www"

## change uid to <uid> (default: don't care)
server.groupname           = "www"

#### include
#include "rails_app.conf"
```

```
#vim /usr/local/etc/rails_app.conf
```

And add the following lines to it: see Listing 3.

So what does this configuration file do? It tells our Lighttpd web server that we want to run Ruby on Rails application using FastCGI system. It states where our Rails application resides (`/home/blog/public`) and it creates some server related tweaks such as number of processes to run. These tweaks should be adopted to server configuration. Simply, use these setting and monitor how your server works with them. If you are not satisfied with the performance simply tweak them till you reach the desired performance.

Give me a blog to build a dream on

Wait, but we don't have an application on that location. How is Lighttpd supposed to work with no application in `/home/blog/public`? Well, purpose of this text is not to teach you how to write Ruby and Ruby on Rails code but how to run it on FreeBSD using Lighttpd. In order to do that we'll just have to use some existing application. This will be a good chance to learn how to play with gem tool. And what better web application to use for demonstration than blog software! For this purpose we're going to intall a great blogging software called Typo. The easiest way to do this is using gem:

```
#gem install typo
```

It is a part of your gems collection now. You may notice that Typo will pull down older version of Rails (2.2.2). This is where the power of gems comes to stage. You are able to run different versions of various gems and not cause anything to crash.

By deafult Typo uses MySQL database. Installing MySQL is beyond the scope of this article. We'll presume that you have MySQL running and that you have added proper database and user for it.

Our next step is to add Ruby MySQL driver:

```
# gem install mysql
```

Check out your local gems collection and both mysql and typo should be on the list. Once you install Typo into your gems collection it becomes an executable like any other on the system. All you have to do now is extract its content to some location. And this is how Typo is installed:

```
#typo install /home/blog/ db_
user=$user db_name=$database db_
host=localhost db_password=$pass
```

Listing 3. Rails application in Lighttpd

```
$HTTP["host"] =~ "blog.mydomain.org" {
  server.document-root = "/home/blog/public"
  server.errorlog = "/var/log/blog.error.log"
  accesslog.filename = "/var/log/blog.access.log"
  url.rewrite = ( "^/$" => "index.html", "^([^.]+)$" => "$1.html" )
  server.error-handler-404 = "/dispatch.fcgi"

  fastcgi.server = ( ".fcgi" =>
    ( "localhost" =>
      (
        "socket" => "/tmp/ruby-railsapp.fastcgi",
        "bin-path" => "/home/blog/public/dispatch.fcgi",
        "bin-environment" => ( "RAILS_ENV" => "production" ),
        "min-procs" => 5,
        "max-procs" => 5,
        "idle-timeout" => 60
      )
    )
  )
}
```

This will extract Typo into `/home/blog` where we wanted to run our application at first place. It will use database coordinates we have already prepared for it. If you don't want to use MySQL there are two other options. One is PostgreSQL and the other is SQLite. In both cases you'll need proper database setup and Ruby drivers (postgresql and sqlite3) installed.

Typo will start an instance of its installation on port 7000. If you wish you may use this for testing purposes. We do not want this since we've setup Lighttpd to serve our blog. So simply kill this process. We're all ready to test everything. Make sure that the last line of lighttpd.conf file is not commented since it will include Rails application setup. Then simply restart your Lighttpd server:

```
#/usr/local/etc/rc.d/lighttpd restart
```

If no errors were present when you tweaked config files you may point your browser to *http://blog.mydomain.org* and you'll see a fresh Typo installation. Voila! Your blog is ready for usage.

Conclusion?

We had the power
We had the space
We had a sense of time and place
We knew the words
We knew the score
We knew what we were fighting for...

Ending this article with lyrics from Adrenochrome performed by the Sisters of Mercy (playing in the background as this article is being written). Combination of Ruby, Rails, Lighttpd and FreeBSD puts a great power in our hands. Power whose limits are set by the imagination of us – users. So, there is no real conclusion to this story. It is just a starting point for all curious readers of this magazine.

On the 'Net

- *http://www.freebsd.org/*
- *http://www.lighttpd.net*
- *http://www.ruby-lang.org/en/*
- *http://rubyonrails.org/*
- *http://wiki.github.com/fdv/typo/*

OpenBSD, NetBSD and FreeBSD as file sharing servers – Part 1 – NFS

Petr Topiarz

How to share files between multiple operation systems and keep your data safe.

Why BSD as a file sharing server?

This article came to being as a result of my experience with using both OpenBSD and NetBSD as file servers. In fact, the 5 computers I care about are used as routers and database servers as well, however I want to look at them from the perspective of file sharing. The important reason why I would truly recommend BSD is that during the last 5 years I have run the servers, despite many electricity breakdowns, I have never lost a single byte on either the OpenBSD or NetBSD served machines. The most destructive case occurred some two years ago, when electricity went down seven times in a short period and we had no UPS to keep the machines alive or give us time to turn them off safely. The problem was that while all computers in our offices were coming up and fsck was applied to them, the power went down again and again. Two machines running Linux (on the desktops) unfortunately did not make it (with ext3fs). It resulted in manual check and data loss at those Linux stations. There were no Windows machines in our office, so no comparison, however I believe readers know what happens to Windows in such cases. There was one Apple laptop at that time in our office that survived on battery so again no comparison. Finally, there were two BSD servers and there was absolutely no problem on those BSDs. All services were up and about in just a short time and all files were found complete, unbroken and clean. That is why when I write about file sharing, I write about BSD.

File sharing options.

How do you share files between multiple operating systems and keep your data safe? There are many options and we will go through all of them. Sharing files between Windows, Apple and Linux or BSD machines means sharing using various protocols:

- NFS
- Samba
- FTP
- SSH/SCP
- Apple talk/Apple share

There are many differences between the ways you can share your files using these different protocols. There can be some incompatibility between versions used on different machines, such as NFS3/NFS4 or various versions of Samba. Most problems are solved by setting your firewalls competently and opening all the ports the protocols need to use. A typical example of a problem is the FTP server not only needs ports 22 and 21 open for communication, but also a range of ports between 1024-5000 for data transfer.

NFS – Network File System

We will start with the Unix way first. Sharing your files with NFS is an old and perfectly functional way that preserves all Unix features such as user and group permissions, which are probably very important for every Unix administrator and users as well. NFS will work perfectly with Linux, BSD, Mac OS/X, and any kind of commercial Unix such as AIX or Solaris.

NFS has many very good features which allow not only transfer but real work in the system over the net. Shares are mounted as local disks and depending on the configuration options allow users to feel as if the disk is part of their computer even if the server is thousands miles away. Users who mount the shares can open files directly, read, write and even start programs from the mounted disks. If you read some NFS related blogs or how-tos you can come across a discussion about mounting *soft* or *hard*. Mounting *soft* means data are written to your PC's cache first and, if the server does not respond, your computer caches the data locally and moves it to the server when the connection is regained. So it is comfortable for you, however the data is not safe. If you lose the connection for long periods and switch off your machine, you might assume the data was written even though it was not. The *hard* option is writing the data to the server without delay which helps keep it safe, but in case of slow or low quality connection it can result in a frequent computer freezes.

The machine working as a client – that is your PC – will think the disk is not answering and thus stops working until it gets answer again. Now if you ask me, the best choice is *hard* if you have reliable connection and important data, and *soft* if your connection is unstable and you do not work on a serious stuff.

So, enough theory, let's see how to setup and configure NFS on OpenBSD and NetBSD, and FreeBSD, then explain how to start it and access from various clients.

OpenBSD NFS server

In `/etc/rc.conf` these options have to be set:

```
   NFS_server=YES          #
see sysctl.conf for NFS client
configuration
   nfsd_flags="-tun 4"
   lockd=YES
   portmap=YES
```

in `/etc/rc.conf.local` you have to set :

```
   portmap=YES
```

In `/etc/export` you have to specify what and how you are going to share and who can access that:

```
   /mnt/disk1 -alldirs  -ro
   /home/david -alldirs -
network=192.168.1.0 -
mask=255.255.255.0
   /var -alldirs 10.27.68.83 -
maproot=501:505
```

The above configuration example allows everyone to mount `/mnt/disk1` for reading.

Everyone in the network 192.168.1.0 with net mask 255.255.255.0 is allowed to mount David's home directory, and everyone from the computer with the IP 10.27.68.83 to mount `/var` and map users with UID between 501 and 505 to root. If you are interested in more options, study man exports.

If you make changes to `/etc/exports` you have to do the following to make it work without reboot:

```
#  kill -HUP `cat /var/run/mountd.pid`
```

NetBSD NFS server

In `/etc/rc.conf` you have to set:

```
   rpcbind=yes
   mountd=yes
   NFS_server=yes
   NFS_client=yes
   lockd=yes
   statd=yes
```

In `/etc/export` you have to specify what and how you are going to share and who can access that:

```
   /mnt/disk1 -alldirs  -ro
   /home/david -alldirs -network
192.168.1.0 -mask 255.255.255.0
   /var -alldirs 10.27.68.83 -
maproot=501:505
```

The above configuration example allows the same behavior as demonstrated above with OpenBSD. One can clearly recognize that OpenBSD and NetBSD are based on BSD. The only difference in our example is defining network and net mask with the absence of the `=` in NetBSD's config file.

If you make changes to `/etc/exports` you have to do the following to make it work without reboot:

```
# ``kill -HUP `cat /var/run/
mountd.pid`''
```

Free BSD NFS server

In `/etc/rc.conf` you have to set:

```
   rpcbind_enable="YES"
   NFS_server_enable="YES"
   mountd_flags="-r"
   rpc_lockd_enable="YES"
   rpc_statd_enable="YES"
```

In `/etc/export` you have to specify what and how you are going to share and who can access that:

```
   /mnt/disk1 -alldirs  -ro
   /home/david -alldirs -network
192.168.1.0 -netmask 255.255.255.0
   /var -alldirs 10.27.68.83 -
maproot=501:505
```

So again exactly the same configuration as with NetBSD, but instead of `-mask` use `-netmask`. If you make changes to `/etc/exports` you have to do the following to make it work without reboot:

```
   # /etc/rc.d/mountd onereload
```

Configuring the NFS client and mounting the share

Connecting your station to the server and mounting the share is generally very similar on all NFS-capable systems. You have to install the NFS client, enable portmap and then mount the share. If you issue a command:

```
   # showmount -e 10.27.68.81
```

It will show you which shares are available on the server with IP 10.27.68.81, so you will see something like:

```
   Exports list on 10.27.68.81:
   /home/NFS      10.27.68.80
```

Which tells you there is a share `/home/NFS` which computers in network 10.27.68.80 can connect to. If you are in the 10.27.68.80 network, you can mount it, that can be done like this:

```
   # mount -t NFS 10.27.68.81:/home/
NFS  /mnt/myshares
```

If you want to specify more options you can do so with `-o`:

```
   # mount  -t NFS -o rw,nosuid,nodev
,hard,noexec,soft 10.27.68.81:/home/
NFS  /mnt/myshares
```

Now if you go to `/mnt/myshares` on your computer you will see the shared files. Whether you can access them or not depends on many things such as permissions and the configuration of the server and your client.

On BSD's

use the command line as shown above (with the alternation to use: `mount_NFS` command)

On Linux

While many distros have their own specific tools to mount the shares, e.g. Mandriva and OpenSuSE can do so very easily through their configuration centers, all Linux distros can use the command line `mount` command to mount NFS as well. There is one point worth mentioning here, modern Linux usually uses wsize and rsize 8192 by default and that can cause problems with BSD servers, as many support only wsize and rsize 1024. I suggest you add the option -o wsize=1024,rsize=1024 when you mount the share on your Linux machines.

On Mac

A very comfortable way in graphical interface is using the Mac's finder:

- With the Finder active, choose Connect to Server from the Go menu; the Connect to Server window appears.
- In the Address text box, enter: `NFS://10.27.68.81:/home/NFS`
- Click Connect; the Connect to Server window closes. With Mac OS X 10.1 or later the NFS exports will mount on the desktop (unless you've turned that feature off on the server, then the export will only appear in the Finder window).

On AIX

The mount command for AIX is also quite similar: `mount 10.27.68.81:/home/NFS /mnt/myshares`

The biggest difference is that AIX stores this information in `/etc/filesystems` instead of `/etc/fstab`. Here is an example of the format used, as it differs from Linux and BSD:

```
/mnt/myshares:
    dev      = "/home/NFS"
    vfs       = nfs
    nodename    = 10.27.68.81
    mount       = true
    options       = rw,hard,wsize=10
24,rsize=1024,vers=3,proto=udp
    account    = false
```

On Solaris

```
mount -F nfs  10.27.68.81:/home/NFS
/mnt/myshares
```

On Windows

Technote 324055 describes the process of installing and setting up the NFS client from the Services for Unix media. In short:

- Download Services for Unix from Microsoft at *http://www.microsoft.com/downloads*
- Run the self-installer (I recommend giving it a specific path)
- Navigate to the directory and doubleclick SfuSetup.msi
- Choose the custom installation
- Choose *Entire feature will be installed on hard drive* for both NFS and Authentication Tools for *NFS›User Name Mapping/Server* for NFS Authentication
- De-select the rest
- Accept the defaults the rest of the way.
- Launch All *Programs›Windows Services for Unix›Services* for Unix Administration
- Click on Client for NFS
- Set the Transport, mount type, read and write buffers here under Performance
- Reboot
- Mounting is now possible via the normal *Explorer›Tools›Map Network Drive* as `10.27.68.81:/home/NFS`

If you want to make the mount permanent under Linux, you have to add a line in your `/etc/fstab`. Here you should be careful, as various Unix systems behave differently if the share is set as `auto`, but the server is unreachable. It can slow down the start of your machine or interrupt it completely. Modern user-aimed Linux distros such as Ubuntu will let you continue and skip this, so you do not have to worry too much.

Summary

Using NFS for file sharing allows you to mount the share like a local disk without reservations. User restrictions and group privileges apply here. Hard or soft options have large effect on the behavior. NFS can be used on all systems with an NFS client. Permanent NFS records can affect the start of your client machine. Most systems require root permissions to mount NFS shares.

To be continued

In the next issue of BSD-Mag we will have a look at the SAMBA share and its options.

Special thanks to Mike Bybee for adding the how-to for the NFS on Windows part.

Sources and recommended reading:

- *http://openbsd.org/faq/faq6.html#NFS*
- *http://www.netbsd.org/docs/guide/en/chap-net-services.html#chap-net-services-nfs*
- *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/network-nfs.html*
- *http://onlamp.com/bsd/2000/11/14/OpenBSD.html*
- *http://support.apple.com/kb/TA22243?viewlocale=en_US*
- *http://www.freebsd.org/cgi/man.cgi?query=export&apropos=0&sektion=0&manpath=FreeBSD+8.0-RELEASE&format=html*
- *http://www.openunix.eu/*
- *http://ubuntuforums.org/showthread.php?t=310168*
- *http://support.microsoft.com/kb/324055*
- *http://tldp.org/HOWTO/NFS-HOWTO/interop.html*
- *http://unixtechtips.blogspot.com/2007/01/using-nfs-solaris.html*
- Michael Lucas: FreeBSD
- Jon Lasser: THINK Unix

About the Author

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

WHY REMSYS?

- ▶▶ PROFESIONAL EXPERTISE
- ▶▶ TRUE 24/7 AVAILABILITY
- ▶▶ COST EFFECTIVE SOLUTIONS
- ▶▶ SATISFACTION GUARANTEE

REMSYS
MANAGED TECHNICAL SOLUTIONS

- INFRASTRUCTURE MANAGEMENT
- SERVER MANAGEMENT
- VIRTUALIZATION
- MONITORING
- SECURITY
- MANAGED BACKUPS
- DISASTER RECOVERY
- MANAGED CLUSTERS
- TECHNICAL SUPPORT

US: + 1 2023792323
EUROPE: + 373 22595103
E-MAIL: sales@remsys.com
HTTP: //WWW.REMSYS.COM

IPsec VPNs

An Introduction to IKE and IPsec

Paul McMath

This article concerns itself with IPsec and IKE, the protocols used to build IPsec based VPNs (hereafter referred to simply as a VPN).

Introduced are the basic elements of IPsec and IKE which are useful for understanding how to configure, monitor or trouble-shoot a VPN configuration. An example VPN using OpenBSD is presented and subsequently examined to demonstrate aspects of how IPsec and IKE work. Not discussed are the relative merits of design choices or configuration options. Whenever possible, OpenBSD's default configuration options are used.

This article has four parts. The first is an overview of an IPsec VPN, the ESP protocol and IKE key management. The second discusses configuring VPN gateways and demonstrates some of the tools to establish, manage and debug a VPN on OpenBSD. The third part looks at VPN traffic at the packet level using `tcpdump`, and the final part demonstrates a basic configuration for roaming clients.

IPSEC VPN

IPsec refers to two protocols, *Authentication Header* (AH) and *Encapsulating Security Payload* (ESP), both of which provide different types of protection to IP datagrams. Both protocols have two modes of operation – transport mode and tunnel mode. The kinds of protection they provide are confidentiality (data can only be read by the intended recipient), data integrity (certainty that the data hasn't been modified while in transit), authenticity (certainty about the sender), and replay protection (traffic belonging to one session cannot be used as part of a new session). This kind of security is achieved through the use of encryption algorithms, hash algorithms, and standard authentication mechanisms (public key encryption, shared secret, digital signatures). In this article, *IPsec traffic* refers to network traffic protected by either ESP or AH.

In addition to AH and ESP, there is the *Internet Key Exchange* (IKE) protocol. IKE is a hybrid protocol, meaning it combines functionality defined by other protocols, specifically the ISAKMP protocol, the OAKLEY protocol, and the SKEME protocol. The role of IKE in a VPN is to establish and manage a *Security Association* (SA) between two peers. The SA defines many configurable attributes of IPsec traffic, in particular, how the secure communication between two peers is achieved (e.g., which algorithms are used, the authentication method, key lengths, etc). IKE *negotiates* the SA between peers, whereby both reach an agreement on the set of SA attributes to use for protecting their IPsec traffic. Both peers must have and maintain matching SAs in order to secure traffic between them.

In this article, we will examine an OpenBSD IPsec VPN implementation that will use the ESP protocol in tunnel mode to secure traffic between two private, internal networks. VPN gateways acting as multi-homed hosts will protect the traffic as it traverses a WAN (i.e., the public internet). In our example, both gateways will run OpenBSD's isakmpd, a daemon process which implements the IKE protocol. Later, we will modify the configuration to accommodate roaming users (also running OpenBSD).

Figure 1 is a representation of our demonstration network. The hosts `vpn-gateway1` and `vpn-gateway2` have external interfaces on the WAN. Both gateways are also multi-homed, each being connected to an internal network (`corp-net1` and `corp-net2`) that uses a private address space. Any host on either of the internal networks can securely communicate with any host on the other network. The two gateways will protect their traffic as it traverses the WAN.

Overview of ESP (in tunnel mode)

Figure 2 shows a typical IPv4 datagram before and after the application of *Encapsulating Security Payload* (ESP) in tunnel mode. ESP puts data in front of and behind the original IPv4 datagram. After ESP is applied, all of the original IPv4 datagram is encrypted.

Important values within the ESP header are:

- *Security Parameter Index* (SPI) – a randomly generated value used to determine which SA applies to a ESP packet.
- Sequence number – This value provides anti-replay protection. It is initialized to 0 when the SA is established and incremented by 1 for every packet sent.

The ESP-Trailer contains padding (necessary for boundary alignment) and some other fields not covered in this article.

The Authentication Data field contains the *Integrity Check Value* (ICV). This value (derived cryptographically) is used to ensure the authenticity and integrity of the packet. As the diagram shows, it is calculated over the entire ESP packet, excluding the Authentication Data portion.

Also note that because the original IP header is now encrypted, it can no longer be used to route the packet. Therefore a new IP header is placed at the beginning of the packet. This is what is meant by tunnel mode – the true source and origin of the datagram is now concealed and protected.

Figure 3 illustrates packet flow for an ESP packet in tunnel mode on one side of the VPN. The top diagram shows an inbound IPv4 datagram before and after ESP encapsulation. The diagram below shows traffic flow for this packet. It arrives on the external bge0 interface of `vpn-gateway1` as an ESP packet; its inner contents are still encrypted. The source address in the outer IP header is 2.2.2.2 (`vpn-gateway2`); the destination address is 1.1.1.1 (`vpn-gateway1`).

The presence of the ESP header identifies the packet for IPsec processing on `vpn-gateway1`. After de-capsulation the original IPv4 datagram is no longer encrypted; the outer IP header is now the original header (with source address, 192.168.10.20, a host on the `corp-net2`, and a destination address for a host on the `corp-net1` network). When `vpn-gateway1` forwards the datagram to its final destination the datagram appears as normal IPv4 traffic.

Hosts on the `corp-net1` network have their routing tables configured to use `vpn-gateway1` for all traffic to the `corp-net2` network. The host1 machine sends a normal IPv4 datagram to the gateway for forwarding. The gateway determines that the datagram should be processed by IPsec based upon values in the IPv4 header. When the packet leaves `vpn-gateway1`, the original IPv4 datagram is encrypted and a new IP header identifies the source and destination IP address as those of the two VPN gateways.

IKE Overview

IKE is a hybrid protocol based primarily on ISAKMP with additional functionality derived from other protocols. ISAKMP terminology is therefore often used when discussing IKE.

The purpose of IKE is to establish and manage security associations between two endpoints (peers). Each endpoint of the VPN must have a SA which corresponds to a SA on the other endpoint.

Before a SA comes into being both peers must reach an agreement on the new SA's attributes. These attributes determine characteristics of the SA; examples include the particular IPsec protocol (ESP or AH), encryption algorithms and the authentication method used to verify the identity of the

Listing 1. /etc/ipsec.conf for vpn_gateway1

```
#ipsec.conf vpn_gateway1
#
#corp_net1: 10.0.10.0/24
#corp_net2: 192.168.10.0/24

# Macro for remote vpn peer
vpn_gateway2 = "2.2.2.2"

ike passive esp tunnel from 192.168.10.0/24 to 10.0.10.0/24 \
         peer $vpn_gateway2 \
        main auth hmac-sha1 enc aes group modp1024 \
        quick auth hmac-sha2-256 enc aes
```

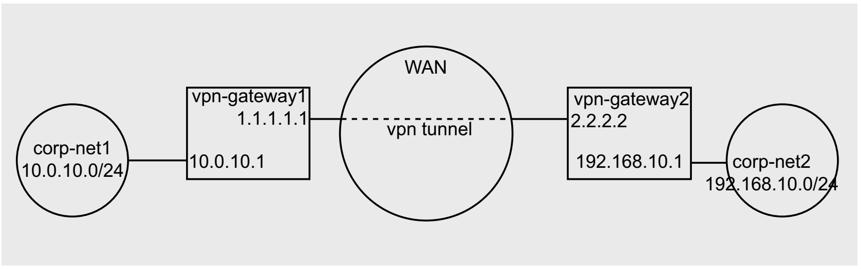


Figure 1. Two private networks connected accross a public WAN through a VPN tunnel

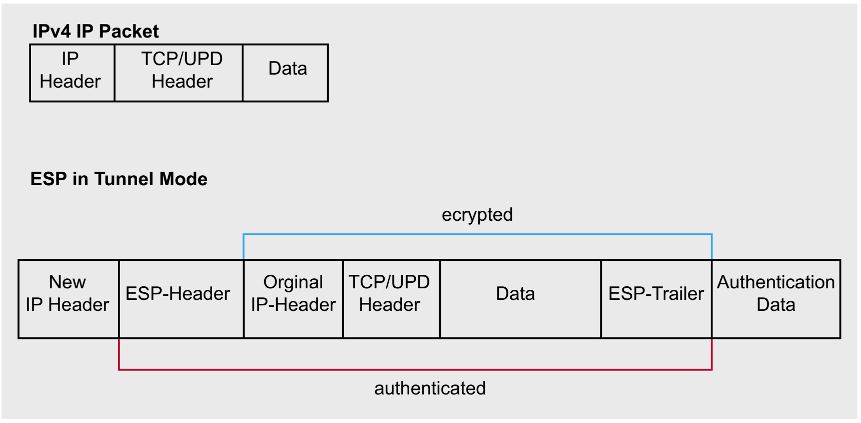


Figure 2. IPv4 datagram before and after ESP encapsulation in tunnel mode

Listing 2. /etc/ipsec.conf for vpn_gateway2

```
#ipsec.conf vpn_gateway2
#
#corp_net1: 10.0.10.0/24
#corp_net2: 192.168.10.0/24

# Macro for remote vpn peer
vpn_gateway1 = "1.1.1.1"

ike active esp tunnel from 10.0.10.0/24 to 192.168.10.0/24 \
        peer $vpn_gateway1 \
       main auth hmac-sha1 enc aes group modp1024 \
       quick auth hmac-sha2-256 enc aes
```

remote endpoint. The process whereby the two peers reach agreement on the SA's attributes is called *negotiation*.

Other steps must also be completed before a new SA is created. Depending on the type of SA, these may include establishing a shared secret (used to generate secret keys), authenticating the identity of the peer or generating keying material.

Another characteristic of SAs is that they have lifetimes, that is, they expire and a new SA must be created. Duration of life is measured in either time or kilobytes of data transferred. These are attributes which must also be negotiated.

Daemons implementing the IKE protocol communicate by exchanging information in *messages* sent in UDP datagrams to port 500 or 4500. Messages are always exchanged within the context of an *exchange type*. An exchange type defines the purpose and the content of a message. Some exchange types are intended to establish new SAs and therefore define the number and content of the messages which must be exchanged to successfully instantiate the SA. Other exchange types are for sharing status information. IKE defines five exchange types: *Identity Protection*, *Aggressive*, *Informational*, *Quick Mode*, and *New Group Mode*. The first three are derived from ISAKMP, the latter two are specific to IKE. A particular message belongs to one (and only one) exchange type. The exchange types *New Group Mode* and *Aggressive* will not be discussed as they are not used in our implementation.

Negotiation itself consists of an exchange of messages containing *proposals*. A proposal contains a proposed set of *attribute/value* pairs for the SA being negotiated. Peers can either accept or reject the proposals sent from the other peer. Usually, the *initiator* of a connection sends a list of one or more proposals, and the remote peer, the *responder*, replies with a proposal chosen from the list. If the initiator doesn't send a proposal that the responder can accept, then the negotiation fails and no SA is created.

(Although the terms *initiator* and *responder* suggest a *client/server* relationship, outside the context of establishing the SAs, both initiator and responder communicate with each other as equal peers.)

IPsec traffic requires establishing two types of SAs – an ISAKMP SA and an IPSEC SA. They are created sequentially, and the first must complete successfully before the second can begin. Establishment of the first is called *Phase 1* and results in an ISAKMP SA. In our example, the IKE messages which establish the ISAKMP SA are of the *Identity Protection* exchange type (phase 1 using *Identity Protection* is also referred

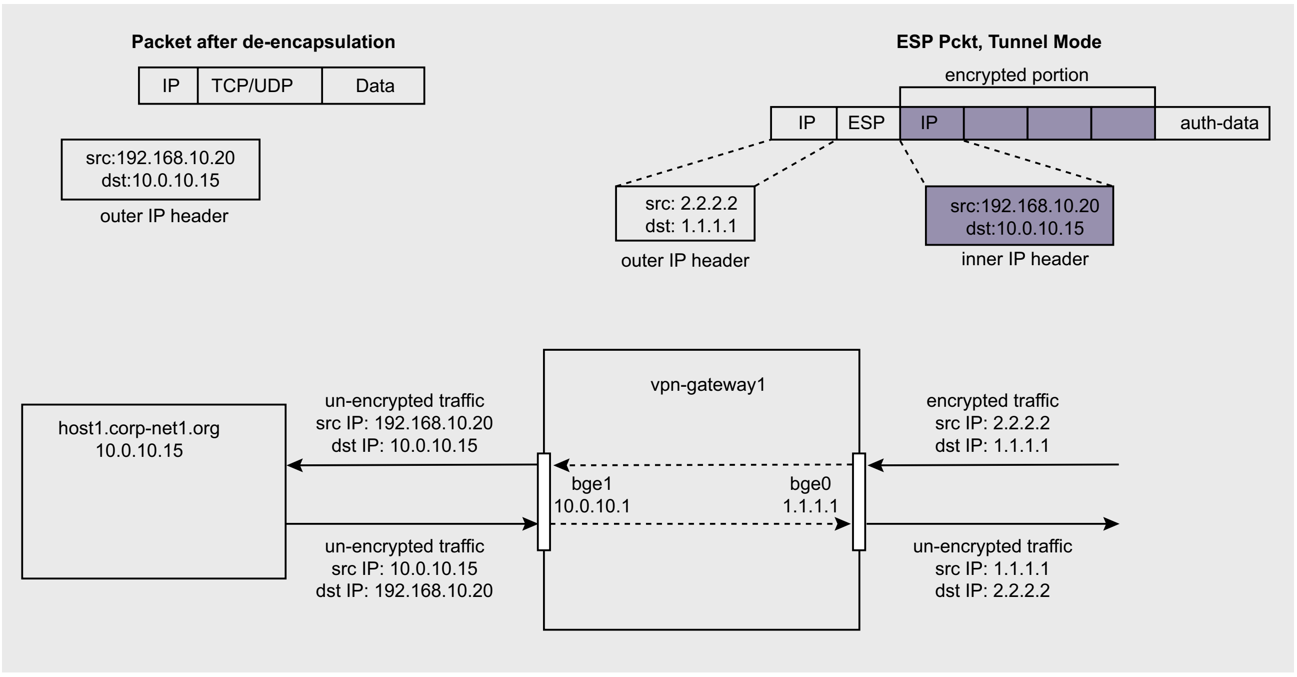


Figure 3. Traffic flow for an ESP packet

to as *main mode*). This exchange type mandates six messages be exchanged, three from each peer. The purpose of these messages is to 1) negotiate the proposed SA's attribute values, 2) establish a shared secret using the Diffie-Hellman protocol, and 3) authenticate the identity of the remote peer using either a pre-shared secret, public key encryption or a digital signature. When completed, the new ISAKMP SA is used to protect messages sent in *Phase 2*, which will establish the IPSEC SA.

Phase 2 messages are of the exchange type *Quick mode* (*quick*, because if *Perfect Forward Secrecy* (PFS) is not required, then an expensive Diffie-Hellman calculation is not necessary). Quick mode defines a three message exchange which will negotiate proposed SA attribute values, ensure against the possibility of replay and generate keying material. If PFS is required, there will still only be three messages exchanged, but keying material for Diffie-Hellman will be included in the messages. When phase 2 completes there will be an IPSEC SA capable of protecting traffic with ESP or AH.

IPSEC SAs apply to traffic in one direction only, therefore they are created in pairs, each peer having two, one for inbound and the other for outbound traffic. When discussing establishment of an IPSEC SA, what is always meant is establishing a pair of IPSEC SAs on each peer for bi-directional traffic.

The IPSEC SA and the ISAKMP SA have distinct functions. The IPSEC SA defines the protection for IPsec traffic (e.g., the encryption and authentication algorithms defined for the IPSEC SA are used when processing an ESP packet). In our example, this is traffic between the two private internal networks. The ISAKMP SA protects the UDP traffic between the two daemons running on the gateways and communicating via the IKE protocol. This *IKE traffic* contains the IKE messages exchanged as part of managing SAs or creating new ones, either at the request of initiators or due to old SAs reaching their maximum life and expiring.

Information about IPSEC SAs (e.g., when and how IPsec protection is applied to IP datagrams) is kept in two databases: the *Security Policy Database*

Listing 3. pf rules for vpn_gateway1

```
# pf rules for vpn_gateway1
#
#corp_net1: 10.0.10.0/24
#corp_net2: 192.168.10.0/24
vpn_gateway1="1.1.1.1"
vpn_gateway2="2.2.2.2"
########
# These rules must be present for any IPsec/ESP traffic
# pass esp traffic
pass in on $ext_if proto esp from $vpn_gateway2 to $vpn_gateway1
pass out on $ext_if proto esp from $vpn_gateway1 to $vpn_gateway2
# pass IKE traffic on ports 500 and port 4500
pass in on $ext_if inet proto udp from $vpn_gateway2 to $vpn_gateway1 \
     port { 500, 4500 }
pass out on $ext_if inet proto udp from $vpn_gateway1 to $vpn_gateway2 \
     port { 500, 4500 }

# IP-in-IP traffic flowing between gateways on the enc0 interface.
pass in on enc0 proto ipencap from $vpn_gateway2 to $vpn_gateway1 \
     keep state (if-bound)

# pass traffic on the enc0 interface
pass in on enc0 from 10.0.10.0/24 to 192.168.10.0/24 \
     keep state (if-bound)
pass out on enc0 from 192.168.10.0/24 to 10.0.10.0/24 \
     keep state (if-bound)
```

Listing 4. pf rules for vpn_gateway2

```
# pf rules for vpn_gateway2
#
#corp_net1: 10.0.10.0/24
#corp_net2: 192.168.10.0/24
vpn_gateway1="1.1.1.1"
vpn_gateway2="2.2.2.2"
########
# These rules must be present for any IPsec/ESP traffic
# pass esp traffic
pass in on $ext_if proto esp from $vpn_gateway1 to $vpn_gateway2
pass out on $ext_if proto esp from $vpn_gateway2 to $vpn_gateway1
# pass IKE traffic on ports 500 and port 4500
pass in on $ext_if inet proto udp from $vpn_gateway1 to $vpn_gateway2 \
     port { 500, 4500 }
pass out on $ext_if inet proto udp from $vpn_gateway2 to $vpn_gateway1 \
     port { 500, 4500 }

# IP-in-IP traffic flowing between gateways on the enc0 interface.
pass in on enc0 proto ipencap from $vpn_gateway1 to $vpn_gateway2 \
     keep state (if-bound)

# pass traffic on the enc0 interface
pass in on enc0 from 192.168.10.0/24 to 10.0.10.0/24 \
     keep state (if-bound)
pass out on enc0 from 10.0.10.0/24 to 192.168.10.0/24 \
     keep state (if-bound)
```

Listing 5. Output of '/sbin/ipsecctl -sa'

```
vpn-gateway1> ipsecctl -sa
FLOWS:
flow esp in from 10.0.10.0/24 to 192.168.10.0/24 peer 2.2.2.2 srcid
1.1.1.1/32 dstid 2.2.2.2/32 type use
flow esp out from 192.168.10.0/24 to 10.0.10.0/24 peer 2.2.2.2 srcid
1.1.1.1/32 dstid 2.2.2.2/32 type require
SAD:
esp tunnel from 2.2.2.2 to 1.1.1.1 spi 0x1795569e auth hmac-sha2-256 enc aes
esp tunnel from 1.1.1.1 to 2.2.2.2 spi 0x6bbaeabd auth hmac-sha2-256 enc aes
```

Listing 6. Output of '/sbin/ipsecctl -vv -s all'

```
vpn-gateway1> ipsecctl -vv -s all
FLOWS:
@0 flow esp in from 10.0.10.0/24 to 192.168.10.0/24 peer 2.2.2.2 srcid
1.1.1.1/32 dstid 2.2.2.2/32 type use
@1 flow esp out from 192.168.10.0/24 to 10.0.10.0/24 peer 2.2.2.2 srcid
1.1.1.1/32 dstid 2.2.2.2/32 type require
SAD:
@0 esp tunnel from 1.1.1.1 to 2.2.2.2 spi 0x6bbaeabd auth hmac-sha2-256
enc aes
        sa: spi 0x6bbaeabd auth hmac-sha2-256 enc aes
                state mature replay 16 flags 4
        lifetime_cur: alloc 0 bytes 0 add 1259977522 first 0
        lifetime_hard: alloc 0 bytes 0 add 1200 first 0
        lifetime_soft: alloc 0 bytes 0 add 1080 first 0
        address_src: 1.1.1.1
        address_dst: 2.2.2.2
        identity_src: type prefix id 0: 1.1.1.1/32
        identity_dst: type prefix id 0: 2.2.2.2/32
        src_mask: 255.255.255.0
        dst_mask: 255.255.255.0
        protocol: proto 0 flags 0s
        flow_type: type use direction out
        src_flow: 192.168.10.0
        dst_flow: 10.0.10.0
        remote_auth: type rsa
@0 esp tunnel from 2.2.2.2 to 1.1.1.1 spi 0x1795569e auth hmac-sha2-256 enc aes
        sa: spi 0x1795569e auth hmac-sha2-256 enc aes
                state mature replay 16 flags 4
        lifetime_cur: alloc 0 bytes 0 add 1259977522 first 0
        lifetime_hard: alloc 0 bytes 0 add 1200 first 0
        lifetime_soft: alloc 0 bytes 0 add 1080 first 0
        address_src: 2.2.2.2
        address_dst: 1.1.1.1
        identity_src: type prefix id 0: 2.2.2.2/32
        identity_dst: type prefix id 0: 1.1.1.1/32
        src_mask: 255.255.255.0
        dst_mask: 255.255.255.0
        protocol: proto 0 flags 0
        flow_type: type use direction in
        src_flow: 10.0.10.0
        dst_flow: 192.168.10.0
        remote_auth: type rsa
```

(SPD) and the *Security Association Database* (SAD). An entry in the SPD is called a *policy*. A policy determines the *disposition* of all IP traffic, i.e., whether or not a particular IP datagram should be given IPsec protection. Within the policy are values called *selectors* which correspond with values found in an IP header and transport layer header. These typically include source/destination IP addresses, and *source/destination* port. An IP datagram will be processed by IPsec if its headers have values that match the selectors for a policy in the SPD. (A policy is sometimes referred to as a *flow* – a uni-directional flow of traffic between two endpoints which will be protected by IPsec. In our example (see Figure 1), there will be two flows, each extending from one internal, private network to the other internal, private network.)

Each policy in the SPD references a SA in the *Security Association Database* (SAD). The SAD holds information about the active SAs. This includes attribute values negotiated when the SA was established as well as values which are stateful (e.g., current lifetime). When a datagram is processed by an IPsec protocol, values in the SAD determine specifically how the datagram should be processed (which *authentication/encryption* algorithms to use, etc).

Because a VPN gateway may simultaneously have multiple phase 2 IPSEC SAs protecting traffic from any number of networks or hosts, and each SA may use different attributes to protect their traffic, there must be a means of determining which SA applies to particular ESP packet. Within the SAD, uniqueness for a SA is determined by a triplet consisting of the IPsec protocol (AH or ESP), destination IP address and an arbitrarily chosen number called *Security Parameter Index* (SPI).

The SPI is included in every ESP header processed by IPsec and used by the remote peer to determine the corresponding SA in its SAD. When the matching SA is found, the attributes of the SA are used to process the packet (e.g., authenticate and de-crypt its contents, etc).

OpenBSD has a tool for examining the contents of the SAD and the SPD. Its usage is discussed later.

Listing 7. tcpdump capture of Phase 1 packet exchange

```
2.2.2.2.500 > 1.1.1.1.500: isakmp v1.0 exchange ID_PROT
        cookie: b3ec3c1bd7d14984->0000000000000000 msgid: 00000000 len: 184
        payload: SA len: 56 DOI: 1(IPSEC) situation: IDENTITY_ONLY
            payload: PROPOSAL len: 44 proposal: 1 proto: ISAKMP spisz: 0 xforms: 1
                payload: TRANSFORM len: 36
                    transform: 0 ID: ISAKMP
                        attribute ENCRYPTION_ALGORITHM = AES_CBC
                        attribute HASH_ALGORITHM = SHA
                        attribute AUTHENTICATION_METHOD = RSA_SIG
                        attribute GROUP_DESCRIPTION = MODP_1024
                        attribute LIFE_TYPE = SECONDS
                        attribute LIFE_DURATION = 3600
                        attribute KEY_LENGTH = 128
        payload: VENDOR len: 20 (supports OpenBSD-4.0)
        payload: VENDOR len: 20 (supports v2 NAT-T, draft-ietf-ipsec-nat-t-ike-02)
        payload: VENDOR len: 20 (supports v3 NAT-T, draft-ietf-ipsec-nat-t-ike-03)
        payload: VENDOR len: 20 (supports NAT-T, RFC 3947)
        payload: VENDOR len: 20 (supports DPD v1.0)

 1.1.1.1.500 > 2.2.2.2.500: isakmp v1.0 exchange ID_PROT
        cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: 00000000 len: 184
        payload: SA len: 56 DOI: 1(IPSEC) situation: IDENTITY_ONLY
            payload: PROPOSAL len: 44 proposal: 1 proto: ISAKMP spisz: 0 xforms: 1
                payload: TRANSFORM len: 36
                    transform: 0 ID: ISAKMP
                        attribute ENCRYPTION_ALGORITHM = AES_CBC
                        attribute HASH_ALGORITHM = SHA
                        attribute AUTHENTICATION_METHOD = RSA_SIG
                        attribute GROUP_DESCRIPTION = MODP_1024
                        attribute LIFE_TYPE = SECONDS
                        attribute LIFE_DURATION = 3600
                        attribute KEY_LENGTH = 128
        payload: VENDOR len: 20 (supports OpenBSD-4.0)
        payload: VENDOR len: 20 (supports v2 NAT-T, draft-ietf-ipsec-nat-t-ike-02)
        payload: VENDOR len: 20 (supports v3 NAT-T, draft-ietf-ipsec-nat-t-ike-03)
        payload: VENDOR len: 20 (supports NAT-T, RFC 3947)
        payload: VENDOR len: 20 (supports DPD v1.0)

 2.2.2.2.500 > 1.1.1.1.500: isakmp v1.0 exchange ID_PROT
        cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: 00000000 len: 228
        payload: KEY_EXCH len: 132
        payload: NONCE len: 20
        payload: NAT-D len: 24
        payload: NAT-D len: 24

 1.1.1.1.500 > 2.2.2.2.500: isakmp v1.0 exchange ID_PROT
        cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: 00000000 len: 228
        payload: KEY_EXCH len: 132
        payload: NONCE len: 20
        payload: NAT-D len: 24
        payload: NAT-D len: 24

 2.2.2.2.500 > 1.1.1.1.500: [udp sum ok] isakmp v1.0 exchange ID_PROT
        cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: 00000000 len: 332
        payload: ID len: 12 type: IPV4_ADDR = 2.2.2.2
```

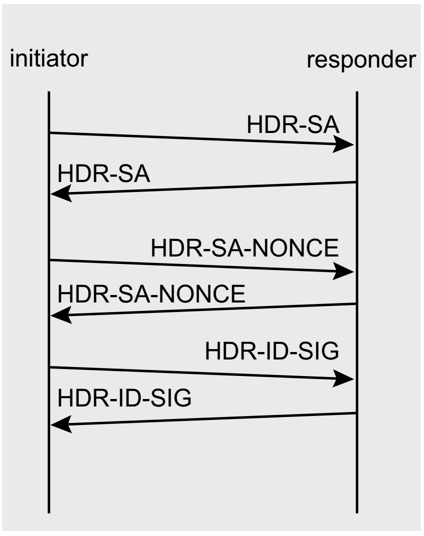


Figure 4. Sequence diagram for Phase 1 ISAKMP Security Assocation authenticating with digital signatures

OpenBSD Configuration

The daemon which implements the IKE functionality in OpenBSD is `/sbin/isakmpd`. Each gateway will run an instance of isakmpd; they will communicate via UDP over port 500.

isakmpd requires a configuration which specifies the details of the SAs it will negotiate. By default, isakmpd reads configuration files in `/etc/isakmpd/`. The syntax for these files can be quite complex however, and while one may use them, a newer tool called `/sbin/ipsecctl` exists which makes configuration of isakmpd much simpler and more intuitive.

Additionally, OpenBSD implements a pseudo-device called `enc(4)`. This is a virtual interface specifically for IPsec traffic, and there is only one supported (`enc0`). Traffic captured on this interface (with `tcpdump`) is seen prior to encapsulation and after de-capsulation (on other interfaces, the encapsulated packets are encrypted and their contents not visible). One purpose of the `enc0` interface is to allow packet filtering of IPsec traffic with pf, OpenBSD's packet filter. Note that if you're running pf on a vpn gateway, then rules must be added to explicitly allow IPsec traffic to pass on the `enc0`.

Rules for configuring isakmpd with ipsecctl are typically kept in a file called `/etc/ipsec.conf`. Each peer must have a SA which corresponds with a SA on the remote peer. Therefore, a rule in ipsec.conf on one gateway must have a corresponding rule in the ipsec.conf of the other gateway.

The `ipsec.conf` for `vpn-gateway1` is Listing 1. The keyword *passive* tells isakmpd not to initiate a connection, but rather wait for incoming connections. We specify *esp* in *tunnel* mode (both are defaults). The *from* and *to* keywords identify the source and destination addresses for traffic that will be protected by ESP. The *peer* keyword is the remote peer running an implementation of IKE (`vpn-gateway2`).

The keyword *main* specifies attributes used for establishing the Phase 1 SA, the ISAKMP SA. (*main* refers to *main mode* – the Identity Protection exchange type). These are also default values. The keywords are:

- `auth` – algorithm used to authenticate IKE messages
- `enc` – algorithm used to encrypt all IKE messages once secret key is established
- `group` – defines values required for the Diffie-Hellman exchange

The fourth line, starting with the keyword *quick* (referring to the Quick Mode exchange type) contains attributes for the phase 2 IPSEC SA. Keywords *auth* and *enc* in quick mode specify the algorithms used for authentication and encryption of the ESP packets. These values are also the defaults.

Listing 2 is the ipsec.conf file `vpn-gateway2`. It is almost identical to the one on `vpn-gateway1`. Differences are:

- Instead of *passive*, we use *active* (the default). This causes isakmpd on `vpn-gateway2` to initiate a connection.
- The *from* and *to* networks are reversed
- the *peer* references `vpn-gateway1`.

Now the pf rules which must be included on `vpn-gateway1` (Listing 3):

The first two rules allow ESP traffic between the two gateways on the external interface. The second two allow IKE UDP traffic on ports 500 and 4500 (use of port 4500 is discussed later). The next rule is specifically for *ipencap* – the protocol for tunneling IP within IP datagrams. The final two rules filter traffic on `enc0` based upon source/destination IP addresses found within the unencrypted IPv4 datagram. Note that the final three rules explicitly bind the state to the enc0 interface. The reason for this is explained in the `enc(4)` manpage. Rules for `vpn_gateway2` (Listing 4) are analogous.

Before we can start the daemons, it will be necessary to exchange public keys of the hosts `vpn-gateway1` and `vpn-gateway2`. The isakmpd daemon uses a public/private key pair found in `/etc/isakmpd/local.pub` and `/etc/`

Listing 7. tcpdump capture of Phase 1 packet exchange

```
52          payload: SIG len: 260
53          payload: NOTIFICATION len: 28
54              notification: INITIAL CONTACT (b3ec3c1bd7d14984->f3dc38d41c937286)


55      1.1.1.1.500 > 2.2.2.2.500: isakmp v1.0 exchange ID_PROT
56          cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: 00000000 len: 328
57          payload: ID len: 12 type: IPV4_ADDR = 1.1.1.1
58          payload: SIG len: 260
59          payload: NOTIFICATION len: 28
60              notification: INITIAL CONTACT (b3ec3c1bd7d14984->f3dc38d41c937286)
```

`isakmpd/private/local.key`. This is a 2048 bit RSA key pair that is generated automatically on first boot.

Part of establishing a phase 1 SA is authenticating the identity of the peer. The isakmpd daemon defines four ways a host can specify its identity: as an IPv4 or IPv6 address, a *Fully Qualified Domain Name* (FQDN) or a *User Fully Qualified Domain Name* (UFQDN). By default, isakmpd sends its IPv4 address as the identity. An identity of a different type would have to be specified in *ipsec.conf*.

The identity type also determines the file system directory where the public key of the remote peer is kept. Since we're going with isakmpd's default choice of an IPv4 identity, the remote peer's public key should go in `/etc/isakmpd/pubkeys/ipv4/` and should be renamed to match the IP address the remote peer will use (e.g., the public key of `vpn-gateway1` is in `/etc/isakmpd/pubkeys/ipv4/1.1.1.1` on `vpn-gateway2`).

It is also necessary that hosts on the internal networks route traffic intended for the remote internal network to the VPN gateway (a host on `corp-net1` will send packets destined for `corp-net2` to `vpn-gateway1` for forwarding). The gateways could also be default routes for all traffic leaving the internal networks. In this case, only traffic intended for the remote corporate network is sent through the VPN tunnel.

After loading our new pf rules (pfctl `-f /etc/pf.conf`), we can start the isakmpd daemon and load our configuration from ipsec.conf. Starting with `vpn-gateway1` we execute:

```
/sbin/isakmpd -vKL
```

The `-v` flag enables verbose logging; the `-K` flag tells isakmpd not look for any configuration files (we will use ipsecctl to configure isakmpd). The `-L` flag tells isakmpd to capture an unencrypted copy of the IKE negotiations to a file which can be later read by `tcpdump`. The default location of this file is `/var/run/isakmpd.pcap`. We will examine this file when looking at how isakmpd negotiates and maintains SAs.

The command to load our configuration from `ipsec.conf` is:

```
ipsecctl -v -f /etc/ipsec.conf
```

The `-v` flag is for verbosity, and the `-f` flag passes the file name containing our `ipsec.conf` file.

The above two commands are then executed on `vpn-gateway2`.

With these options, isakmpd will write information to `/var/log/daemon` and `/var/log/messages`. This can provide important information about the negotiation. ipsecctl can show information about the SAs established.

Listing 5 is the output from *ipsecctl -sa*. This confirms the existence of our SAs. The first two lines identify flows in the SPD. The values *in* and *out* refer to the particular direction of traffic to which this flow applies. The output for the second flow basically says that an out-bound IPv4 datagram with a source address on the 192.168.10.0/24 network and a destination address on the 10.0.10.0/24 network will be processed by ESP and sent to the remote peer at 2.2.2.2. Also shown are the source and destination identities of the VPN gateways which are authenticated during phase 1.

Listing 8. tcpdump capture of Phase 2 packet exchange

```
 2.2.2.2.500 > 1.1.1.1.500: isakmp v1.0 exchange QUICK_MODE
  cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: b12eaece len: 428
  payload: HASH len: 24
  payload: SA len: 52 DOI: 1(IPSEC) situation: IDENTITY_ONLY
    payload: PROPOSAL len: 40 proposal: 1 proto: IPSEC_ESP spisz: 4
xforms: 1 SPI: 0x1795569e
      payload: TRANSFORM len: 28
        transform: 1 ID: 3DES
           attribute LIFE_TYPE = SECONDS
           attribute LIFE_DURATION = 1200
           attribute ENCAPSULATION_MODE = TUNNEL
           attribute AUTHENTICATION_ALGORITHM = HMAC_MD5
           attribute GROUP_DESCRIPTION = 14
  payload: NONCE len: 20
  payload: KEY_EXCH len: 260
  payload: ID len: 16 type: IPV4_ADDR_SUBNET = 10.0.10.0/255.255.0
  payload: ID len: 16 type: IPV4_ADDR_SUBNET = 192.168.0.0/255.255.0

   1.1.1.1.500 > 2.2.2.2.500: isakmp v1.0 exchange QUICK_MODE
    cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: b12eaece len: 416
    payload: HASH len: 24
    payload: SA len: 52 DOI: 1(IPSEC) situation: IDENTITY_ONLY 2
      payload: PROPOSAL len: 40 proposal: 1 proto: IPSEC_ESP spisz: 4
xforms: 1 SPI: 0x6bbaeabd

             payload: TRANSFORM len: 28
               transform: 1 ID: 3DES
                attribute LIFE_TYPE = SECONDS
                attribute LIFE_DURATION = 1200
                attribute ENCAPSULATION_MODE = TUNNEL
                attribute AUTHENTICATION_ALGORITHM = HMAC_MD5
                attribute GROUP_DESCRIPTION = 14
    payload: NONCE len: 20
    payload: KEY_EXCH len: 260
    payload: ID len: 16 type: IPV4_ADDR_SUBNET = 10.0.10.0/255.255.255.0
    payload: ID len: 16 type: IPV4_ADDR_SUBNET = 192.168.10.0/
255.255.255.0

  2.2.2.2.500 > 1.1.1.1.500: isakmp v1.0 exchange QUICK_MODE
   cookie: b3ec3c1bd7d14984->f3dc38d41c937286 msgid: b12eaece len: 60
   payload: HASH len: 24
```

The output for the SAD shows the peers, the SPI value and the authentication and encryption algorithms negotiated in phase 2.

Listing 6 executes ipsecctl with maximum verbosity. Here we see many values for the entries in the SAD. The `identity_src` and `identity_dst` specify the peers (the VPN gateways). The `src_flow` and `dst_flow` specify the `src/destination` of traffic that will get ESP protection. Also shown is information about the lifetime's hard and soft limits and the current values.

The `tcpdump` utility is very useful for debugging if there are problems. If pf is configured to log blocked packets, then `tcpdump` can capture them on the `pflog0` interface:

```
tcpdump -e -i pflog0
```

The `-e` flag will print information about the rule that caught the packet. This includes the interface where the packet was seen. If a packet on the `enc0` interface shows up in the output, then the pf rules are not correct. `tcpdump` on the `enc0` interface is also useful:

```
tcpdump -vn -i enc0
```

The `-v` (verbose) flag will show both inner and outer IP headers. (Note: to use expressions to filter traffic on `enc0`, it is necessary to pass the `-l` option to `tcpdump` and grep for the desired output: `tcpdump -vnl -i enc0 | grep 192.168.10.20`) netstat m' and the *route -n* will show will show entries for the ESP *encap family*.

Read the man pages for `enc(4)` and `ipsecctl(8)` and `isakmpd(8)` for more ways to examine the status of the VPN.

Packet Analysis for SA establishment and ESP traffic

We can now look at the file containing the unencrypted copies of the packets exchanged during negotiation of the SAs. The file `/var/run/isakmpd.pcap` gets created when the `-L` flag is passed to isakmpd on start up.

IKE communicates with its peer using *messages* sent in UDP packets to port 500. Every IKE message begins with an ISAKMP header. Important fields in this header include the exchange type (e.g. *Quick Mode*), and cookie. The purpose of the cookie is to prevent certain kinds of DOS attacks where an attacker floods a host with IKE messages containing random forged source IP addresses. In such a scenario, the recipient host would make repeated expensive (and useless) Diffie-Hellman

Listing 9. ESP traffic between two gateways

```
22:58:14.994055 esp 2.2.2.2 > 1.1.1.1 spi 0x1795569e seq 1 len 116 (DF)
22:58:14.996830 esp 1.1.1.1 > 2.2.2.2 spi 0x6bbaeabd seq 1 len 116 (DF)
22:58:14.997749 esp 2.2.2.2 > 1.1.1.1 spi 0x1795569e seq 2 len 100 (DF)
22:58:15.015284 esp 1.1.1.1 > 2.2.2.2 spi 0x6bbaeabd seq 2 len 116 (DF)
22:58:15.016143 esp 2.2.2.2 > 1.1.1.1 spi 0x1795569e seq 3 len 116 (DF)
22:58:15.020448 esp 1.1.1.1 > 2.2.2.2 spi 0x6bbaeabd seq 3 len 884 (DF)
22:58:15.022903 esp 2.2.2.2 > 1.1.1.1 spi 0x1795569e seq 4 len 884 (DF)
22:58:15.216085 esp 1.1.1.1 > 2.2.2.2 spi 0x6bbaeabd seq 4 len 100 (DF)
22:58:15.216907 esp 2.2.2.2 > 1.1.1.1 spi 0x1795569e seq 5 len 116 (DF)
22:58:15.224242 esp 1.1.1.1 > 2.2.2.2 spi 0x6bbaeabd seq 5 len 244 (DF)
```

Listing 10. tunneled IP datagrams captured on enc0

```
23:23:41.668665 (authentic,confidential): SPI 0x1795569e: 2.2.2.2 > 1.1.1.1: 192.168.10.20.4754 > 10.0.10.15.22: .
ack 1 win 16384
23:23:41.684920 (authentic,confidential): SPI 0x6bbaeabd: 1.1.1.1 > 2.2.2.2: 10.0.10.15.22 > 192.168.10.20.4754: P
1:22(21) ack 1 win 17376
23:23:41.687152 (authentic,confidential): SPI 0x1795569e: 2.2.2.2 > 1.1.1.1: 192.168.10.20.4754 > 10.0.10.15.22: P
1:22(21) ack 22 win 16384
23:23:41.690609 (authentic,confidential): SPI 0x6bbaeabd: 1.1.1.1 > 2.2.2.2: 10.0.10.15.22 > 192.168.10.20.4754: P
22:806(784) ack 22 win 17376
23:23:41.695056 (authentic,confidential): SPI 0x1795569e: 2.2.2.2 > 1.1.1.1: 192.168.10.20.4754 > 10.0.10.15.22: P
22:814(792) ack 806 win 15600
23:23:41.890620 (authentic,confidential): SPI 0x6bbaeabd: 1.1.1.1 > 2.2.2.2: 10.0.10.15.22 > 192.168.10.20.4754: .
ack 814 win 17376
23:23:41.892862 (authentic,confidential): SPI 0x1795569e: 2.2.2.2 > 1.1.1.1: 192.168.10.20.4754 > 10.0.10.15.22: P
814:838(24) ack 806 win 16384
23:23:41.899418 (authentic,confidential): SPI 0x6bbaeabd: 1.1.1.1 > 2.2.2.2: 10.0.10.15.22 > 192.168.10.20.4754: P
806:958(152) ack 838 win 17376
23:23:41.904189 (authentic,confidential): SPI 0x1795569e: 2.2.2.2 > 1.1.1.1: 192.168.10.20.4754 > 10.0.10.15.22: P
838:982(144) ack 958 win 16384
23:23:41.944319 (authentic,confidential): SPI 0x6bbaeabd: 1.1.1.1 > 2.2.2.2: 10.0.10.15.22 > 192.168.10.20.4754: P
958:1678(720) ack 982 win 17376
```

calculations as part of negotiating a phase 1 SA. To prevent this, the initiator and the responder each include a cookie in the headers of the first two messages they exchange. When a peer receives a message with the cookie it sent, it knows the source address of the sender is valid.

An individual IKE message is made up of one or more *payloads*. Payloads carry relevant information about either the existing or proposed SA. A payload may itself contain another payload. These are sometimes called *nested* or *hierarchical* payloads. There are different, pre-defined types of payloads, and their type determines their contents.

Three types which can be understood together are the *SA* payload, the *Proposal* payload, and the *Transform* payload. A SA payload is an example of a nested or hierarchical payload. The SA payload contains at least one Proposal payload, which contains at least one Transform payload. The Transform payload is a list of proposed values for the SA's attributes. These include values set in `/etc/ipsec.conf`. The attributes listed in the Transform payload are what must be agreed upon by both peers.

The *Key Exchange* payload and the *Nonce* payload contain keying material for session management, including Diffie-Hellman calculations. The *Identification* payload contains the identity of the peer (e.g., a IPv4 address, FQDN or UserFQDN). This is the identity which will be verified through authentication. Our implementation authenticates with digital signatures sent in the *Signature* payload.

The *Vendor* payload contains vendor specific information and allows peers to determine which IKE implementation the remote side is running. The *Notification* payload is used to exchange various error or status information. A *Delete* payload informs the receiver that the sender is removing a SA from it SAD and IPsec traffic for that SA will no longer be valid.

Figure 4 shows the six packet exchange that creates a Phase 1 ISAKMP SA. The first two packets contain the ISAKMP header (HDR) and the SA payloads. Packets 3 and 4 contain the header, the *Key exchange payloads* (KE) and *Nonce payloads* (NO). Packets 5 and 6 have Identification payloads and Signature payloads. These last two are encrypted using the information exchanged in packets 1 through 4. This exchange is specific to a Phase 1 in main mode using authentication with digital signatures. Were a different authentication mechanism used, the exchange would still require six packets, but packets 3 through 6 would have different payloads.

Listing 7 is the output from `tcpdump -nv -r /var/run/isakmpd.pcap`. This is the unencrypted version of the exchange initiated earlier (the lines are numbered for easier reference – blank lines separate individual packets). The ISAKMP header will be in the first two lines of each packet. The last field on line 1 has the exchange type of this packet, `ID_PROT` (*Identity Protection*). Since this is the initiator's first packet, the responder cookie (on line 2) is set to all zeros. Payloads start on line 3. This packet contains the SA, Proposal and Transform payloads; the indentation by `tcpdump` reflects the nesting. The Transform payload contains the attributes which are being negotiated and the proposed values. Lines 14-18 contain Vendor payloads identifying the particular IKE implementation.

The second packet (lines 19-36) contains the proposal chosen by the responder. This is identical to the proposal sent by the initiator and sending this packet constitutes agreement on the values for the SA's attributes. This packet also contains the responder's cookie. The values for the cookies remain constant for the duration of the SA being established.

The next two packets (lines 37-42 and 43-48) are an exchange of Key Exchange and Nonce payloads. These contain the necessary information to generate a Diffie-Hellman shared secret. The shared secret is then used to generate

Listing 11. Roaming configuration in ipsec.conf on vpn_gateway1

```
# ipsec.conf vpn_gateway1
#
# roaming connections
ike passive esp tunnel from 192.168.10.0/24 to any
```

Listing 12. Roaming client's ipsec.conf

```
# ipsec.conf for roaming client
#

# rule for paul@r500.com
#
vpn_gateway1="1.1.1.1"

ike dynamic esp tunnel from egress to 192.168.10.0/24 \
    peer $vpn_gateway1 \
    srcid paul@r500.com
```

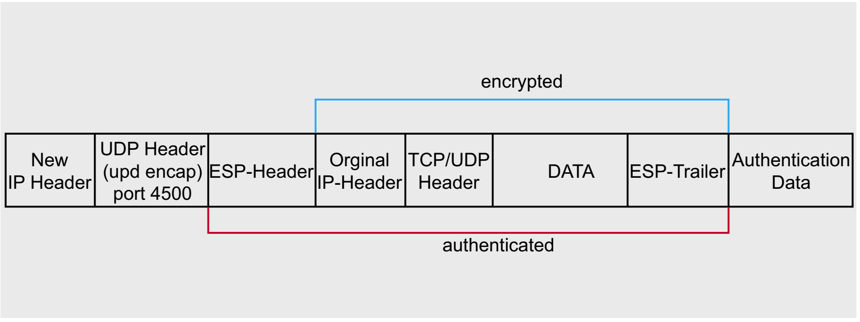


Figure 5. ESP encapsulation in tunnel mode with UDP encapsulation for NAT-T

keying material. After exchanging these two packets the peers are able to encrypt subsequent IKE traffic.

In packets 5 and 6, data after the ISAKMP header is encrypted. The Signature (lines 52 and 58) payloads, along with data derived from the previous messages, are used to authenticate the identities specified in the ID (Identification) payloads (lines 51 and 57).

Listing 8 shows the 3 packets that are the Phase 2, *Quick mode* negotiation which creates an IPSEC SA. These packets are cryptographically protected by the Phase 1 SA. Note that all three packets now have the exchange type `QUICK_MODE` in their header.

The first two packets have the proposals just as in phase 1, but attributes in the Transform payloads differ (the particular attributes of a Transform payload are determined by the exchange type). The proposals also contain the sender's SPI for this SA (last values in lines 5 and 21), and the Identity payloads contain `IPV4_ADDR_SUBNET` identities (lines 15,16 and 31,32) which correspond to the source and destination addresses in the ipsec.conf files.

The final packet (lines 33-35) completes the exchange.

Listing 9 contains ESP traffic captured on the external interface of `vpn_gateway1` with `tcpdump`. Here we see contents of the ESP header – the *Security Parameter Index* (SPI), and the sequence number (for anti-replay protection), being incremented for each packet a peer sends.

Listing 10 contains traffic captured on the `enc0` interface on `vpn-gateway1`. The `-v` flag was used (some of the output has been trimmed) to show contents of the ESP header in addition to the tunneled IP datagram.

Listing 13. vpn_gateway1 pf rules for roaming client

```
# vpn-gateway1 pf.conf rules for roaming clients

vpn_gateway1 = "1.1.1.1"

# pass IKE traffic on ports 500 and port 4500
pass in on $ext_if inet proto udp from any to $ext_if \
     port { 500, 4500 }

pass out on $ext_if inet proto udp from $ext_if to any \
     port { 500, 4500 }

# IP-in-IP traffic flowing between gateways on the enc0 interface.
pass in on enc0 proto ipencap from any to $vpn_gateway1 \
     keep state (if-bound)

# pass traffic on the enc0 interface
pass in on enc0 from any to 192.168.10.0/24 keep state (if-bound)
pass out on enc0 from 192.168.10.0/24 to any keep state (if-bound)
```

Listing 14. pf rules for roaming client

```
# Roaming client's pf.conf rules

wifi_if="iwn0"
vpn_gateway1="1.1.1.1"

# pass IKE traffic on ports 500 and port 4500
pass in on $wifi_if inet proto udp from $vpn_gateway1 to $wifi_if \
     port { 500, 4500 }
pass out on $wifi_if inet proto udp from $wifi_if to $vpn_gateway1 \
     port { 500, 4500 }

# IP-in-IP traffic flowing between gateways on the enc0 interface.
pass in on enc0 proto ipencap from $vpn_gateway1 to $wifi_if \
     keep state (if-bound)

# Rules for enc0
pass in on enc0 from 192.168.10.0/24 to $wifi_if keep state (if-bound)
pass out on enc0 from $wifi_if to 192.168.10.0/24 keep state (if-bound)
```

Roaming clients (road warriors).

We now present a configuration for roaming clients. In this scenario, the same host is both the VPN gateway (i.e., a peer running an implementation of IKE) and the source and destination of the ESP traffic. (In the previous example, the source/destination of ESP traffic was the private internal networks). Physically, these clients may be connecting from an airport, hotel, cafe, etc.

For this example, it is assumed that the roaming clients are connecting from behind a *'NAT'ed* gateway and have a dynamically assigned IP address on an internal network. They will run isakmpd locally and have access to hosts on `corp-net1`. The VPN tunnel will extend from `vpn-gateway1` all the way to the client machine (i.e., through whatever firewall is in between). Traffic originating from the client with a destination on the `corp-net1` network will be sent through the tunnel. Traffic destined for other networks will leave the client normally. This is referred to as a *split horizon* configuration.

Before showing the actual configuration, it will be necessary to briefly discuss issues regarding ESP traffic as it passes through gateways that perform NAT and PAT. Performing NAT/PAT requires the ability to modify values in the IP and transport layer headers of a packet (e.g., src/dest IP addresses, `src/dest` port numbers, checksums). Because ESP encapsulates and encrypts data in the original IP datagram, the contents of the original IP header and TCP/UDP header are unreadable. Gateways that try to apply NAT or PAT to an ESP packet usually corrupt the packet and break IPsec traffic.

The solution is called *NAT-Traversal* (NAT-T). While it doesn't solve all problems related to NAT/PAT and IPsec interoperability, it does cover some common cases. NAT-T is applied by encapsulating all traffic (IKE traffic and ESP packets) within UDP headers. This is referred to as *UDP encapsulation* and is illustrated in Figure 5. The UDP header provides PAT with a source port which it can remap. When NAT-T is implemented, all traffic protected by the IPSEC SA (i.e., the ESP traffic) and all traffic protected by the ISAKMPD SA (IKE traffic) use port 4500 (recall that IKE normally uses port 500). We can expect the gateway that performs PAT to remap port 4500 to another number in the *ephemeral range*.

Because NAT/PAT translation mappings on the gateway have a timeout based upon a period of inactivity, there is the possibility of a gateway dropping the map entry for IKE and IPsec traffic. This would break the SA because source and destination port numbers are used to identify which datagrams should be processed by IPsec. To solve this, peers send each other UDP packets at intervals shorter than the NAT/PAT timeout thresholds. These are called *keep-alive* packets, and their sole purpose is to keep the translation mappings of the senders NAT/PAT from expiring. These keep-alive packets serve no purpose for the recipient and are therefore ignored when they arrive.

Whether or not to use NAT-T is determined by IKE during the phase 1 and phase 2. The first step is for both peers to advertise that they support NAT-T. Then they perform *NAT Discovery* to test for the presence of any NAT occurring between them. If NAT-T is deemed necessary, then the peers will begin UDP encapsulation and switch to port 4500 for all IKE and ESP traffic. Although NAT-T was not required in the

Listing 15. Output of '/usr/sbin/systat rules'

```
1 users    Load 0.11 0.11 0.08                      Wed Dec 30 22:21:52 2009

RULE  ACTION   DIR       IF          PR    PKTS       BYTES  STATES  INFO

 11      Pass   In     bge0         tcp   333         51006    1      inet from any to 1.1.1.1/32 port = ssh
 12      Pass   In     bge0         udp    4           936     1      inet from any to 1.1.1.1/32 port = isakmp
 13      Pass   In     bge0         udp   424         59809    1      inet from any to 1.1.1.1/32 port = ipsec-nat-t
 14      Pass   Out    bge0         udp    0              0    0      inet from 1.1.1.1/32 to any port = isakmp
 15      Pass   Out    bge0         udp    0              0    0      inet from 1.1.1.1/32 to any port = ipsec-nat-t
 16      Pass   In     enc0      ipencap  61           8521    2      inet from any to 1.1.1.1/32
 17      Pass   In     enc0               124         15466    1      inet from any to 192.168.10.0/24
 18      Pass   Out    enc0                0            0      0      inet from 192.168.10.0/24 to any
```

Listing 16. udp encapsulation of isakmpd and esp traffic on port 4500

```
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [no cksum] udpencap: esp abc-internet-cafe.com > vpn-
gateway1.org spi 0x19b15c99 seq 27 len 196
vpn-gateway1.org.4500 > abc-internet-cafe.com.54721: [no cksum] udpencap: esp vpn-gateway1.org > abc-internet-
cafe.com spi 0xb38daea1 seq 26 len 180 (DF)
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [no cksum] udpencap: esp abc-internet-cafe.com > vpn-
gateway1.org spi 0x19b15c99 seq 28 len 100
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [udp sum ok] udpencap: isakmp v1.0 exchange INFO encrypted
cookie: 1539d90ef4af6130->5ae83f08105b67c2 msgid: cb5aacc9 len: 92
vpn-gateway1.org.4500 > abc-internet-cafe.com.54721: [udp sum ok] udpencap: isakmp v1.0 exchange INFO encrypted
cookie: 1539d90ef4af6130->5ae83f08105b67c2 msgid: 606a9a2b len: 92
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [no cksum] udpencap: esp abc-internet-cafe.com > vpn-
gateway1.org spi 0x19b15c99 seq 29 len 244
vpn-gateway1.org.4500 > abc-internet-cafe.com.54721: [no cksum] udpencap: esp vpn-gateway1.org > abc-internet-
cafe.com spi 0xb38daea1 seq 27 len 132 (DF)
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [no cksum] udpencap: esp abc-internet-cafe.com > vpn-
gateway1.org spi 0x19b15c99 seq 30 len 228
190: vpn-gateway1.org.4500 > abc-internet-cafe.com.54721: [no cksum] udpencap: esp vpn-gateway1.org > abc-
internet-cafe.com spi 0xb38daea1 seq 28 len 148 (DF)
abc-internet-cafe.com.54721 > vpn-gateway1.org.4500: [udp sum ok] NAT-T Keepalive
```

previous example, the daemons did perform negotiation and NAT Discovery and subsequently determined that NAT-T wasn't necessary. In Listing 7, lines 14-17 (and 32-35) show each peer sending Vendor payloads identifying themselves and their support for a NAT-T implementation. In the third and fourth packets, payloads of the NAT-D type contain data used for NAT Discovery (lines 41-42 and 47-48, respectively). Were the peers to detect the presence of NAT, then the last two packets of the phase 1 exchange would have occurred over port 4500 and all subsequent IKE traffic would use this port number.

The need for NAT-T with ESP traffic is also negotiated in phase 2. Typically, if NAT is detected in phase 1, the initiator sets the value for the *Encapsulation Mode* attribute to *UDP-Encapsulated-Tunnel* in the phase 2 proposal.

Another (configuration) issue involves the identity of the client. As in the previous example, both the client and the gateway have exchanged public keys. The `isakmpd` daemon maps an identity type (IPv4, IPv6, FQDN, UFQDN) to a particular directory in the filesystem where it expects to find a public key for the remote identity (e.g., `/etc/isakmpd/pubkeys/ipv4/`). Roaming clients cannot use the IPv4 identity type, because their IP address will change depending upon their location. We therefore use the UFQDN type, naming the remote client's public key after a UFQDN (e.g., *paul@r500.com*) and putting it in `/etc/isakmpd/pubkeys/ufqdn/`. The identity is also specified in `ipsec.conf` of the roaming client.

Listing 11 is the ipsec.conf on `vpn-gateway1`. Since this is the gateway, we specify *passive*. The *any* keyword is used for the destination, since the roaming client's IP address is not known, The *peer* IP address is not specified; the default is to use the IP address of the connecting machine. Because were using defaults, we don't show the values for phase 1 and phase 2 negotiation.

Listing 12 is the ipsec.conf on the roaming client. The *dynamic* keyword turns on Dead-Peer-Detection. This will cause the IKE peers to exchange informational messages containing Notification payloads which verify the continued presence of the remote peer (these can be seen in the file containing unencrypted IKE traffic). The source is defined as *from egress* (*egress* is a default network interface group name) which translates to the IP address assigned to our NIC. The *to* keyword is the destination network and *peer* specifies the IP address of `vpn-gateway1`. The *srcid* is the identity that the client will send in the Identity payload. This will be the identity which is authenticated.

Now consider the rules for pf.conf on `vpn-gateway1` (see Listing 13). Note that unlike the previous example, we don't need to pass rules for the ESP protocol, since the ESP header will be encapsulated by the UDP header.

The first two rules pass UDP traffic on ports 500 and 4500. The next rule is for tunneled IP traffic (ipencap) on `enc0`, and the last two filter on source/destination addresses on `enc0`.

Listing 14 has the pf rules for the roaming client. Here, the macro `$wifi_if` substitutes for whatever IP address is currently assigned to the wireless interface, `iwn0`. As before, the first two rules allow IKE UDP traffic on ports 500 and 4500. The last three are for the `enc0` interface.

As in the previous example, isakmpd and ipsecctl are executed on both peers. Listing 15 is the (abbreviated) output from the command *systat rules* on `vpn-gateway1`. Shown are the number of packets which matched individual pf rules. Rule 12, the *pass in* rule for port 500, had 4 packets; these were the first 4 packets of the phase 1 exchange. After these, all traffic switched to port 4500 (`ipsec-nat-t`) because NAT was discovered between the peers. This is reflected in the packet count for rule 13, the *pass in* rule for port 4500. The packet counts for rules 16 and 17 also show that traffic is being passed on the `enc0` interface.

Listing 16 is `tcpdump` output for traffic on the external interface of `vpn-gateway1` to port 4500. The two hosts are *abc-internet-cafe.com*, the roaming client's external gateway, and `vpn-gateway1`. This output contains IKE traffic, IPsec traffic and (on the last line) a NAT-T Keepalive. The `tcpdump` run outputs the particular inner protocol (esp or isakmp) after *udpencap*.

Conclusion

This article has been an attempt to provide a high-level overview of IKE and IPsec, to show two functional configurations using OpenBSD, and to explain the contents of IKE and IPsec traffic at the packet level. We specifically covered the use of Encapsulating Security Payload in tunnel mode, showing how an IPv4 datagram is *encapsulated* and encrypted by ESP, and how that traffic is routed before and after *de-capsulation*. We discussed how IKE daemons exchange messages containing payloads in UDP datagrams, and saw at the packet level how IKE negotiates a phase 1 ISAKMP SA in *main mode* (i.e., of the Identity Protection exchange type) for protecting IKE traffic, and how the phase 2 IPSEC SA is established to protect traffic with ESP. Also discussed was a simple configuration for roaming clients. We explained how two peers attempt to detect the presence of NAT and apply NAT-T, encapsulating IKE traffic and ESP packets within UDP datagrams exchanged on port 4500.

There are many more details to IPsec and IKE. The focus here has been to provide an introduction to some basic concepts and give an example of how they are implemented. As always, the OpenBSD manpages are an important resource for additional information – specifically `ipsec(4)`, `ipsec.conf(5)`, `ipsecctl(8)`, `enc(4)`, `isakmpd(8)`; and for more in-depth reading: *Demystifying the IPsec Puzzle* by Sheila Frankel (Artech House, 2001), and *VPNs Illustrated* by Jon C. Snader (Addison-Wesley, 2006).

About the Author

Paul McMath has worked as a Unix admin for over 10 years in Europe and the United States. He became interested in BSD Unix in 2002 after installing OpenBSD on a 32-bit Sparc machine. Current interests include networking and BSD kernels.

HAKIN9

PRACTICAL PROTECTION

SAINT Integrated Vulnerability and Penetration Testing
HAKIN9
PRACTICAL PROTECTION HARD CORE IT SECURITY MAGAZINE
HARDWARE KEYLOGGER A SERIOUS THREAT
HOW TO STAY SAFE
CWNA TRAINING
THE FEAR FACTOR – STUDY OF A NEW GENRE OF MALWARES CALLED "SCAREWARES"
DEBUGGERS – KNOW YOUR ENEMY
SMS TRICKERY IN PUBLIC TRANSPORT
FILE CARVING – TERMINOLOGY, BASICS AND TOOLS
BEHAVIORAL ANALYSIS OF UNWISE_.EXE MALWARE
APPLICATIONS ON THE CD
CERTIFIED WIRELESS NETWORK ADMINISTRATOR TRAINING BY SECURITY.ORG
AD AWARE PRO 4.29 BY LAVASOFT
PLUS
WINDOWS TIMELINE ANALYSIS, PART 3 (ADVANCED & ALTERNATIVE TOPICS) BY HARLAN CARVEY

IT SECURITY MAGAZINE

WWW.HAKIN9.ORG/EN

LDAP on FreeBSD

Eric Vintimilla

Keeping your information synced across multiple systems can be a pain. While there are many ways to ensure consistency in your media and documents (rsync and scp work wonders in this area), there are not too many options for maintaining your address book.

Luckily, there is a common solution for this problem: set up an LDAP server. Whether you maintain a large corporate network or just want to centralize the information for your home network, LDAP will help immensely.

What is LDAP?

LDAP stands for *Lightweight Directory Access Protocol*. It is basically a directory-based storage system that can be used to hierarchically store information. Each entry has a unique identifier, known as its *Distinguished Name* (DN) and a number of named attributes (which hold the information you wish to store).

```
dn: cn=Barbara J Jensen,dc=example,dc=com
cn: Barbara J Jensen
objectclass: person
sn: Jensen
```

This example shows a very basic layout, where the DN consists of the relative distinguished name (CN), which is the person's name, and the *domain component* (DC), which is their domain (in this case, it is *example.com*).

Again, this is a very basic setup. There are many options for the elements in an LDAP directory, and new ones can easily be created.

Installing and Configuring openldap

To start, we are going to install openldap, which is the open source implementation of LDAP, onto our FreeBSD machine. This can be found in FreeBSD's ports tree.

```
[root@moe ~]# cd /usr/ports/net/openldap24-server/
[root@moe /usr/ports/net/openldap24-server]# portinstall -P
```

If you are presented with any installation options, just choose the defaults (unless you want to perform any customizations for your own system).

Next, you will start to personalize your installation. Using your favorite editor, open up the *slapd.conf* file.

```
[root@moe ~]# nano /usr/local/etc/openldap/slapd.conf
```

We are going to make some configurations to get your LDAP directory up and running. Note, that this set up will be relatively insecure, so you will want to add TLS support and probably deny anonymous access to your data, but we will not concern ourselves with that now. Edit your *slapd.conf* so it looks like Listing 1 (you can change the domain to whatever your domain is).

Now, we will test our configuration to make sure there are no errors.

```
[root@moe ~]# sudo  /usr/local/libexec/slapd -Tt
bdb_db_open: warning - no DB_CONFIG file found in
directory /var/db/openldap-data: (2).
config file testing succeeded
```

It looks like we are missing a database configuration file. Fortunately, there is a sample one that we can copy to the correct location.

```
[root@moe ~]# cp /usr/local/etc/openldap/DB_
CONFIG.example /var/db/openldap-data/DB_CONFIG
  [root@moe ~]# /usr/local/libexec/slapd -Tt
config file testing succeeded
```

Next, you will have to edit ldap.conf file.

```
[root@moe ~]# nano /usr/local/etc/openldap/ldap.conf
```

The two options you are concerned with are BASE and URI. Under the BASE heading, you will enter your domain component and under the URI heading you will enter the URLs associated with your LDAP server. See Listing 2 for an example.

Now, the LDAP server is almost fully set up. Unfortunately, there is one blaring security issue that should be fixed. In the slapd.conf file, there is a field called `rootpw`, which shows what the root password to your directory is… in a plain text. Luckily, openldap comes with a way to hide your real password: the slappasswd command. To use it, just type (substituting your desired password for *secret*):

```
[root@moe ~]# slappasswd -s secret
{SSHA}3t7MIHB3VSkuqBZtBs37qXzXnejQan8x
```

Now, type this hash string in the `rootpw` field in you slapd.conf file.

LDAP and Mozilla Thunderbird

This system is almost ready for use. However, it will be very helpful if we can

Listing 1. slapd.conf

```
#
# See slapd.conf(5) for details on configuration
options.
# This file should NOT be world readable.
#
include         /usr/local/etc/openldap/schema/
core.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have
a working directory
# service AND an understanding of referrals.
#referral       ldap://root.openldap.org

pidfile         /var/run/openldap/slapd.pid
argsfile        /var/run/openldap/slapd.args

# Load dynamic backend modules:
modulepath      /usr/local/libexec/openldap
moduleload      back_bdb
# moduleload    back_hdb
# moduleload    back_ldap

# Sample security restrictions
#       Require integrity protection (prevent hijacking)
#       Require 112-bit (3DES or better) encryption
for updates
#       Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
#       Root DSE: allow anyone to read it
#       Subschema (sub)entry DSE: allow anyone to read
it
#       Other DSEs:
#               Allow self write access
#               Allow authenticated users read access
#               Allow anonymous users to authenticate
#       Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema" by * read
 access to *
        by self write
        by users read
        by anonymous auth

# if no access controls are present, the default
policy
# allows anyone and everyone to read anything but
restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

#######################################################
# BDB database definitions
#######################################################

database        bdb
suffix         "dc=example,dc=com"
rootdn          "cn=Manager,dc=example,dc=com"
# Cleartext passwords, especially for the rootdn,
should
# be avoid.  See slappasswd(8) and slapd.conf(5) for
details.
# Use of strong authentication encouraged.
rootpw          secret
# The database directory MUST exist prior to running
slapd AND
# should only be accessible by the slapd and slap
tools.
# Mode 700 recommended.
directory       /var/db/openldap-data
# Indices to maintain
index   objectClass     eq
```

Listing 2. ldap.conf configuration

```
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world
writable.

BASE   dc=example,dc=com
URI    ldap://ldap.example.com ldap://192.168.1.13:389

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never
```

Listing 3. mozillaorgperson.schema

```
# mozillaOrgPerson schema v. 0.6.3
#
# req. core
# req. cosine
# req. inetorgperson

# attribute defs

attributetype ( 1.3.6.1.4.1.13769.2.1.1
        NAME ( 'mozillaNickname' )
        SUP name )

attributetype ( 1.3.6.1.4.1.13769.2.1.2
        NAME ( 'mozillaUseHtmlMail' )
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.3
        NAME 'mozillaSecondEmail'
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.13769.2.1.4
        NAME 'mozillaHomeLocalityName'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )

attributetype ( 1.3.6.1.4.1.13769.2.1.5
        NAME 'mozillaPostalAddress2'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )

attributetype ( 1.3.6.1.4.1.13769.2.1.6
        NAME 'mozillaHomePostalAddress2'
        EQUALITY caseIgnoreListMatch
        SUBSTR caseIgnoreListSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.41 )

attributetype ( 1.3.6.1.4.1.13769.2.1.7
        NAME ( 'mozillaHomeState' ) SUP name )

attributetype ( 1.3.6.1.4.1.13769.2.1.8
        NAME 'mozillaHomePostalCode'
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{40} )

attributetype ( 1.3.6.1.4.1.13769.2.1.9
        NAME ( 'mozillaHomeCountryName' )
        SUP name SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.10
        NAME ( 'mozillaHomeFriendlyCountryName' )
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

attributetype ( 1.3.6.1.4.1.13769.2.1.11
        NAME ( 'mozillaHomeUrl' )
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

attributetype ( 1.3.6.1.4.1.13769.2.1.12
        NAME ( 'mozillaWorkUrl' )
        EQUALITY caseIgnoreIA5Match
        SUBSTR caseIgnoreIA5SubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{256} )

# un-comment for all LDAP server NOT supporting SYNTAX
2.16.840.1.113730.3.7.1
attributetype ( 1.3.6.1.4.1.13769.2.1.13
        NAME ( 'nsAIMid' )
        DESC 'AOL Instant Messenger (AIM) Identity'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )

attributetype ( 1.3.6.1.4.1.13769.2.1.14 NAME (
'mozillaHomeStreet' )
        EQUALITY caseIgnoreMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{128} )

attributetype ( 1.3.6.1.4.1.13769.2.1.96
        NAME ( 'mozillaCustom1' )
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.97
        NAME ( 'mozillaCustom2' )
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.98
        NAME ( 'mozillaCustom3' )
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.99
        NAME ( 'mozillaCustom4' )
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE )

attributetype ( 1.3.6.1.4.1.13769.2.1.100
        NAME ( 'mobile' 'mobileTelephoneNumber' )
        DESC 'RFC1274: mobile telephone number'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
```

just import an LDIF file from our email client (in this case, Mozilla Thunderbird). This process is more involved than the previous steps, but it will save a lot of time (and headaches) in the long run. First, you will have to create a schema

Listing 3. mozillaorgperson.schema

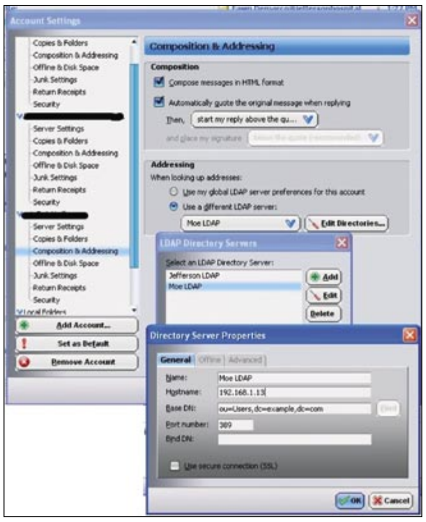
```
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )

attributetype ( 1.3.6.1.4.1.13769.2.1.101 NAME 'pager'
        DESC 'RFC2256: Telephone Number'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50{32} )

attributetype ( 1.3.6.1.4.1.13769.2.1.102
        NAME ( 'homePhone' 'homeTelephoneNumber' )
        DESC 'RFC1274: home telephone number'
        EQUALITY telephoneNumberMatch
        SUBSTR telephoneNumberSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.50 )

# objectClass defs

objectclass ( 1.3.6.1.4.1.13769.2.2.1
        NAME 'mozillaOrgPerson'
        SUP top
        AUXILIARY
        MAY (
        sn $
        givenName $
        cn $
        mozillaNickname $
        title $
        telephoneNumber $
        facsimileTelephoneNumber $
        mobile $
        pager $
        homePhone $
        street $
        postalCode $
        mozillaPostalAddress2 $
        mozillaHomeStreet $
        mozillaHomePostalAddress2 $
        l $
        mozillaHomeLocalityName $
        st $
        mozillaHomeState $
        mozillaHomePostalCode $
        c $
        mozillaHomeCountryName $
        mozillaHomeFriendlyCountryName $
        ou $
        o $
        mail $
        mozillaSecondEmail $
        mozillaUseHtmlMail $
        nsAIMid $
        mozillaHomeUrl $
        mozillaWorkUrl $
        description $
        mozillaCustom1 $
        mozillaCustom2 $
        mozillaCustom3 $
        mozillaCustom4 ) )
```

Listing 4. Our LDAP structure.

```
# Organization
dn: dc=example,dc=com
objectClass: dcObject
objectClass: organization
dc: example
o: Name Of Organization
description: Description of Organization

# Organizational Role for Directory Manager
dn: cn=root,dc=example,dc=com
objectClass: organizationalRole
cn: Manager
description: Directory Manager

dn: ou=Users,dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: Users
description: This is the tree were user accounts are
stored
```

Listing 5. Thunderbird LDIF file.

```
# 1st User Entry
dn: uid=username1, ou=users, dc=example, dc=com
objectClass: inetOrgPerson
uid: username1
userPassword: userpass1
cn: Johnny FreeBSD
givenName: Johnny
sn: FreeBSD
title: Title of user
mail: johnnyfreebsd@example.com
telephoneNumber: none
homePhone: 610 555-1212
homePostalAddress: 123 First St.
facsimileTelephoneNumber: none
pager: none
mobile: none
o: Acme Inc
l: Anytown
st: PA
postalAddress: 54321 Street
postalCode: 19380
description: Additional notes go here
```



Composition & Addressing

Composition

[✓] Compose messages in HTML format

[✓] Automatically quote the original message when replying

Then, start my reply above the qu...

Addressing

When looking up addresses:

( ) Use my global LDAP server preferences for this account

(●) Use a different LDAP server: Moe LDAP

LDAP Directory Servers

Select an LDAP Directory Server: Jefferson LDAP, Moe LDAP

Directory Server Properties

Name: Moe LDAP

Hostname: 192.168.1.13

Base DN: ou=Users,dc=example,dc=com

Port number: 389

Bind DN:

[ ] Use secure connection (SSL)

Figure 1. Thunderbird Configuration

file for Thunderbird's LDIF structure. Its contents can be seen in Listing 3.

```
   [root@moe ~]# nano /usr/
local/etc/openldap/schema/
mozillaorgperson.schema
```

Once you have created the schema file, you will have to add it to your configuration file. Add the following lines to your *slapd.conf* file:

```
include         /usr/
local/etc/openldap/schema/
mozillaorgperson.schema
include         /usr/local/etc/
openldap/schema/cosine.schema
include         /usr/local/etc/
openldap/schema/inetorgperson.schema
```

Once these changes have been made to the configuration file, we can test it again to be sure. You may receive an error that says *Inconsistent duplicate* for the `homePhone` and mobile attributes.

```
[root@moe ~]# /usr/local/libexec/
slapd -Tt
/usr/local/etc/openldap/schema/
cosine.schema: line 637 attributetype:
Inconsistent duplicate attributeType:
"mobile"
slaptest: bad configuration file!
```

If you see this message, then edit `/usr/local/etc/openldap/schema/cosine.schema` and comment out the offending attribute declarations.

```
#attributetype ( 0.9.2342.19200300.1
00.1.41
#       NAME ( 'mobile'
'mobileTelephoneNumber' )
#       DESC 'RFC1274: mobile
telephone number'
#       EQUALITY telephoneNumberMatch
#       SUBSTR telephoneNumberSubstr
ingsMatch
#       SYNTAX 1.3.6.1.4.1.1466.115.1
21.1.50 )
```

We are finally ready to define the basic structure of the directory. Listing 4 shows what we will use. To import it, we will use the slapadd command.

```
   [root@moe ~]# slapadd -l initial3.ldif
_################### 100.00% eta
none elapsed            none fast!
Closing DB...
```

We can now start to add our Thunderbird LDIF file. In this example, we will be importing the contents of Listing 5. Again, use the slapadd command to add the LDIF file.

```
[root@moe ~]# slapadd -v -l
mozilla.ldif
added: "uid=username1,ou=users,dc=exa
mple,dc=com" (00000004)
_################### 100.00% eta
none elapsed            none fast!
Closing DB...
```

Next, start up the Stand-alone LDAP daemon. You can use sockstat to make sure it is running.

```
[root@moe ~]# /usr/local/libexec/slapd
[root@moe ~]# sockstat -4 -p 389
USER     COMMAND    PID   FD PROTO
LOCAL ADDRESS           FOREIGN ADDRESS
root     slapd      11891 7  tcp4    *:
389                     *:*
```

Now that the daemon is running, we can perform a search to make sure our data was properly imported. The ldapsearch command will output the recently imported information.

```
[root@moe ~]# ldapsearch -x -b
'dc=example,dc=com' '(objectclass=*)'
```

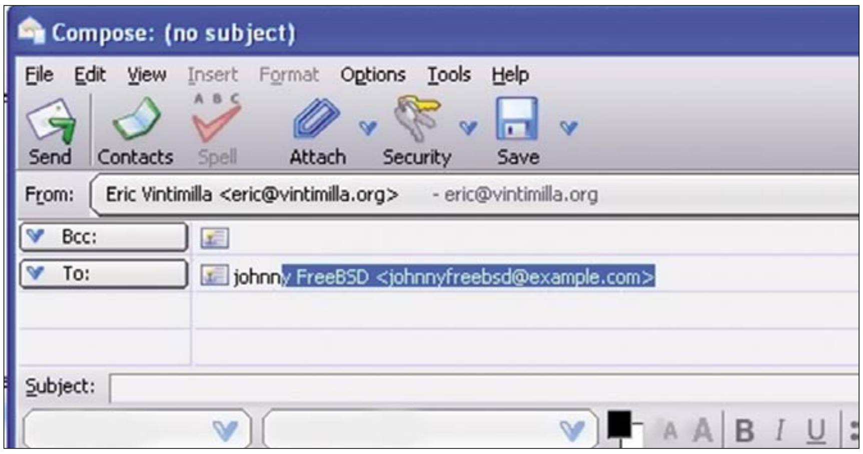
Do not forget to add the following to your rc `.conf` file, so your LDAP server will automatically restart if you reboot your system:

```
   slapd_enable="YES"
```

Conclusion

Congratulations! You now have an LDAP server where you can centrally store your contact information. Now you have this directory of information, but how can you access it (besides the command line)? You can easily set up LDAP directory searching in Mozilla Thunderbird! Just right-click on your e-mail account in the folder listing on the left side, then click *Properties*. Click on *Composition & Addressing* and then select the *Use a different LDAP Server* option. Click the *Add* button and fill in your LDAP server's information. Now, whenever you start to type in an email address in a message's *To:* field, it'll autocomplete using your LDAP data!

Once you have set up all of your e-mail clients to use your new LDAP server, you will never have to worry about syncing your contacts between multiple machines again! You will only have to maintain the information in your LDAP directory (phpldapadmin can help you in this area). In the long run, this data storage will help save you time and headaches, and it will be well worth the initial effort!



From: Eric Vintimilla <eric@vintimilla.org> - eric@vintimilla.org

Bcc:

To: johnny FreeBSD <johnnyfreebsd@example.com>

Subject:

Figure 2. Composing a new message

Install your server in our colocation center

Standard-size 1U server

- 99.99 percent server uptime guarantee in writing
- Fully redundant BGP6 network throughout our data center
- 100 mbps port speed on all our colocation plans
- From 1U to full-size racks (42U) and ½ racks available as well
- The best price in the industry with the most reliable technical support
- Data center manned by security guards 24 hours a day, 365 days a year
- Fully redundant UPSs and Caterpillar diesel power generators
- Free reboots and free visits to the data center 24 hours a day, 365

Get the best deal on a reliable dedicated server

- We offer Dell or HP dedicated servers
- Your choice of Linux or Windows
- 99.99 percent uptime guarantee
- Raid 1 servers also available with SAS70 certification
- Your choice of cPanel, Plesk or no control panel if you prefer
- All servers offer Intel CPUs and Azus mother boards

Tell us that you saw our ad in BSD Magazine and get a 10% price discount!

sunhosting™

International: 1-514-630-1153
US & Canada, toll free: 1-800-547-4149
Our motto is "Host Only Once"

www.sunhosting.ca
support@sunhosting.ca

Sun Hosting was established in 1995 and is one of the largest hosting company in Canada.
PRM-BSDMAG

Secure and stable mailservers with OpenBSD and qmail

Matthias Pfeifer

Secure and stable email servers are important for everyone who is using email. Most of the communication in companies is done by sending and receiving emails. So, reliable email systems are very important for each Internet company.

Many companies sell special email systems and antispam gateways which are reliable and designed to be secure. In fact, many of these email systems are running Linux or BSD with one of the popular MTAs on top.

You can build your own email system. And, of course, this could be reliable, stable and secure too.

I have installed many email systems on many different machines. Over the time, I found that systems with OpenBSD and qmail work best for me and my customers.

My customers always want stable, secure and highly flexible email systems.

From my point of view, it is also very easy to manage and upgrade the following email system. With this article, you get a very extensible and reliable email system without the need to buy user licenses or software.

The MTA we will use is qmail. Qmail was developed by Daniel J. Bernstein in 1995. In 1995 there were a very few alternatives MTA available. The most common MTA was sendmail, which has many security problems. Bernstein followed a new way with this MTA: modularity. Bernstein also designed qmail for security and speed. No-one has found a security flaw in qmail since 1997. See References for more information.

For managing virtual hosting and non /etc/passwd user accounts, we will use vpopmail. Vpopmail makes the handling of users and domains in our qmail installation very easy. You will need virtual domains if you plan to add more than one domain to your email system.

We will also use some other software like daemontools and ucspi, also developed by Daniel J. Bernstein.

The rest of this Article is a how-to. You can follow this installation by copying and pasting the posted commands.

Please do not forget to buy an OpenBSD CD set or at least donate.

Lets start!

I assume that you have a fresh OpenBSD installation. Make sure that you have enough disk space under `/var`, `/tmp` and `/usr`. See the following disk layout for example:

```
/dev/sd0a       2.0G      52.1M  1.8G     3%    /
/dev/sd0e       3.9G       2.0K  3.8G     0%    /tmp
/dev/sd0f       39.4G      457M  37.0G    1%    /usr
/dev/sd0g       1.7T       2.6M  1.6T     0%    /usr/home
/dev/sd0d       19.7G     27.4M  18.7G    0%    /var
```

Of course, this is a very large system. In OpenBSD, `/var` is mounted with the nosuid flag by default. We have to change this if we want to get qmail running properly. In qmail, there is just one program which is using setuid – qmail-queue. Two other programs are running as root: qmail-start as qmail-lspawn.

All qmail binarys will be installed in `/var/qmail/bin`. Change the entry in fstab from

```
/dev/sd0e /var ffs rw,nodev,nosuid 1 2
```

to

```
/dev/sd0e /var ffs rw,nodev 1 2
```

Reboot the machine or remount the `/var` partition.

Get the Software

Load the following tarball which contains all software packages you will need for the installation. Create a directory and fetch the package:

```
mkdir /qmail
cd /qmail
```

```
ftp http://www.freshmail.de/mailgate/
package.tgz
tar xvzf package.tgz
```

Start the `prepare.sh` script. The script will do the following things:

- create directories
- create users and groups
- unpack and move software packages
- patch qmail

Just execute `prepare.sh`:

```
sh prepare.sh
```

Now, lets build qmail

The system is prepared and we are ready for installing qmail itself:

```
cd /usr/source/qmail/qmail-1.03
make man
make setup check
./config-fast YOUR_DOMAIN_NAME
```

Change `YOUR_DOMAIN_NAME` to the name of your mail system. For example *mail.your-company.com*. Next, we need to install `ucspi-tcp`. This package contains the program tcpserver. tcpserver works like the inetd superserver, but has more features. We will use the program for managing incoming connections on port 25.

```
cd /usr/source/qmail/ucspi-tcp-0.88/
make
make setup check
```

We also need daemontools. This package contains the programs supervise and multilog. Supervise is used for monitoring qmail processes. If an important process dies, the supervise process takes note and tries to restart the process. Multilog is used for logging. Installing daemontools:

```
cd /usr/source/qmail/admin/
daemontools-0.76
sh package/install
```

Reboot the server. It is the easiest way to check all the steps we have completed.

After the reboot, log into the server and look for qmail like this: see Listing 1.

You should see some qmail related processes running. Check for the readproctitle service like this:

```
ps -waux | grep read
root      3140  0.0  0.0   304
400 ??  I      5:34PM    0:00.00
readproctitle service errors: .......
....................................
```

readproctitle is a part of daemontools. It maintains an rotated log in memory. This log can be inspected with ps. When you see this line of dots, qmail is working well. Otherwise there will be error messages in the output of readproctitle.

Installing vpopmail

Now, that qmail is up and running we can install vpopmail. You can decide to use a MySQL database for running vpopmail. Using MySQL as a user data back-end helps you to extend the system later. Please note that only data which relates to a user account is stored in the database. Emails are still stored in the file system. In this case I will show you how to install vpopmail by using MySQL.

Installing MySQL on OpenBSD

Set up the `PKG_PATH` environment variable. For example:

```
export PKG_PATH="ftp://
ftp.openbsd.org/pub/OpenBSD/4.6/
packages/i386/"
```

Install the Server:

```
pkg_add mysql-server
```

You now need to follow the instructions in `/usr/local/share/doc/mysql/README.OpenBSD` and you are done.

Vpopmail and MySQL

MySQL is installed and running. It is time to set up vpopmail's MySQL configuration. Create the needed directory and set user rights:

```
 mkdir ~vpopmail/etc
 chown vpopmail:vchkpw ~vpopmail/etc
```

Create the configuration file for connecting the MySQL database and set up the user rights. Insert your vpopmail `USER` and `PASSWORD` here.

```
 echo "localhost|0|VPOPMAILUSER|PA
SSWORD|vpopmail" > ~vpopmail/etc/
vpopmail.mysql
 chown vpopmail:vchkpw ~vpopmail/etc/
vpopmail.mysql
 chmod 640 ~vpopmail/etc/
vpopmail.mysql
```

Listing 1. qmail related processes

```
ps -waux | grep qm
root      21393    0.0   0.0    364    512  ??  I      5:48PM    0:01.71 supervise qmail-send
root      6655     0.0   0.0    308    512  ??  I      5:48PM    0:00.49 supervise qmail-smtpd
root      31989    0.0   0.0    360    508  ??  I      5:48PM    0:00.92 supervise qmail-pop3d
qmaill    25905    0.0   0.0    424    512  ??  I      5:52PM    0:00.02 multilog t s100000 n20 /var/log/qmail/qmail-send
qmails    2029     0.0   0.0    340    752  ??  I      5:52PM    0:00.06 qmail-send
qmaill    17911   0.0   0.0    276    512  ??  I      5:52PM    0:00.02 multilog t s100000 n20 /var/log/qmail/qmail-smtpd
vpopmail 30315    0.0   0.0    324    752  ??  I      5:52PM    0:00.01 /usr/local/bin/tcpserver -v -R -l mail.your-
domain.com (...)
qmailr    5584     0.0   0.0    248    576  ??  I      5:52PM    0:00.01 qmail-rspawn
qmaill    8636     0.0   0.0    416    512  ??  I      5:52PM    0:00.01 multilog t s100000 n20 /var/log/qmail/qmail-pop3d
root      21233   0.0   0.0    408    556  ??  I      5:52PM    0:00.01 tcpserver -H -R -v -c100 0 110 qmail-popup
(...)
qmailq    13451   0.0   0.0    228    668  ??  I      5:52PM    0:00.01 qmail-clean
root      13990   0.0   0.0    424    616  ??  I      5:52PM    0:00.01 qmail-lspawn ./Maildir
```

Log into your MySQL database as root and execute the following SQL statement:

```
 CREATE DATABASE vpopmail;
 GRANT select,insert,update,delete,cr
eate,drop ON vpopmail.* TO
 VPOPMAILUSER@localhost IDENTIFIED BY
'PASSWORD';
```

Use the same vpopmail `USER` and `PASSWORD` as specified in the configuration.

Now the vpopmail database and the vpopmail user have been created. We can now build vpopmail:

```
cd /qmail/source
tar zxvf vpopmail-5.4.27.tgz
cd vpopmail-5.4.27
```

I recommend the following configuration. Feel free to customize it to your needs. Just type `./configure` -–help to see the configuration options.

```
./configure --enable-logging=p --
enable-auth-module=mysql
--disable-passwd --enable-clear-passwd
--disable-many-domains
--enable-auth-logging --enable-sql-
logging --enable-valias
--disable-mysql-limits

make && make install-strip
```

If you get no errors here then vpopmail should be installed correctly.

Finishing the installation

Run the following script to check your installation. It will tell you if something is wrong.

```
cd /qmail
sh finish.sh
```

The last steps must be done by hand. Open the following files, search for `mail.example.com` and change it to the name of your mail system:

```
/var/qmail/supervise/qmail-pop3d/run
/var/qmail/supervise/qmail-smtpd/run
```

Stop qmail and enable selective relaying:

```
qmailctl stop
echo '127.:allow,RELAYCLIENT=""' >>
/etc/tcp.smtp
qmailctl cdb
```

Go to the `/qmail` directory and run the script below:

```
cd /qmail
sh default_user.sh
```

Add a default user here. I recommend the user *postmaster@your-domain.com*.

Note: Do not forget to add you domain to the system by using vpopmail (See *vpopmail, adding a domain* in this article).

Since sendmail is the default MTA under OpenBSD, we have to deactivate it. Kill all running sendmail processes:

```
pkill -9 sendmail
```

Then move the original sendmail binary:

```
mv /usr/sbin/sendmail /usr/sbin/
sendmail.old
chmod 0 /usr/sbin/sendmail.old
echo sendmail=NO >> /etc/rc.conf.local
```

qmail ships with a sendmail compliant sendmail binary. Link the qmail sendmail to the place of the original OpenBSD sendmail:

```
ln -s /var/qmail/bin/sendmail /usr/
sbin/sendmail
ln -s /var/qmail/bin/sendmail /usr/
lib/sendmail
```

There is also a crontab entry for qmail. Run contab -e and delete the sendmail line. If you are done, start `qmail`:

```
qmailctl start
```

Run the following script to check your installation:

```
sh /qmail/check_installation.sh
```

If the script says it is ok, congratulations. You now have a OpenBSD system with qmail and vpopmail running.

Using vpopmail

Vpopmail is used to manage virtual domains and users

- vpopmail programs can be found in `/usr/home/vpopmail/bin`
- domains and users are stored in `/usr/home/vpopmail/domains`

Adding a domain

```
cd /usr/home/vpopmail/bin
./vadddomain my-domain.com MY_PASSWORD
```

The user postmaster is created automatically.

Adding a user

```
cd /usr/home/vpopmail/bin
./vadduser new-user@my-domain.com
USER_PASSWORD
```

Please read *http://www.qmailwiki.org/Vpopmail* for more information about vpopmail and its programs.

Congratulations! Your email system is running. The installation was easy, but don't underestimate the management of email systems. If you are new to Internet email servers, you should learn to control your system. Take your time and play around with all the configuration options. Use the references for information about the installed software. In following articles I will show you how to extend this installation with Dovecot (IMAP) and qpsmtpd.

On the 'Net

- OpenBSD installation – *http://www.openbsd.org/faq/faq4.html*
- qmail website – *http://www.qmail.org/*
- qmail configuration options – *http://www.lifewithqmail.org/lwq.html#configuration*
- vpopmail Website – *http://www.inter7.com/index.php?page=vpopmail*
- Using vpopmail with qmail – *http://www.qmailwiki.org/Vpopmail*
- UCSPI-TCP – *http://cr.yp.to/ucspi-tcp.html*
- daemontools – *http://cr.yp.to/daemontools.html*
- Author's Website – *http://www.freshmail.de*

About the Author

Matthias Pfeifer works as a system administrator at an internet company and for freshmail.de.

BSDCan 2010
The Technical BSD Conference
High value. Low cost. Something for everyone.

BSDCan, a BSD conference held in Ottawa, Canada, has quickly established itself as the technical conference for people working on and with 4.4BSD based operating systems and related projects. The organizers have found a fantastic formula that appeals to a wide range of people from extreme novices to advanced developers.

BSDCan 2010 will be held on 13-14 May 2010 at University of Ottawa, and will be preceded by two days of Tutorials on 11-12 May 2010.

There will be related events (of a social nature, for the most part) on the day before and after the conference.

http://bsdcan.org/

BSDCan 2010

Developing Secure Storages: Now On FreeBSD

Theodore Tereshchenko, EldoS Corp.

Developers of server-side, desktop and mobile applications working with FreeBSD now get access to Solid File System – a well-known component designed by EldoS Corporation. FreeBSD-developers have an ability to store documents and files in a highly secure robust and flexible file system with no run-time fees. Clean room implementation allows royalty-free business applications.

Are you developing an in-house application? Working on a code for sale? You must ensure a convenience of work with files that are in use. This applies to files processed by software application you develop, as well as to auxiliary files your application uses or creates. As a thoughtful software architect, you are taking care about organization of correct approach to file management from your application, and also about providing a convenient secured storage for them.

For years, Solid File System is successfully used by developers to achieve these goals. Following numerous requests from FreeBSD-developers community, EldoS Corporation has ported the product for this popular platform. Moreover, after appearance of the Standard Edition of Solid File System for FreeBSD, the time of Driver Edition came out as well.

Solid File System is primarily oriented for use in applications located either on a server, desktop or a mobile platform, operating with binary files or compound objects. For example, if your application operates with documents or collections of documents, business or other user-defined entities, object models or classes, formatted text or raw binary data of variable size – Solid File System is worth taking a look at. You will benefit from exploring Solid File System capabilities if you develop software of any of the following types:

- Business and corporate software
- Applications for PDA, handheld computers and mobile phones
- Data backup and archiving software
- 3D graphics/design/engineering/science applications
- Document management systems, document storage systems
- Secure video and other media archives
- Internet and intranet applications

Business and corporate software

According to National Law Journal (2006), about 90% of business documents are stored in electronic form, while between 60 and 70% of corporate data reside in e-mails or in attachments. Volumes can reach terabytes...Compliance with current government regulations (eDiscovery in the US, analogous laws elsewhere) is a must for companies who can not afford significant downtime due to any routine investigations.

The most critical aspects of file and document processing in corporate environment are providing security of sensitive information, protection against loss of valuable information, preservation of business documents integrity, management of documents access permissions, reliability and ease of access to requested data by end-users.

Use of DBMS (*database management system*) is justified only for homogeneous data arrays that can be ordered by a chosen characteristic. In some cases binary data can also be ordered accordingly and stored in a database. In the remaining cases, when your application should manage data in form of files or compound objects, keeping them in a regular database is not rational.

From the very nature of files follows the use of file system as a storage: ensuring easy access and other necessary functionality. But modern file systems emphasize read/write speed, sacrificing at the same time security, and, in some cases, – reliability.

Standard file systems are designed as general-purpose file storages. One size does not fit all, as well as there is no single file system suitable for all types of applications. If an application that you develop creates or receives files for processing or storage, it is better to base it upon the file system developed especially for this purpose.

Experienced developers starting to use Solid File System usually mention its flexibility and ease of use for applied tasks, comparing to standard file systems. On the other hand, novices learn the API quite easily: the methods and syntax are the same as in standard Windows API and there is no need to apply extra efforts to use the extended functionality.

To demonstrate the ease of use of Sold File System API take a look at the code example demonstrating encryption of a file in storage. To do this you only need to add this simple code:

```
StorageSetFileEncryption(storage,
filename, ecAES256_HMAC256, old_
password, old_password_length, new_
password, new_password_length);
```

where `old_password` is current encryption password if the file is encrypted or the empty string otherwise, `new_password` is the password to be used with new encryption mode (if any).

If you want to use your own encryption algorithms – you can implement it in a just few steps. Replace correspondent `StorageSetFileEncryption()` parameter in the example above from `ecAES256_HMAC256` to ecCustom256, then add a code to process `OnDataEncrypt()`,`OnDataDecrypt()`,`OnHashValidate()` and `OnHashCalculate()` events correspondingly.

Solid File System allows you to assign user-defined attributes and tags to stored files. This dramatically decreases your efforts to implement ordering, search and further processing code within your application context.

Moreover, if you develop a cross-platform application, then use of such a cross-platform components can be a wise decision. Besides FreeBSD version, EldoS released versions for Windows, Windows Mobile, Linux, MacOS X. The kernel of Solid File System is written in ANSI C. This allows an easy port to other software and hardware platforms.

ELDOS

Feel safer

Applications for PDA, handheld computers and mobile phones

Specific design of handheld hardware and mobile phones dictates peculiarities of data handling. The main problems are data protection from power failures, ensuring data security from loss or theft, and compensating for relatively low speed of handheld devices hardware.

Solution of these problems is mainly responsibility of hardware manufacturer, but, nevertheless, during development of applications for PDA handheld computers and mobile phones you must take these problems into account.

First, these devices can unexpectedly loose power. This is not an ordinary situation, but still occurs regularly. A user of your application will hardly be happy to loose data because of this cause, insignificant from his point of view. Therefore, this factor of partial information loss must be taken into account during development of an application. The measures should be taken to ensure data integrity. It is nice when a file storage that you use helps you to do this.

For example, during creation of Solid File System storage you can set the value of parameter `usertransactions` to *true*, launching, therefore, a mechanism of operations journaling. In this case an integrity of files and whole storage is insured, even in case of an outside failure, by recording each file operation processed within transactional frame.

Loss of a mobile device is not a rarity either. One can not be sure that it always will end up in hands of an honest person. If your application works with data that can be considered sensitive – make sure that your secrets are well protected from unauthorized access. The easiest way to do so is to use encryption algorithms. No, you will not have to implement them yourself: Solid File System supports transparent strong encryption on both per-stream basis and encryption of the whole storage.

Solid File System may play a role of a superstructure over existing file system (for example, creating a file system inside a single file of other file system), or be a fully fledged replacement of standard file system – by processing of corresponding events of sector-by-sector read and write. Therefore, by implementing optimal algorithms of events processing for required type of memory you will get a fully-fledged royalty-free file system.

Data backup and archiving software

There are 7 traditionally defined tiers of data recovery: from regular data backup without a hot site, up to the highest level – highly integrated automated data backup solutions. For every tier of data recovery Solid File System may be of significant help to you.

For the reserve copying purposes, it is very convenient to place data into a Solid File System based storage. All documents will be conveniently stored in one file. There is no need to rewind the tape searching for a specific document – the whole storage can be quickly restored.

But what is even more important, according to Computer Crime and Security Survey about 44% of US companies face an attack against their servers each year. A survey published by Ponemon Institute LLC in 2008 (The 2008 Annual Study: Cost of a Data Research) states that an average loss related to malicious attack against business amount about 6.6 million US dollars per company. These numbers are the justification for paying additional attention to data security and protection by encryption.

Therefore, the fact that Solid File System has built-in cryptographic protection, allows you to entrust tape storage to almost any third-party service provider without risk of information leaks. In this case the keys or passwords used for encryption should be kept separately from backups. A loss of such key will not effect feasibility of storage restoration, but will make access do stored data impossible.

Solid File System also allows you to use incremental backup systems working on the sector-by-sector basis: you will not have to update the whole storage file when minimal changes have been made to the data. Practicability of this approach depends on the frequency of stored file changes, i.e. on the specific application. The advantage of reserve copying whole storages is that the backup system does not need to know the internal structure, encapsulation level, or directory tree of the storage. The whole storage will be copied without possibility of loss of a single file attribute.

In addition, Solid File System supports native data compression. If your Solid File System storage contains data susceptible to compression, use of Solid File System for whole storage compression is much more time- and cost-effective than use of regular compression tools applied to separate files or folders. Solid File System based storages use journaling for self-integrity checks. If a part of a tape or sector on disk becomes physically damaged and unreadable, the whole storage, save the damaged file(s), remains intact and functional. You also can backup separate files from your Solid File System storage, if necessary. Solid File System Driver Edition allows making access to your storage the same way you access regular files and folders from the application of reserve copying or any other application. This also makes possible development of a monitoring tools watching the changes made to files inside a Solid File System storage and exporting them in any convenient format for reserve copying or any other manipulations.

Naturally, the restoration of a whole Solid File System storage takes more time than a single file, but, as a result, you are getting the whole working storage with all files inter-dependencies and directory content preserved. Such data-restore operation can be executed by less qualified personal than that required for a full manual re-assembly of storage structure.

In addition, use of storage based on Solid File System makes possible easy separation of storage back-ups from operating system back-up procedures: quickly restore your storage independently from software operation environment.

3D graphics/design/engineering/ science applications

Development of additional enhancements for an existing application or even development of a full-fledged application for 3D graphics processing, industrial engineering and design are very challenging tasks. In addition to non-trivial algorithms of application itself, you need to take care of storage of logical structures, models and objects processed by your application. Even if an internal structure of the objects is pre-defined, it is often necessary to keep them away from end-user direct access.

A good approach to these problems is to use a securely protected storage or several storages with robust authentication system and access delimitation. Therefore, by using Solid File System based storages you can focus your efforts on development of major functionalities of your application. Moreover, if there will be a need to make run-time access to files and documents in a storage from a third-party application – you can always count on Driver Edition. This edition of allows you to create a controlled virtual drive from a storage. This drive can be mounted as a local drive. Moreover, access to this storage will not be different from that to real local files.

Transparent compression on per-stream basis allows you to keep objects developed by you maximally compacted. The compression is done absolutely automatically and does not require any additional compression algorithms: its is done on the level of storage file system. Thus, a minimal size of objects is maintained, which is especially important when an object contains a lot of text information.

Huge size of data processed by your application will not be a problem for distributed storage based on Solid File System. This storage can function on different computers and even under different operating systems. You will jut have to implement sector-by-sector read/write callback functions. All other aspects of storage operations will be unchanged and no modifications of software code will be necessary.

Scientific data are no different. Just to give you an example, it s expected that the Hadron Collider to be launched by CERN will generate terabytes of diverse data. Millions of the particles trajectories with mass, charge and other data need to be sorted through and stored. The heterogeneous nature of cluster-based storage does not allow use of more traditional storage solutions. Solid File System based storage will provide fast access, transferability and integrity of all expensive and expansive scientific data.

Virtual drug lead library are universally used in the filed of molecular medicine and drug design. They are models of chemical compounds that are used for virtual screening during searches for potential novel drugs. A single virtual library may contain millions of compound models that need to be screened according to specific criteria. The storage, again, is usually highly distributed. The libraries contain patented information and company know-hows, and, therefore serious security measures must be taken to protect them. Solid File System allows creation of secure storages that, at the same time, can be easily and quickly accessed via standard command line FreeBSD interface and therefore can be analyzed and searched through with conventional FreeBSD batch scripts.

Document management systems, document storage systems

Developers of document storage subsystems or developers of document management systems plan their application on the basis of chosen file storage method. Some of them use databases to store documents and their versions, some use features provided by file systems. While pros of file system comparing to databases are evident, the cons are limitations of file attributes, logic of their ordering, weak protection of file information from unauthorized access.

During Solid File System development we removed above-mentioned short-comings of ordinary file systems, and, therefore, by using storage based on Solid File System your application will be able to preserve all range of data necessary for description of your documents in user-defined file attributes. Ability to use build-in tags on the level of file system extends existing hierarchical folder structure and use of symbolic links. Attribute- and tag-based search is also implemented on the file system level.

Multi-stream access to Solid File System files allows simultaneous work with files of different users, which is a must for modern DMS systems. Moreover, a number of simultaneously used storages is also unlimited.

Protection of data in storages build on Solid File System is secured by modern reliable encryption algorithms, such as AES and SHA (HMAC) algorithms with 256-bit key. Additionally, as a developer you can determine whether you will apply encryption to a whole storage, to a file or even to single stream within a file.

Document archiving imposes additional requirements on economical use of storage space allocated to documents. Solid File System has built-in ability to compress files on-the-fly with Zip algorithm so that you would not have to worry about saving your storage space.

Journaling and special recovery functions of storages based on Solid File System ensure high probability of full recovery from serious software and hardware failures. Naturally, only reserve copy can help in case of complete failure of memory device where the storage is kept. But even in this case standard functionality of Solid File System will simplify task of preserving document integrity.

An ability to place storages not only on local disks, but also on remote servers, in memory, inside database records and on custom devices will also be of use for you. By implementing callback functions of sector-by-sector read/write operations you do not have to restrict yourself while selecting place for actual data storage. Your application will be able to work with storages placed locally, as well as with remote storages. And you will not even have to change the code of your software application.

Secure video/document archives

During the last decade, volume of industrial video and audio recordings has been growing exponentially. IDC, a leading market research firm, estimates that total amount of stored digital information is around 2.8 exabytes (2.8 million terabytes) with tenfold expected growth every five years! All this huge amount of information needs to be stored and retrieved upon demand.

Moreover, author rights protection is a requirement: its is important to secure media archives against unauthorized access. Currently, many solutions for implementation of media-archives storage do not address this problem, or it is often approached through administrative restrictions. This can be explained by the fact that encryption of huge files requires significant time and consumes twice the storage space amount. To add insult to injury, when user needs just a small fragment from the inside of the file – the whole file should be decrypted, searched for the fragment, played back and then the unencrypted file must be deleted.

If you use Solid File System based storage, everything gets much more simple, since encryption/decryption operations are made on-the-fly, imperceptibly to the application. Therefore, to encrypt a media file you just needs to copy it to the storage. Encryption operation will be performed automatically. In order to play this record back, it is necessary to jump to record beginning and start playback. The decryption operation will be done automatically.

Special requirements for the speed of retrieval exist in TV industry, where a specific story must be quickly found by its description and other metadata and delivered to the mixing studio. As for many other applications, reliability and impenetrability are very important. The problem is complicated by a fact that many studios have huge legacy tape storages, while transfer to new media is slow due to budgetary considerations.

Solid File System makes possible creation of huge but easily accessible storages (maximum storage size of 256 terabytes… just ask EldoS if you need more) for video files. Any material can be quickly retrieved by its metadata, which are user-defined and can hold all necessary descriptive information. Legacy storages can be integrated into the system without expensive transfer from tapes to modern media.

Internet and intranet applications

The main ideology behind Internet and intranet application is that data used by these applications are stored on remote servers located away from end-users. Therefore, the problem of control over storage data, ensuring their integrity and confidentiality, becomes extremely important.

Everybody has heard about hacker attacks on Internet servers, stealing passwords and other user data. Little attention is paid to the fact that such an attack is often done either on the level of server operating system or on additional web-services – not on a level of applied end-user software. Meanwhile, this is an extremely important moment: when a hacker gets access to files he can get access to sensitive information only if files are stored unencrypted.

To put it in other way, if your Internet/intranet application stores user information in regular files, it is advised to think about securing the files against outside malicious attacks. Therefore, by using Solid File System for creation of secured file storage you continue working with files in your application, plus make them secure. Even by getting access to the storage, intruders will not be able to get access to files inside it. Users of your application will certainly appreciate it.

If you are working on one of the discussed applications or need to implement similar storage functions – considering Solid File System is the right step. You can learn more from the EldoS Corporation website *http://www.EldoS.com/solfs/*.

Web Server Benchmarking

Mikel King

I cannot lie; I had an extremely difficult time determining what to write about for this issue of the magazine.

Since this issue centers on the topic of servers, I found it a particularly hard since there has been at least one article about a server project or application in every issue. It's a topic that we've covered more or less throughout the entire span of the magazine. After considerable meditation I was fortunate enough to settle on the topic of Web Server Benchmarking, one that would not be a rehashing of something we've already talked about.

Have you ever built up an application on a server where initially performance was good however after you bring several other programmers onboard the application grew exponentially over night? When you got out to the beta testers things start falling apart. Programmers start pointing fingers at other departments. The hardware guys shrug, saying we wanted to deploy on the latest and greatest hardware but it wasn't in the budget. Management is on the war path because, let's face it, they're just management and generally have no clue. More importantly they don't want to buy the most expensive piece of big iron because it will cut into their budget and their year-end bonuses.

So what's good sysadmin to do? Well that's where benchmarking comes in. You could cobble together your own tools and that's all fine and dandy, but probably far more of an investment then management is willing to spend. The remainder of this article we'll discuss the basic use of a couple of tools. The first and the easiest is the command line based Apache Bench utility known as *ab*.

If you have an Apache installation then you already have *ab*. What I like about *ab* is that you can test any http(s) server with this tool. It is extremely lightweight and very flexible.

The second is a tool called siege and it is available in the ports collection. Once again this is a command line friendly utility that is rather lightweight and offers a lot of the same features as *ab* but in a slightly different form.

The last tool I will talk about is *jmeter* which is another utility from the Apache Group. If you are afraid of the command line, then you've probably already stopped reading BSD Magazine. In case you've hung in there and have Java installed on your machine, by all means give *jmeter* a go. Personally I do not like *jmeter*, finding it both awkward to use and difficult to get good results from.

Let's take a quick look at *ab* and how you might use it to test your server. The first thing I learned is that the URL you give *ab* must have at least one leaf. For instance if I wanted to test my company's website I have to pick a specific page as *ab* will not test the domain root. Therefore I need to give it a complete URL like *http://www.olivent.com/about*. In addition I have found that at a minimum you need to set the number of requests and the concurrency in order to gather any useful information.

An example of the Apache Bench command:

```
ab -n 2 -c 2 http://
www.olivent.com/about
```

An example of the Siege command:

```
siege -r 2 -c 2 http://
www.olivent.com/about
```

While the output of these commands is interesting it is not necessary for the purposes of this article. What is important is that you understand that before you can test the web server and your application, you need to define the battery of tests you wish to run.

Properly implemented, these tools can assist you with determining if the problem is truly the fault of your hardware, the application code, or even an outside source such as network latency. As you can imagine, benchmark testing is a time-consuming process and you need to be able to approach management with all of your punctuation in the right place.

At a minimum I recommend that you perform several baseline tests on a static page host on the server in question. You must have a local server test to eliminate any inconsistencies relating to your LAN environment. Obviously you should have a corresponding result from another machine (or several) on the LAN to determine if your application is experiencing any latency from a poor network topology.

Next you will want to run the similar tests against various phases of your application. Perhaps a page that generates the same data as the static HTML page previously tested. If there is a noticeable discrepancy between the two then you have isolated your issue to the application's processing engine. It could be that the programming language you are working in performs poorly.

If the difference between the static HTML page and the dynamic page is not too large, then have the language pull that text from your database before rendering. Keep in mind that if you have a dynamic database-driven application, and the database is on another server, you will have to rely on the connectivity between the two servers.

In any event you have enough information at this point to approach management and get the programmers to tweak the performance of their work. If that is not possible then you might consider installing a language optimizer. I worked on one project written in PHP that just ran slowly.

Eventually we redeployed it under the Zend Server community edition on the exact same hardware and received a huge boost in performance. Obviously the application was written using the Zend Framework, thus making the port rather trivial.

On another project I worked with some Ruby developers and, sad to say, even purchasing *better* hardware didn't seem to improve performance. My firm feeling is that Ruby was not to blame but these particular developers were in over their heads. Eventually the important parts of the application were rewritten in PHP and integrated into an existing system. Understand the latter choice was a result of the existing application and not any language preference of PHP vs. Ruby.

Overall the use of these sorts of benchmarking tools can help an application project immensely. For a web programmer these sorts of tools could help them optimize their code as the write it. These tools definitely help the sysadmin ensure that the hardware is running at optimal levels.

One thing worth considering is the installation method of you web server. Did you install a pre-compiled package built with someone else's choices in an unknown environment? Did you compile the server yourself optimizing it for your hardware? Many believe that a production application should be compiled from source on the actual hardware for maximum efficiency.

Ultimately the choice is yours, but for the record I like to compile. Hopefully this 30,000 meter view of web server benchmarking was enough to give you pause to consider such things while you are building you next great web application. I would like to encourage you to give these utilities a try. If there is interest in discussing your findings, perhaps we can start a thread on the BSD News Network's site at *http://BSDNews.net.*

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 and tricks

OpenSSH: common but underappreciated

by Machtelt Garrels

What do you know about OpenSSH, the tool you use every day? Can you configure it on a per-host basis? Do you know how tunnels can help you in a tight corporate environment? Do you know how it can help you to securely browse in an unsecure environment? Does remote portforwarding ring a bell? Do you know how to build your own subsystems? If you answered *no* to at least one of these questions, you might learn a thing or two from this article.

What do you need?

We used in our examples OpenSSH as shipped with the operating system. Any version of OpenSSH will do the trick, as long as it is more recent than December 2005. Other implementations of SSH might support the features described below to a lesser extend, e.g. Putty for Windows and Mac, or WinSCP. The command line version on UNIX or alikes is always the most flexible. Apart from *BSD, OpenSSH is available for a wide variety of platforms, including all Linux distributions, most UNIX flavours, Mac OS X and Cygwin, see *http://www.openssh.com.* OpenSSH was originally developed by the OpenBSD team. There is a second team focussing on portability to other platforms.

SSH Configuration

The system-wide configuration of the SSH clients (`ssh`, `scp`, `sftp`) is done through `/etc/ssh/ssh_config`. All info can be found in man ssh_config. Use this command to get a comprehensible overview of how your setup differs from the default:

```
grep -v ^# /etc/ssh/ssh_config
```

On a per-user basis, the configuration is done in `~/.ssh/config`.

The file is just called config, and not `ssh_config` like the system-wide file.

Example:

```
Host dev.example.org
        User petra
        NoHostAuthenticationForLocalhost yes
Host web.example.org
        SendEnv JAVA_HOME
Host *
        StrictHostKeyChecking yes
```

Here you see how different hosts can take different options. Specifying the username, for instance, prevents you from having to type `ssh -l user hostname`. Other options would normally be defined on the command line like `ssh -o option=value hostname`. In our example, we also use:

- `NoHostAuthenticationForLocalhost`: used in an environment with shared home directories so that you don't get warnings about changed host keys.
- `SendEnv`: Specify which environment variables of the local host environment will be exported to the remote host environment.
- `StrictHostKeyChecking`: do not add host keys automatically to `~/.ssh/known_hosts`; SSH refuses to connect to hosts of which the host key has changed.

When starting an SSH session, the remote host name is looked up in the table. The options found in the first match are applied. If the host name is not found in the configuration file, default options (*) are applied.

More options can be found in the man page.

SSH Subsystems

A subsystem specifies a command to be executed by the SSH daemon on the remote host. The most common subsystem is `sftp`. Subsystems are defined in `/etc/ssh/sshd_config` on the remote host:

```
Subsystems sftp      /usr/lib/openssh/sftp-server
```

In order to make your own subsystem, perform the following steps:

- Create a script that holds the command(s) to be executed, for instance:

```
echo "echo Hello World" > /usr/lib/openssh/subsystest
```

- Make the script executable by chmodding it to 555.
- Edit `/etc/ssh/sshd_config`:

```
Subsystem helloworld      /usr/lib/openssh/subsystest
```

- Test:

```
ssh -s remote_host helloworld
```

SSH Agent Forwarding

One of the interesting features of the SSH agent is forwarding or keychaining. This feature allows users to take their identity with them from one host to another, as demonstrated in the Figure 1.

In the apicture below, this is what would happen without using the agent:

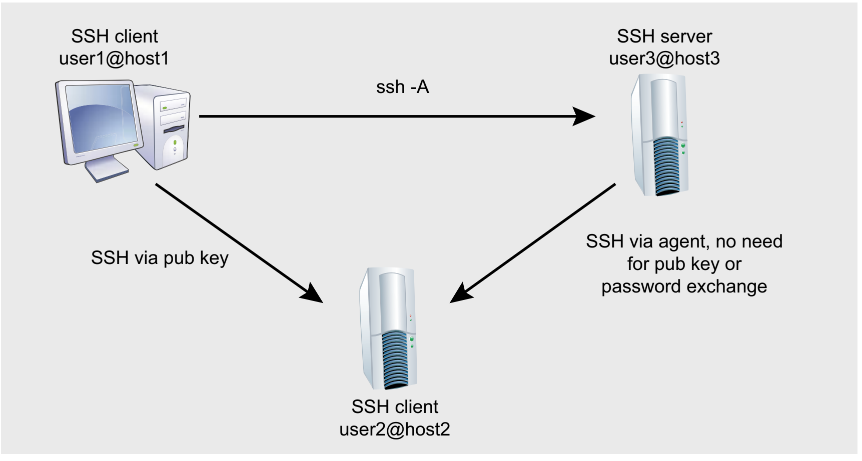


Figure 1. ssh agent

- ssh as user1 on host1 to user2 on host2: works using public key authentication, user2 on host2 has the public key info of user1 on host1 in his/her file, so user1 does not have to provide a password.
- ssh as user1 on host1 to user3: only works using password authentication.
- ssh as user3 on host 3 to user2 on host2: only works using password authentication.

Using the SSH agent, however, allows user3 to connect from host3 to user2 on host2 as well, without providing a password, eventhough this would normally not be allowed because user2 on host2 has not added user3's key to `~/.ssh/authorized_keys`. For host2, it would appear as if user3 on host3 is actually user1 on host1, the public key of the user is found in `authorized_keys` and authentication is done using keys instead of passwords.

Usage of the agent:

- Start the agent: `ssh-agent` – (this could also be done at login time, so all user processes know about it and can use the feature).
- Add your key to the agent: `ssh-add`
- Use the agent: `ssh -A user@host`
- Configure the agent in `~/.ssh/config`

```
Host trusted.example.org
   ForwardAgent yes
Host *
   ForwardAgent no
```

Read more in the man pages for `ssh-agent` and `ssh-add`.

SSH Tunnels

Local Port Forwarding

Local port forwarding is used for instance when the SSH port is open in a firewall, but the port you actually need is blocked, as shown in the image where we need to connect to a MySQL database. Moreover, in the case of databases you usually secure them and only allow connections from localhost. With SSH, we can fake this behaviour (see Figure 2). The general syntax for a local port forwarding is as follows:

```
ssh -Llocal_port:localhost:remote_port
user@host
```

The practical example:

```
ssh -L12345:127.0.0.1:3306
mysqladmin@server
```

Usage of the tunnel:

```
mysql -u admin -p -h 127.0.0.1 -P
12345
```

Remote Port Forwarding

Say that your home machine is in a local, non-routable home network, unaccessible from the outside. Yet when you are at work, you would like to have access to your home machine. This is a case for using remote port forwarding. You will need a server which you can reach both from work and from home through normal SSH. Setting up the tunnel works as follows:

- On your home machine, make a connection to a public machine, using the remote port forwarding syntax. Generally speaking, this would be

```
 ssh user@server -R local_port:
your_local_ip:remote_port
```

- In practice:

```
 ssh user@pubserver -R 54321:
192.168.10.2:22
```

Note that the local_port is local on the remote server, not on the local host, as demonstrated in the next step:

- Go to work, there connect to the public server and use the tunnel:

```
 ssh -p 54321 localhost
```

This will connect you to the normally unreachable home computer.

Secure Browsing

How often have you found yourself in an untrusted environment, like when using the wireless network at a conference? How do you know if your moves are monitored and logged? Can you surf the web in all safety? Using an SSH feature called the built-in SOCKS proxy, you don't need to worry. The SOCKS proxy is a special type of tunnel to a secure server. This server will make the HTTP requests for you, and securely send all information through the encrypted SSH channel. Start the tunnel as follows:

```
ssh -D proxy_port server
```

Now configure your browser to use SOCKS proxy localhost on this proxy_port; do not configure any HTTP proxy.

Just do it

So you've read through this article, now grab your keyboard and get started! And remember, there is lots of information in the man pages, including examples and other features that we couldn't include here. Keep in mind that there is more than just `man ssh`; try

```
apropos ssh
```

to see which man pages are installed on your system and just read them all!

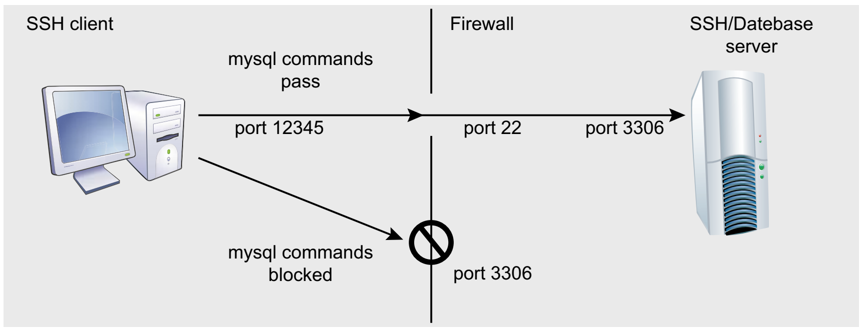


Figure 2. ssh tunnels

Interview with Olivier Cochard-Labbé, Founder of FreeNAS

Jesse Smith

The FreeNAS project (http://freenas.org/freenas), founded by Olivier Cochard-Labbé in 2005, is an open source network attached storage distribution. The project offers a simple, elegant way for home users and network administrators to host data on a small, stable platform at very low cost. Back in December there was talk of the FreeNAS project moving away from its FreeBSD roots and using Debian as the base for future releases. A short time later, iXsystems offered to the take the FreeNAS project under the company's wing and continue development using the FreeBSD platform. M. Cochard-Labbé was kind enough to take a few minutes from his busy schedule to talk about the project.

BSD Mag: Monsieur Cochard-Labbé, thank you for taking the time to answer some questions. To start, would you please tell us a little about yourself? Where you are from and how you became interested in open source?

OCL: I'm living in France, 33 years old, married with two daughters and I work as a network consultant at Orange Business Services. I discovered Slackware Linux during my studies in 1996, and have been a Linux desktop user since.

BSD Mag: There are not a lot of open source projects providing specialized NAS services. What prompted you to create FreeNAS?

OCL: In mid 2005, I wanted to transform one of my old PCs into a NAS server for home. My goals were simple:

- Boot from my USB key (the OS should be small)
- Use a software RAID-5 with 4 PATA hard drives

I didn't find an open source project that filled my needs, so I chose to build my own. My second motivation was that I was a simple computer user, and wanted to use this *exercise* to explore the operating system more deeply. I already had a m0n0wall system at home, and I wanted to have the same interface for my NAS. This is the historical reason why FreeNAS is based on FreeBSD. I learned PHP and discovered FreeBSD by studying the m0n0wall code. And after some days, the first release of FreeNAS was available.

I never imagined that my little *customized m0n0wall to NAS* would become so famous... And this created some problems: Because I'm not a developer, I had to learn how to use subversion, how to write PHP that supported translation (getext), and I was afraid of unknown potential bugs. FreeNAS was only a hobby, and couldn't be prioritised over my family life and paid job. Managing this project eats into lots of my sleep time.

BSD Mag: For your latest release, there were over 10,000 downloads for the project's live CD. From the feedback you've received, would you say most of your users are home users or businesses?

OCL: I believe lots of users are home users. The *business* features of FreeNAS are presently too limited (user authentication, user/group fine permission and quota management). But, because I never used a professional NAS (like netapp), it's not easy for me to find out what type of features are needed for business.

And it's not easy to work alone, at home, on the professional features. For example, for adding the MS Active Directory authentication, I needed to build a MS Windows server VM, and understand how Windows servers work before testing it with FreeNAS. This is why I never had time to add LDAP integration and other more complex features.

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

BSD Mag: A little while ago, you announced that FreeNAS would move away from FreeBSD to a Debian base. Would you mind explaining your reasons for the change?

OCL: It's a little more complex story:

I created FreeNAS during October 2005, and was the only developer until July 2006 when Volker Theile joined the project as a developer. But my professional and personal status changed a lot in 2007 and prevented me from continuing to spend my free time on FreeNAS. Then I chose to give the project keys to Volker in April 2008 to prevent slowing down the development of FreeNAS: I didn't contribute to the code after that. Volker has worked, almost alone, on FreeNAS since. And in September 2009, we had an internal discussion about the future of FreeNAS regarding its present technical limitations (the biggest is the difficulty of adding user plug-ins). The conclusion was FreeNAS needed a full rewrite. Volker (still project leader and the only developer) preferred to use a Linux base because he found it easier to develop under Linux. I approved this choice. But soon after the public announcement of this big planned change, I received a lot of email from the FreeBSD world. And the most important one from iXsystems that proposed to take FreeNAS under their wings. This is why I came back actively to the project, to manage the transition period with iXsystems.

BSD Mag: Regarding the arrangement with iXsystems, how will FreeNAS benefit from their involvement and how does iXsystems benefit? Will we still see a Debian version of FreeNAS in the future?

OCL: They will give FreeNAS a more business-use direction (still keeping the home user features). Regarding the *Debian version of FreeNAS*, as a full new project, we can't call it the *Debian version of FreeNAS*. This project was created by Volker and is called OpenMediaVault, and you can follow the rapid development here: *http://blog.openmediavault.org*.

BSD Mag: FreeNAS recently reached version 0.7. What new features or improvements are we going to see between now and version 1.0?

OCL: You need to ask this question to iXsystems now. *

BSD Mag: What are some advantages to choosing FreeNAS over other, proprietary, NAS solutions?

OCL: I've never used proprietary NAS solutions, so I can't really answer this question. Ask this question to the FreeNAS users.

BSD Mag: Lots of people in the open source community like to tinker. Are there areas where the FreeNAS project could use some help (developing, testing, writing documentation)?

OCL: The documentation needs a big update. And we always need developers. It's very easy to contribute on FreeNAS: It's only some PHP pages that create text files!

BSD Mag: NAS systems tend to get very large. How do you test the storage/speed limits of FreeNAS?

OCL: I can't test the storage and speed limits, because these tests need some hardware that I don't have.

BSD Mag: The ZFS file system seems designed for servers and NAS systems. What are some things you like about ZFS and is there anything you don't like about it?

OCL: I'm not a system engineer. I didn't really have the skills for comparing ZFS to another file system. What I've read on the Feature Requests list to add to ZFS, I've never heard about before. ZFS needs a big system that is not very compatible with a little home NAS.

BSD Mag: Are you working on any other projects at the moment? What sort of things are on your plate?

OCL: Learning from my FreeNAS experience and as a network guy, I've started a new project: BSD Router Project (*http://bsdrp.net*). It's an embedded open source router distribution (based on nanoBSD) configurable from the CLI only. But, with my coming back to the FreeNAS project, I can't work as I would like to on this project.

BSD Mag: Is there anything else you'd like to share with our readers?

OCL: Before starting FreeNAS, I'd never touched a FreeBSD system and didn't know PHP/Shell programming. It's by creating FreeNAS that I've learned these technologies. This means that anyone who has a good idea, without programming skills, can very easily create an *appliance* OS such as FreeNAS.

Thanks again to M. Cochard-Labbé for his time and inspiring words.

* As Cochard-Labbé said, the future of the FreeNAS project is now in the hands of iXsystems. Back in December, Josh Paetzel gave an interview to BSD Talk in which he chatted a little about the project and its future. During the interview he mentioned that not only does iXsystems use FreeNAS internally for storage, but they also sell and support it as a product. One of the reasons iXsystems opted to take over the FreeNAS project is in order to maintain a steady support path for their clients. Keeping the project on a FreeBSD base will additionally insure their customers can continue to use ZFS for their storage needs. Mr. Paetzel went on to say that iXsystems has three general objectives for FreeNAS:

- Move FreeNAS to the new FreeBSD 8.0 platform.
- Improve modularity and add-in packages.
- Maintain smooth support for existing FreeNAS installs.

For those interested, you can listen to the entire enlightening Paetzel interview on the BSD Talk website at *http://bsdtalk.blogspot.com/2009/12/bsdtalk-182-freenas-with-josh-paetzel.html*.

About the author

Jesse Smith is a system administrator and programmer by training, an open source advocate by choice and a writer at heart. When he's not working with computers, he loves spending time with his family and enjoying the natural beauty of his native Canada.

SAINT®

Integrated Vulnerability Assessment and Penetration Testing

Examine, expose, and exploit your vulnerabilities before an attacker does

Examine your network with the SAINT® vulnerability scanner, and expose the areas where an attacker could breach your network. Then, take the next step and exploit the vulnerability. This allows you to focus on the high-severity vulnerabilities and provides a starting point for prioritizing remediation efforts.

What SAINT can do for you –

- ✔ Lets you exploit vulnerabilities found by the vulnerability scanner
- ✔ Show you if your network is compliant with PCI security standards. SAINT is a PCI DSS approved scanning vendor.
- ✔ Show you where to begin remediation—with the exploitable vulnerabilities
- ✔ Let you scan and exploit both IPv4 and IPv6 addresses
- ✔ Give you cross references to CVE, CVSS, IAVA, OSVDB, and SANS/FBI Top 20
- ✔ Allows you to generate colorful and easy-to-read reports

Get free white papers and more information at www.saintcorporation.com

Contact SAINT's sales team at 1-800-596-2006 x0119 or sales@saintcorporation.com

Extreme Network Security

Orion II iX-N4236

Powerful *4U* Orion II Storage Series

- ✓ Outstanding performance
- ✓ Excellent cooling efficiency
- ✓ Up to 72 TB in 4U, unparalleled storage density

To order today call: **1-800-820-BSDi**

Notable features include:

- Dual Intel® 64-Bit Socket 1366 Quad-Core or Dual-Core, Intel® Xeon® Processor 5500 Series
- 4U Storage Server Chassis with up to 72 TB storage capacity
- 36 x 3.5 Hot-Swap SAS/SATA HDDs (24 front side + 12 rear side)
- 1400 W (1+1) Redundant High Efficiency Power Supply (Gold level 93%+ power efficiency)
- Dual Intel® 5520 chipsets with QuickPath Interconnect (QPI) up to 6.4 GT/s
- Up to 144GB DDR3 1333/1066/800 MHz ECC Registered DIMM/24 GB Unbuffered DIMM
- 2 (x16) PCI-E 2.0, 4 (x8) PCI-E 2.0 (1 in x16 slot), 1 (x4) PCI-E (in x8 slot)
- Intel® 82576 Dual-port Gigabit Ethernet Controller

iXsystems Introduces the Orion II 4U Storage Solution

The iX-N4236 boasts energy efficient technology and maximum, high density storage capacity, creating a 4U powerhouse with superior cooling.

The Orion II has **thirty-six hot-swappable SAS/SATA drive bays**, providing 50% more storage density than its predecessor. By delivering high-end storage density within a single machine, iXsystems cuts operating costs and reduces energy requirements.

Storage sizes for the iX-N4236 are customizable, with 250GB, 500GB, 750GB, 1TB, and 2TB hard drives available. For environments requiring maximum storage capacity and efficiency, 2TB Enterprise-class drives are available from Western Digital®, Seagate®, and Hitachi. These drives feature technologies to prevent vibration damage and increase power savings, making them an excellent choice for storage-heavy deployment schemes.

Powerful Intel® Xeon® 5500 Series Processors have a light footprint, while creating a perfect environment for intense virtualization, video streaming, and management of storage-hungry applications. Energy efficient DDR3 RAM complements the other power saving components while still providing 18 slots and up to144GB of memory overall.

100% cooling redundancy, efficient airflow, and intelligent chassis design ensure that even under the heaviest of workloads, the Orion II remains at an optimal temperature, while still drawing less power than other servers in its class. With a 1400 W Gold Level (93%+ efficient) power supply, the entire system works together to efficiently manage power draw and heat loss.

For more information or to request a quote, visit:
http://www.iXsystems.com/Orion2

iX systems™

intel Xeon inside™

Powerful. Intelligent.