BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

BSD FIREWALLS

EXCLUSIVELY

▶ INTRODUCTION TO NANO BSD

INSIDE

EASIER WINE INSTALLATION ON AMD64 FREEBSD
CONFIGURING IP-BASED SSL
BSD FILE SHARING - PART 4. SSH
BSD OPINION. ROB SOMMERVILLE
REDUNDANT FIREWALLS WITH OPENBSD, CARP AND PFSYNC

VOL.3 NO. 6
ISSUE 6/2010(12)
1ISSN 898-9144

iX systems

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

✓ Increased Performance ✓ Impressive Energy Savings

(PC-BSD Version 8.0 Hubble Edition Screenshot)

Professional Enterprise and Desktop Support for Your Peace of Mind

From optimizing your small office set-up to guidance on large-scale deployments, the iXsystems team can ensure you get the most from your PC-BSD and FreeBSD systems

iXsystems understands the need for both casual and professional users to have peace of mind with their information technology operations. Our professional services staff aims to provide just that. Whether you are running a single desktop or have an HPC, datacenter, or server farm, the experienced technicians at iXsystems can assist you with all aspects of the FreeBSD and PC-BSD operating systems.

Industry Leading Technical Expertise - The iXsystems Professional Services team is comprised of long-time FreeBSD developers, administrators, and project committers. From development to consultation to emergency problem solving, our team will put their many years of experience to work for you. With support staff stationed in North America, Europe and Asia, you can rest assured your operations are in good hands round the clock and around the world.

Large Rollouts - The Professional Services Team provides installation support for large networks. Let our technicians put their expertise to work for you to determine your operational needs and provide the critical support your system administrators need for rollouts and system migrations.

Escalation Management - When the iXsystems Professional Services Team encounters a confirmed bug, we can escalate the bug to the FreeBSD engineering team. We can also work with The FreeBSD Project to create and submit patches to the FreeBSD community for possible inclusion in the latest release.

Custom Development and Consultation Services - iXsystems employs and partners with some of the most brilliant minds in the FreeBSD Community to offer custom development and advanced level FreeBSD consulting solutions. Our Account Management Service Professionals will work with you to coordinate and develop software solutions specific to your business operations. iXsystems offers kernel tuning and system optimization, device driver creation, kernel, userland, and embedded systems development, and a host of other services that allow your company to fully utilize the FreeBSD and PC-BSD platforms.

For more information contact iXsystems at +1-800-820-BSDi or visit our website at http://www.iXsystems.com/bsdsupport and fill out the inquiry form. We will pair you up with an Account Management Service Professional that can assess your needs and create a custom FreeBSD support plan for your organization!

Installation and Configuration

PC-BSD Desktop Support means you will have all the assistance you need getting your system up, running, and configured for optimal performance.

Troubleshooting Assistance

In the event something does not work properly for you, our expert technicians will walk you through the troubleshooting process to determine the cause of your problem and provide you with a solution.

Knowledge Database

With your Desktop Support package, you will gain access to our Knowledge Database. The Knowledge Database contains problems and solutions associated with previous support inquiries, saving valuable time when issues arise.

PBI Creation and Application Installation

Our Professional Services Team can create custom PBIs (push button installers) for your PC-BSD system. These self-contained applications eliminate the problem of shared dependencies and provide the user with point-and-click program installation and removal.

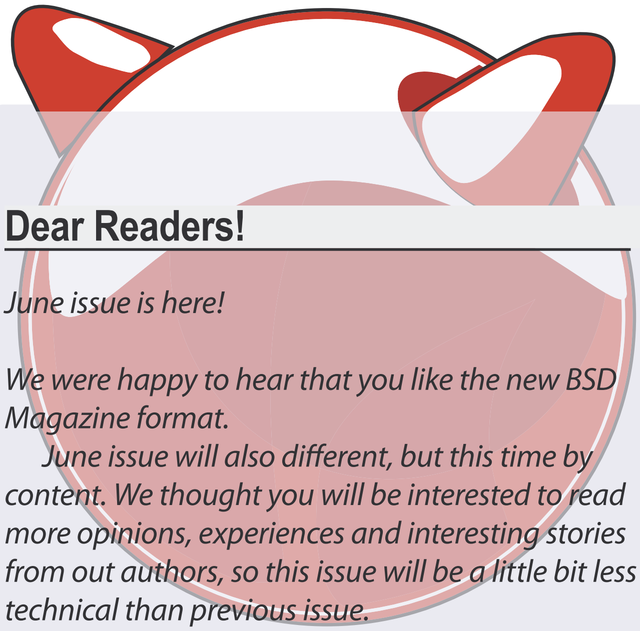
Accelerated Support Solutions

From desktop support to Professional Enterprise Service Level packages, the iXsystems Professional Services Team can support all of your FreeBSD and PC-BSD related needs.

Desktop support packages include 8x5 support with same day/next day prioritized responses. Professional Enterprise Service Level packages are available in both 8x5 and 24x7 models. Pricing is determined by hourly blocks or on a per unit basis.

- ✓ **Increased Uptime**
- ✓ **Improved Productivity**
- ✓ **Better Security**

iXsystems™



Dear Readers!

June issue is here!

We were happy to hear that you like the new BSD Magazine format.

June issue will also different, but this time by content. We thought you will be interested to read more opinions, experiences and interesting stories from out authors, so this issue will be a little bit less technical than previous issue.

We look forward to hear your opinion.

Did you like BSD in this way? Do you agree with our authors? Was it helpful, interesting?

Surely you will find some How-to's inside as always and this time we will present another BSD system – NanoBSD.

The number of our readers has doubled since BSD Magazine became free on-line! 18 000 subscribers are getting direct links with our magazine straight to their e-mail box! If you are still not subcribed, do this and you will always know when the newest issue is published.

Thank you for your feedback and enjoy the magazine that was created with your help!

Olga Kartseva
Editor in Chief
olga.kartseva@software.com.pl

MAGAZINE BSD

Editor in Chief:
Olga Kartseva
olga.kartseva@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:
Olga Kartseva
olga.kartseva@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*

The editors use automatic DTP system AURUS

Mathematical formulas created by Design Science MathType™.

Feature: WebHostingBuzz,

a hosting company all too familiar with BSD

You may have noticed WebHostingBuzz has started to advertise in BSD magazine in recent months. In this article, we catch up with Matthew Russell, CEO and Dennis Arkhangelski, Senior Technical Manager and ask them some questions about their use of BSD within the organisation.

What can you tell us about BSD within WebHostingBuzz?

MR: We've used FreeBSD for many years as the operating system of choice for our shared servers. The stability and performance is the major reason we elected to use this operating system, and it delivers us excellent results.

DA: As well as using FreeBSD for our shared infrastructure, myself and many of our team have a vested interest and have used FreeBSD and OpenBSD in various guises. Some of our workstations within the office are FreeBSD or OpenBSD and we run several internal and external FreeNAS (also BSD) based devices.

How do you intend to use BSD in the future?

DA: We will continue to use FreeBSD based shared servers. We also put a lot of emphasis on research and development and have been thoroughly testing ZFS and migrations to ZFS from UFS2 based systems. We'll probably see more of that in the future.

We've also been looking at OpenBSD/Openbgpd based software routers for clients that require specific routing setups, but have not wanted to spend the large amounts of cash required for Cisco 65xx based setups. This is something else going through our labs right now. We're also setting up a system to contribute back to the FreeBSD source.

Do you use other operating systems?

MR: Actually, yes we do. Depending on the nature of the application or server, we run some physical and some virtual CentOS nodes, as well as the occasional Windows Server 2008 node. We're flexible in terms of the OS we use based on the requirements of the project.

DA: As Matt says, we run a number of different OSes. However, I should point out that BSD based OSes do make up the majority of our hosting nodes.

I'm a BSD professional or developer and need web hosting. What would you recommend?

MR: This will depend on the type of hosting you need and the type of site or application you intend to host. Hopefully you will choose us! If you do, at the cheaper end of the scale our Virtual Private Server plans provide full root access and a number of operating system choices. Our Dedicated Servers are the next step up and we can install pretty much any BSD flavour on these. For the even larger project, we can setup a cluster of servers, perhaps separating the storage, database and web front end.

We're a cPanel partner and fully recommend cPanel (*www.cpanel.net*) if you're looking for a friendly GUI to manage the websites / email addresses on your Virtual Private Server or Dedicated Server. cPanel is fully compatible with FreeBSD.

DA: Don't forget, our experience with BSD means that our management services mean just that. We'll fully manage your server if you wish us to, and we can communicate with you on all levels, no matter how technical (or non-technical) your request may be.

About WebHostingBuzz

WebHostingBuzz is a staunch supporter of BSD Magazine and we're pleased to have them as one of our advertisers. They boast over 35,000 clients with close to 250,000 websites within their network. We will be asking them frequent input and news from their use of BSD within the hosting industry, and you can expect to see more features from them in future.
Visit *www.webhostingbuzz.com*

Looking for help, tip or advice?
Want to share your knowledge with others?

Visit BSD magazine forum

Give us your opinion about the magazine's content
and help us create the most useful source for you!

www.bsdmag.org

Introduction

to NanoBSD

NanoBSD is a tool developed by Poul-Henning Kamp phk@FreeBSD.org. It creates a FreeBSD system image for embedded applications, suitable for use on a Compact Flash card (or other mass storage medium).

What you will learn…

- how to build and deploy a minimalistic customized FreeBSD image suitable for an embedded systems or specialized computer appliances

What you should know…

- basic knowledge of system administration
- basic knowledge of scripting in /bin/sh
- knowledge of how the FreeBSD build system works
- knowledge of how to install FreeBSD packages

It can be used to build specialized install images, designed for easy installation and maintenance of systems commonly called *computer appliances*. Computer appliances have their hardware and software bundled in the product, which means all applications are pre-installed. The appliance is plugged into an existing network and can begin working (almost) immediately. The features of *NanoBSD* include:

- Ports and packages work as in FreeBSD – Every single application can be installed and used in a *NanoBSD* image, the same way as in FreeBSD.
- No missing functionality – If it is possible to do something with FreeBSD, it is possible to do the same thing with *NanoBSD*, unless the specific feature or features were explicitly removed from the *NanoBSD* image when it was created.
- Everything is read-only at run-time – It is safe to pull the power-plug. There is no necessity to run `fsck(8)` (*http://www.freebsd.org/cgi/man.cgi?query=fsck&sektion=8*) after a non-graceful shutdown of the system.
- Easy to build and customize – Making use of just one shell script and one configuration file it is possible to build reduced and customized images satisfying any arbitrary set of requirements.

The design of NanoBSD

Once the image is present on the medium, it is possible to boot *NanoBSD*. The mass storage medium is divided into three parts by default:

- Two image partitions: `code#1` and `code#2`.
- The configuration file partition, which can be mounted under the `/cfg` directory at run time.

These partitions are normally mounted read-only.

The `/etc` and `/var` directories are `md(4)` (*http://www.freebsd.org/cgi/man.cgi?query=md&sektion=4*) (malloc) disks.

The configuration file partition persists under the `/cfg` directory. It contains files for `/etc` directory and is briefly mounted read-only right after the system boot, therefore it is required to copy modified files from `/etc` back to the `/cfg` directory if changes are expected to persist after the system restarts.

Example 1. Making persistent changes to `/etc/resolv.conf`

```
# vi /etc/resolv.conf [...]
# mount /cfg
# cp /etc/resolv.conf /cfg
# umount /cfg
```

Note: The partition containing /cfg should be mounted only at boot time and while overriding the configuration files.

Keeping /cfg mounted at all times is not a good idea, especially if the *NanoBSD* system runs off a mass storage medium that may be adversely affected by a large number of writes to the partition (i.e. when the filesystem syncer flushes data to the system disks).

A *NanoBSD* image is built using a simple nanobsd.sh shell script, which can be found in the /usr/src/tools/tools/nanobsd directory. This script creates an image, which can be copied on the storage medium using the dd(1) (*http://www.freebsd.org/cgi/man.cgi?query=dd&sektion=1*) utility.

The necessary commands to build a *NanoBSD* image are:

```
1# cd /usr/src/tools/tools/nanobsd
2# sh nanobsd.sh
3# cd /usr/obj/nanobsd.full
4# dd if=_.disk.full of=/dev/da0 bs=64k
```

1 Change the current directory to the base directory of the *NanoBSD* build script (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/nanobsd/article.html#NBSD-CD*).
2 Start the build process (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/nanobsd/article.html#NBSD-SH*).
3 Change the current directory to the place where the built images are located (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/nanobsd/article.html#NBSD-CD2*).
4 Install *NanoBSD* onto the storage medium (*http://www.freebsd.org/doc/en_US.ISO8859-1/articles/nanobsd/article.html#NBSD-DD*).

This is probably the most important and most interesting feature of *NanoBSD*. This is also where you will be spending most of the time when developing with *NanoBSD*. Invocation of the following command will force the nanobsd.sh to read its configuration from the myconf.nano file located in the current directory:

```
# sh nanobsd.sh -c myconf.nano
```

Customization is done in two ways:

- Configuration options
- Custom functions

With configuration settings, it is possible to configure options passed to both the buildworld and installworld stages of the *NanoBSD* build process, as well as internal options passed to the main build process of *NanoBSD*. Through these options it is possible to cut the system down, so it will fit on as little as 64MB. You can use the configuration options to trim down FreeBSD even more, until it will consists of just the kernel and two or three files in the userland.

The configuration file consists of configuration options, which override the default values. The most important directives are:

- NANO_NAME – Name of build (used to construct the workdir names).
- NANO_SRC – Path to the source tree used to build the image.
- NANO_KERNEL – Name of kernel configuration file used to build kernel.
- CONF_BUILD – Options passed to the buildworld stage of the build.
- CONF_INSTALL – Options passed to the installworld stage of the build.
- CONF_WORLD – Options passed to both the buildworld and the installworld stage of the build.
- FlashDevice – Defines what type of media to use. Check the FlashDevice.sub file for more details.

It is possible to fine-tune *NanoBSD* using shell functions in the configuration file. The following example illustrates the basic model of custom functions:

```
cust_foo () (
     echo "bar=topless" > \
         ${NANO_WORLDDIR}/etc/foo
)
customize_cmd cust_foo
```

A more useful example of a customization function is the following, which changes the default size of the /etc directory from 5MB to 30MB:

Listing 1. *Installation*

```
install_packages () (
   mkdir -p ${NANO_WORLDDIR}/packages
   cp /usr/src/tools/tools/nanobsd/packages/* \
      ${NANO_WORLDDIR}/packages
   chroot ${NANO_WORLDDIR}
   sh -c 'cd packages; pkg_add -v *;cd ..;'
  rm -rf ${NANO_WORLDDIR}/packages
)
customize_cmd install_packages
```

```
cust_etc_size () (
     cd ${NANO_WORLDDIR}/conf
     echo 30000 > default/etc/md_size
)
customize_cmd cust_etc_size
```

There are a few default pre-defined customization functions ready for use:

- `cust _ comconsole` – Disables getty(8) (*http://www.freebsd.org/cgi/man.cgi?query=getty&sektion=8*) on the VGA devices (the `/dev/ttyv*` device nodes) and enables the use of the COM1 serial port as the system console.
- `cust _ allow _ ssh _ root` – Allow `root` to login via `sshd(8)` (*http://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8*).
- `cust _ install _ files` – Installs files from the `nanobsd/Files` directory, which contains some useful scripts for system administration.

Packages can be added to a *NanoBSD* image using a custom function. The following fuction will install all the packages located in `/usr/src/tools/tools/nanobsd/packages`: see Listing 1.

Configuration file example

A complete example of a configuration file for building a custom *NanoBSD* image can be: see Listing 2.

Updating NanoBSD

The update process of *NanoBSD* is relatively simple:

- Build a new *NanoBSD* image, as usual.
- Upload the new image into an unused partition of a running *NanoBSD* appliance.

The most important difference of this step from the initial *NanoBSD* installation is that now instead of using the `_ .disk.full` file (which contains an image of the entire disk), the `_ .disk.image` image is installed (which contains an image of a single system partition).

- Reboot, and start the system from the newly installed partition.
- If all goes well, the upgrade is finished.

Listing 2. *Configuration file for building a custom NanoBSD image*

```
NANO_NAME=custom
NANO_SRC=/usr/src
NANO_KERNEL=MYKERNEL
NANO_IMAGES=2
CONF_BUILD='
NO_KLDLOAD=YES
NO_NETGRAPH=YES
NO_PAM=YES
'
CONF_INSTALL='
NO_ACPI=YES
NO_BLUETOOTH=YES
NO_CVS=YES
NO_FORTRAN=YES
NO_HTML=YES
NO_LPR=YES
NO_MAN=YES
NO_SENDMAIL=YES
NO_SHAREDOCS=YES
NO_EXAMPLES=YES
NO_INSTALLLIB=YES
NO_CALENDAR=YES
NO_MISC=YES
NO_SHARE=YES
'
CONF_WORLD='
NO_BIND=YES
NO_MODULES=YES
NO_KERBEROS=YES
NO_GAMES=YES
NO_RESCUE=YES
NO_LOCALES=YES
NO_SYSCONS=YES
NO_INFO=YES
'
FlashDevice SanDisk 1G
cust_nobeastie() (
     touch ${NANO_WORLDDIR}/boot/loader.conf
     echo "beastie_disable=\"YES\"" >> \
   ${NANO_WORLDDIR}/boot/loader.conf
)

customize_cmd cust_comconsole
customize_cmd cust_install_files
customize_cmd cust_allow_ssh_root
customize_cmd cust_nobeastie
```

- If anything goes wrong, reboot back into the previous partition (which contains the old, working image), to restore system functionality as fast as possible. Fix any problems of the new build, and repeat the process.

To install new image onto the running *NanoBSD* system, it is possible to use either the `updatep1` or `updatep2` script located in the `/root` directory, depending from which partition is running the current system.

According to which services are available on host serving new *NanoBSD* image and what type of transfer is preferred, it is possible to examine one of these three ways:

Using ftp(1)

If the transfer speed is in first place, use this example:

```
# ftp myhost get _.disk.image "| sh updatep1"
```

Using ssh(1)

If a secure transfer is preferred, consider using this example:

```
# ssh myhost cat _.disk.image.gz | zcat | sh updatep1
```

Using nc(1)

Try this example if the remote host is not running neither `ftp(1)` (*http://www.freebsd.org/cgi/man.cgi?query=ftp&sektion=1*) or `sshd(8)` (*http://www.freebsd.org/cgi/man.cgi?query=sshd&sektion=8*) service:

- At first, open a TCP listener on host serving the image and make it send the image to client:

  ```
  myhost# nc -l 2222 < _.disk.image
  ```

  Note: Make sure that the used port is not blocked to receive incoming connections from *NanoBSD* host by firewall.
- Connect to the host serving new image and execute `updatep1` script:

  ```
  # nc myhost 2222 | sh updatep1
  ```

DANIEL GERZO

I am a FreeBSD user and enthusiast since around 2003. I received a documentation commit bit in 2006 which makes me a FreeBSD developer as well. I am living in central Europe, the capital of Slovakia – Bratislava. I currently study Computer science at the Slovak University of Technology. I also own a small company providing consultancy, administration, and other IT services for FreeBSD servers.

Secure Your Wireless

with IPsec

Wireless access is all the rage. Wireless this, wireless that. Hot spots are turning up everywhere. Many are free. Many have absolutely no security. There are several in my neighborhood. I have no idea who is running them, but at least one is wide open.

This article will show you one method for locking down your wireless network so that nobody but you can use it. This approach will take you beyond WEP and MAC address filtering, both of which are a good start but have known exploits. This article expands upon the IPsec foundation and demonstrates an easy method for securing your *Wireless Access Point* (WAP).

I will be using FreeBSD 4.10-Stable for this excursion. Please keep your hands and legs within the vehicle at all times. In case of emergency, please follow the directions of your crew. They know what to do.

While writing this article, I have assumed the following:

- You have wireless access on your laptop.
- You have a gateway already set up that will link your wireless and wired subnets.
- Your WAP is set up as follows:
  - You have the SSID set.
  - Traffic is flowing.
  - You changed the default admin password.

Why Bother with Wireless?

Wireless is practically wide open for anyone with a laptop, a wireless card, and the appropriate set of tools. WEP is defeatable. MAC addresses are sniffable and spoofable. In short, you need the next level: IPsec.

If wireless is so risky, why use it?

Convenience.

Wireless is convenient. There are no cables to run. Anyone can pop down to the local Future Shop (*http://www.futureshop.ca/*), buy a wireless access point (*http://www.linksys.com/products/product.asp?grid=33&scid=35&prid=505*), plug it in, turn it on, and start surfing the 802.11 information superhighway. Does this sound familiar? It should. I did it and I wrote about it (*http://beta.freebsddiary.org/wap.php*). I did the right thing. I filtered by MAC address. I turned on WEP. Then I turned off WEP when I had trouble. Yes, I was vulnerable, but no one compromised my system ... as far as I know.

Since writing that article nearly 18 months ago, I have moved to a new house. I've gone through the same

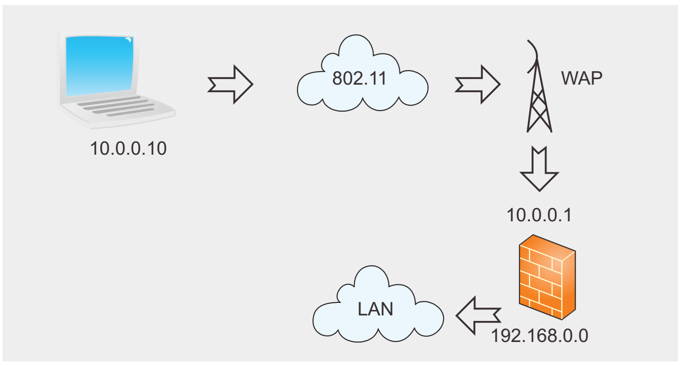


Figure 1. *Wireless network*

process I did last time, setting up a new rack and running some cables through the walls. This time I convinced myself that I would set up a secure wireless network. It took me a few hours, but I finally figured it out.

To answer the original question of this section and to tie it in with my recent move, I use wireless so I don't have to run cables. I want to use the laptop in the living room, the dining room, or on the front step. (I'm actually typing this into my Windows XP workstation sitting at my desk in the basement, SSH'd into the new Antec *http://beta.freebsddiary.org/antec.php* box.) As I work wirelessly, I want to keep people off my private network and keep prying eyes away from my communications. I can do that with IPsec.

Also in my mind is my neighbor, I don't know who, but somebody running a WAP nearby, totally unsecured. I know how easy it would be for me to use that internet connection. I don't want someone using mine.

What Is IPsec?

IPsec is short for IP security. It is a set of protocols for securely exchanging packets at the IP layer. VPNs (*http://beta.freebsddiary.org/pipsecd.php*) frequently use it. We can use the same approach to secure our wireless network.

IPsec uses shared secrets to encrypt data. It also uses security policies to decide what types of traffic to encrypt between which hosts.

FreeBSD-specific details

This section outlines some of the details that are specific to IPsec on FreeBSD. Regardless of the operating system you wish to use, you will need an IPsec-enabled kernel. On FreeBSD, add the following directives to your kernel configuration file, and then compile a new kernel (*http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/kernelconfig-building.html*):

```
options IPSEC         #IP security
options IPSEC_ESP     #IP security (crypto; define w/
                    IPSEC)
options IPSEC_DEBUG   #debug for IP security
```

I haven't actually used the features of `IPSEC_DEBUG`, but it's there if I need it.

Note: If you are running 5.x, use `FAST_IPSEC` instead of the directives mentioned above. Also remove `INET6`, as `FAST_IPSEC` does not support it.

Add the following directive to `/etc/rc.conf` to set up your IPsec database at boot time:

Listing 1. *Laptop rules*

```
add 10.0.0.1 10.0.0.10 esp 691 -E rijndael-cbc "1234567890123456";
add 10.0.0.10 10.0.0.1 esp 693 -E rijndael-cbc "1234567890123456";

spdadd 10.0.0.0/24 0.0.0.0/0 any -P out ipsec esp/tunnel/10.0.0.10-10.0.0.1/require;
spdadd 0.0.0.0/0 10.0.0.0/24 any -P in ipsec esp/tunnel/10.0.0.1-10.0.0.10/require;
```

Listing 2. *Gateway rules*

```
add 10.0.0.1 10.0.0.10 esp 691 -E rijndael-cbc "1234567890123456";
add 10.0.0.10 10.0.0.1 esp 693 -E rijndael-cbc "1234567890123456";

spdadd 10.0.0.0/24 0.0.0.0/0 any -P in ipsec esp/tunnel/10.0.0.10-10.0.0.1/require;
spdadd 0.0.0.0/0 10.0.0.0/24 any -P out ipsec esp/tunnel/10.0.0.1-10.0.0.10/require;
```

Listing 3. *Confirming traffic*

```
# tcpdump -ni dc0 not esp
tcpdump: listening on dc0
13:40:25.651640 0.0.0.0.68 > 255.255.255.255.67: xid:0xebd53d39 [|bootp] [tos 0x10]
13:40:25.656090 10.0.0.1.67 > 10.0.0.10.68: xid:0xebd53d39 Y:10.0.0.10 S:10.0.0.1
     [|bootp] [tos 0x10]
```

```
ipsec_enable="YES"
```

That directive will load your IPsec configuration directives from `/etc/ipsec.conf`. (You can configure the actual filename using `ipsec_file="/your/file/here"`. I will give you examples for that file later in this article.)

Walk First, Then Run

I'm a big believer in starting small and working one's way toward a goal. For my testing, I first tried IPsec over my wired network, then moved it over to the wireless network. You may find this strategy useful too. It allows you to concentrate on the IPsec portion of the problem, make it

Listing 4. *IPsec traffic*

```
# tcpdump -ni dc0
tcpdump: listening on dc0
13:44:34.371866 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0xfe)
13:44:34.385237 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xec)
     (frag 368:1480@0+)
13:44:34.385339 10.0.0.1 > 10.0.0.10: esp (frag 368:48@1480)
13:44:34.387672 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xed)
     (frag 369:1480@0+)
13:44:34.387775 10.0.0.1 > 10.0.0.10: esp (frag 369:48@1480)
13:44:34.390066 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xee)
     (frag 370:1480@0+)
13:44:34.390165 10.0.0.1 > 10.0.0.10: esp (frag 370:48@1480)
13:44:34.390996 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xef)
13:44:34.393155 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xf0)
     (frag 372:1480@0+)
13:44:34.393260 10.0.0.1 > 10.0.0.10: esp (frag 372:48@1480)
13:44:34.394641 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0xff)
13:44:34.396986 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0x100)
13:44:34.398044 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0xf1)
     (frag 373:1480@0+)
13:44:34.398142 10.0.0.1 > 10.0.0.10: esp (frag 373:48@1480)
```

Listing 5. *The ping*

```
13:45:39.886113 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0x118)
13:45:39.887436 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0x10a)
13:45:40.898972 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0x119)
13:45:40.900134 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0x10b)
13:45:41.908735 10.0.0.10 > 10.0.0.1: ESP(spi=0x000002b5,seq=0x11a)
13:45:41.909912 10.0.0.1 > 10.0.0.10: ESP(spi=0x000002b3,seq=0x10c)
```

Listing 6. *The example of traffic that does not use ipsec*

```
$ sudo tcpdump -ni fxp1
tcpdump: listening on fxp1
13:47:39.130953 192.168.0.21.22 > 192.168.0.99.3077: P 903783889:903783933(44)
     ack 4184487194 win 58400 (DF) [tos 0x10]
13:47:39.151794 192.168.0.99.1763 > 66.197.0.145.6665: . ack 305831024 win 64160 (DF)
13:47:39.252127 192.168.0.99.3077 > 192.168.0.21.22: . ack 44 win 64028 (DF)
13:47:39.621526 192.168.0.18 > 203.118.144.45: icmp: echo request
```

work, and then concern yourself with any wireless issues. After you have IPsec running properly, you can remove the wire and start using 802.11 instead.

For this testing, I created a new gateway and put two NICs into the box. This box does NAT (*Network Address Translation*) using `ipnat` and will use `ipf` as a firewall. It's easier to manage with the entire wireless network in a separate subnet. If necessary, I can disconnect the entire subnet by unplugging a single cable or powering off the WAP.

Your NAT box will want to do some forwarding of packets. I recommend the use of `ipf` and `ipnat`. I also use `ipmon`. I have these entries in `/etc/rc.conf`:

```
gateway_enable="YES"
ipfilter_enable="YES"
ipnat_enable="YES
ipmon_enable="YES"
```

Firewall and NAT rules are beyond the scope of this article, but those two links should give you a running start.

A Short Introduction to IPsec

Ipsec (*http://beta.freebsddiary.org/topics.php#firewalls*) can create (*http://beta.freebsddiary.org/topics.php#nat*) a point-to-point tunnel between two hosts. Being encrypted, the data will be safe from prying eyes, and the gateway won't accept modified packets, since they lack an authentic signature. IPsec can also secure traffic between two networks or a network and a gateway. Other configuration options are available, but I will concentrate on just network, not point to point.

The key point to realize is that IPsec cannot exist on its own. You need to have IPsec at both ends of the communication. You cannot just slap IPsec onto your laptop and expect it to work wherever you go. This is why I have decided to create a wireless gateway through which all my wireless traffic will flow.

Network diagram

The following diagram (created with Xfig *http://www.xfig.org/*) illustrates my wireless network. My laptop sits at 10.0.0.10 and communicates over wireless (802.11) to my WAP. The WAP connects to a dedicated gateway box (via a hub) which sits between the WAP and my LAN (see Figure 1).

Listing 7. *The diff*

```
--- racoon.conf.laptop       Wed Sep 15 19:26:03 2004
+++ racoon.conf.gateway Wed Sep 15 19:31:46 2004
@@ -33,6 +33,8 @@
        #isakmp 202.249.11.124 [500];
        #admin [7002];           # administrative's port by kmpstat.
        #strict_address;         # required all addresses must be bound.
+
+       isakmp 10.0.0.1;
 }

 # Specification of default various timer.
```

Listing 8. *The preshared key file*

```
Pre-shared key File
  Pre-shared key file defines a pair of the identifier and the shared
  secret key which are used at Pre-shared key authentication method in
  phase 1.  The pair in each lines are separated by some number of blanks
  and/or tab characters like hosts(5).  Key can be included any blanks
  because all of the words after 2nd column are interpreted as a secret
  key.  Lines start with #' are ignored.  Keys which start with ' are
  hexa-decimal strings.  Note that the file must be owned by the user ID
  running racoon(8) (usually the privileged user), and must not be accessi-
  ble by others.
```

Listing 9. *The ping missing a few steps*

```
[dan@laptop:~] $ ping -A 192.168.0.18
PING xeon.unixathome.org (192.168.0.18): 56 data bytes
64 bytes from 192.168.0.18: icmp_seq=0 ttl=63 time=4.880 ms
64 bytes from 192.168.0.18: icmp_seq=1 ttl=63 time=4.847 ms
64 bytes from 192.168.0.18: icmp_seq=2 ttl=63 time=5.126 ms
64 bytes from 192.168.0.18: icmp_seq=3 ttl=63 time=5.209 ms
64 bytes from 192.168.0.18: icmp_seq=6 ttl=63 time=5.468 ms
64 bytes from 192.168.0.18: icmp_seq=7 ttl=63 time=4.838 ms
64 bytes from 192.168.0.18: icmp_seq=8 ttl=63 time=5.270 ms
^C
--- xeon.unixathome.org ping statistics ---
9 packets transmitted, 7 packets received, 22% packet loss
round-trip min/avg/max/stddev = 4.838/5.091/5.468/0.226 ms
[dan@laptop:~] $
```

Any traffic coming in over the wireless network must pass through the WAP and then the wireless gateway. This gateway has two NICs (one at 10.0.0.1, the other at 192.168.0.55). These are conventional, wired NICS. There is no Wifi in this gateway, but there certainly could be. I have chosen to use a WAP instead. The WAP plugs into a hub, and 10.0.0.1 on the gateway plugs into the same HUB. The other NIC plugs into the main LAN.

The IPsec database

IPsec uses a database to decide how to treat traffic. The database contains the rules on what traffic to encrypt and how to encrypt it. The two main types of rules are policy and association. The Security Policy Database (SPD) determines what traffic IPsec should handle. The Security Association Database (SAD) specifies how to encrypt that traffic.

The main tool for manipulating the database is `setkey(8)` (*http://www.freebsd.org/cgi/man.cgi?query=setkey&apropos=0&sektion=0&manpath=FreeBSD+4.10-RELEASE&format=html*). I will show you one way to use that tool later. Usually, you place these rules in `/etc/ipsec.conf`.

Creating the network tunnel

These rules cause the encryption of all traffic between the network (10.0.0.0/24) and the gateway (10.0.0.1). We will use ESP (*Encapsulating Security Payload*) as found in RFC 2406 (*http://www.faqs.org/rfcs/rfc2406.html*). This ensures that nobody can read your data (see Listing 1).

The first two rules (`add`) are SAD entries. The next two rules (`spdadd`) are SPD entries.

The add items set up the encryption keys for communication between the two computers. Be sure to use different keys; however, if you use IKE you won't need keys. The values shown are just to keep things easy. The `spdadd` items set up the actual tunnel between the two computers. In brief, the above directives mean:

- Between `10.0.0.1` and `10.0.0.10`, use the index `691`, the encryption algorithm known as `rijndael-cbc`, and use a shared secret of `"1234567890123456"`.
- Similarly, in the other direction between `10.0.0.10` and `10.0.0.1`, use the index `693`, the same encryption algorithm, and the same shared secret.
- All outgoing communication between the network (`10.0.0.0/24`) and everywhere else (`0.0.0.0`) must go (`require`) through a `tunnel`.
- In the other direction, incoming communication between everywhere else (`0.0.0.0`) and the network (`10.0.0.0/24`) must come (`require`) from a tunnel.

Gateway rules

The rules for the gateway are very similar to the laptop rules and also are slightly symmetric (see Listing 2).

You can add these rules manually using `setkey -c` and then copy and paste the rules from above (after making adjustments so they refer to your IP addresses, not mine). To exit, press Ctrl-D. When testing, I actually keep it running and copy and paste the commands directly. I use these commands to clear out existing database entries before adding new ones.

```
flush;
spdflush;
```

The `flush` command clears out the SAD entries, while `spdflush` clears out the SPD entries.

In the policy statements (`spdadd`), the second line above states that all traffic from `10.0.0.0/24` to anywhere requires `ESP`. The fourth line states that all traffic from anywhere to `10.0.0.0/24` also requires `ESP`. Combined, these two directives ensure that all traffic from anywhere to anywhere on this network must use `ESP`.

While I tested these rules, I kept them on a local web site. That made it easier to copy and paste the rules from the browser. I'm not suggesting that you publicly publish your rules. This is just a debugging tool. Mind you, the only parts you need to keep secret are the keys (in my example, `1234567890123456`).

If you place your rules in `/etc/ipsec.conf` and have `ipsec_enable="YES"` in your `/etc/rc.conf`, the system will load them at boot-up.

When loading the rules, you'll need to coordinate them. I sat at the gateway console with my laptop beside me. That way, if I messed up the rules, I could reset them without moving. The provided rules worked for me. They should work for you too. If they don't, go back to square one and verify your rules. Ensure that the subnet and the IP addresses are what they should be.

After you implement these rules, the gateway will reject all non-ESP traffic. Furthermore, anyone attempting to communicate with the gateway must have the shared secrets. If you change the secret, nothing will pass the gateway.

Confirming traffic

By this point, you have IPsec on both machines and you have set the IPsec database rules. Traffic is flowing. Now you want to confirm the encapsulation, so nothing should appear in plain text. Here is how I did that.

On my wireless gateway, the `dc0` device has an IP address of `10.0.0.1`. All traffic from the laptop will come in on that NIC. I issued this command to view that traffic: see Listing 3. The above shows `dhclient` starting on the laptop. A bit of ARP traffic follows. If all you see via tcpdump is stuff like this, then you're good to go.

```
13:42:18.225304 arp who-has 10.0.0.1 tell 10.0.0.10
13:42:18.225450 arp reply 10.0.0.1 is-at 0:32:91:32:91:32
```

If all you see is `arp`, then you're good to go.

You should see what the IPsec traffic looks like. Have a look. Shorten the above command to this: see Listing 4.

The above tcpdump is of HTTP traffic as my laptop accessed my development copy of FreshPorts (*http://www.freshports.org/*).

Here is how a ping looks: see Listing 5.

Note: this is all ESP. The following is an example of traffic that does not use `ipsec`: see Listing 6.

There you go. All good. Nothing passes through the gateway unless it matches the rules. The shared secret is the key to this security. This would be more secure if the secret changed occasionally, though.

Racoon Likes to Keep Secrets

Instead of manually changing the shared secrets in your `/etc/ipsec.conf` file, you can keep one shared secret and use the IKE protocol to negotiate a key. Racoon (*http://www.freshports.org/security/racoon/*) speaks IKE (ISAKMP/Oakley), which is a key management protocol.

Listing 10. *Some of the tcpdump traffic*

```
16:36:34.517794 10.0.0.10 > 10.0.0.1: ESP(spi=0x02dc8063,seq=0x1d)
16:36:34.867830 10.0.0.10 > 10.0.0.1: ESP(spi=0x02dc8063,seq=0x1e)
16:36:34.873592 10.0.0.1 > 10.0.0.10: ESP(spi=0x0751ce23,seq=0x1d)
16:36:35.737921 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? inf[E]: [encrypted hash]
16:36:35.904270 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
16:36:36.604941 10.0.0.1.500 > 10.0.0.10.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
16:36:36.605565 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
16:36:36.887859 10.0.0.10 > 10.0.0.1: ESP(spi=0x01f4fcea,seq=0x1)
16:36:37.897889 10.0.0.10 > 10.0.0.1: ESP(spi=0x01f4fcea,seq=0x2)
```

Listing 11. *Making IPsec work again*

```
[root@laptop:/home/dan] # setkey -f /etc/ipsec.conf
[root@laptop:/home/dan] # tcpdump -ni wi0 tcpdump: listening on wi0
17:21:20.168434 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
17:21:46.426485 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
17:21:47.112010 10.0.0.1.500 > 10.0.0.10.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
17:21:47.113115 10.0.0.10.500 > 10.0.0.1.500: isakmp: phase 2/others
     ? oakley-quick[E]: [encrypted hash]
17:21:47.375549 10.0.0.10 > 10.0.0.1: ESP(spi=0x02c19c7a,seq=0x1)
17:21:48.385549 10.0.0.10 > 10.0.0.1: ESP(spi=0x02c19c7a,seq=0x2)
17:21:48.390088 10.0.0.1 > 10.0.0.10: ESP(spi=0x041e48fb,seq=0x1)
17:21:49.395564 10.0.0.10 > 10.0.0.1: ESP(spi=0x02c19c7a,seq=0x3)
17:21:49.399754 10.0.0.1 > 10.0.0.10: ESP(spi=0x041e48fb,seq=0x2)
```

I installed Racoon from the ports tree. Here are the configuration files from both my laptop (*http://beta.freebsddiary.org/samples/racoon.conf.laptop*) and the gateway (*http://beta.freebsddiary.org/samples/racoon.conf.gateway*). The only difference between the two files is that I instruct the gateway to listen on only one address (it has two NICs). Here is the diff, if you are interested: see Listing 7.

The configuration file tells Racoon the main things it needs to know. One of the items is the preshared key file. Look for this directive:

```
# search this file for pre_shared_key with various ID key.
path pre_shared_key "/usr/local/etc/racoon/psk.txt" ;
```

From man `racoon.conf`: see Listing 8.

Listing 12. *The configuratio file*

```
default-lease-time 600;
max-lease-time 7200;

authoritative;
ddns-update-style none;

option domain-name "example.org";

#
# this points to my local DNS server on the other
# side of the wireless gateway
#
option domain-name-servers 192.168.0.101;

default-lease-time 86400;
max-lease-time 86400;


# This is a very basic subnet declaration.

subnet 10.0.0.0 netmask 255.255.255.0 {
        option routers 10.0.0.1;
        range 10.0.0.192 10.0.0.207; # this is 10.0.0.200/27 => (28)

        host laptop.example.org {
                option dhcp-client-identifier "laptop.example.org";
                fixed-address laptop.example.org;
        }
}
```

Here is the `/usr/local/etc/racoon/psk.txt` file on my laptop:

```
10.0.0.1 MySecretValue
```

Here is the file from the wireless gateway:

```
10.0.0.10 MySecretValue
```

With these values, Racoon on my laptop knows that when it talks to 10.0.0.1 (the gateway) it should use the shared key `MySecretValue`. Similarly, the Racoon running on the gateway knows to use the same shared key when speaking to 10.0.0.10 (my laptop).

To start Racoon, issue this command:

```
/usr/local/etc/rc.d/racoon.sh start
```

If you're running a recent version of FreeBSD (for example, 4.10-RELEASE), add this entry in `/etc/rc.conf`:

```
racoon_enable="YES"
```

Ensure that Racoon is running on both the laptop and the gateway. Then remove the SAD entries from both machines, and Racoon should negotiate a new set of keys.

```
# setkey -c
flush;
^D
```

If that doesn't work, try running Racoon in the foreground (after first stopping the one running in the background):

```
/usr/local/sbin/racoon -F
```

It should just work.

Fun with Keys: Understanding What Happens

I thought it might be interesting to clear out the SAD entries and see what happens when it comes time to negotiate new keys. I started a ping running and ran tcpdump while I issued this command:

```
# setkey -F
```

As you can see, the ping missed a few steps. That is understandable see Listing 9.

Two pings went missing and never reached the machine on the other side of the gateway. Here is some of the tcpdump traffic: see Listing 10.

The lines that contain isakmp represent the two Racoon daemons negotiating a new key.

As an experiment, I turned off IPsec on my laptop by commenting out the `ipsec_enable` line in `/etc/rc.conf`. Then I rebooted. Interestingly, I still received an IP address from my DHCP server on the other side of the wireless gateway. However, I could not get through the gateway. Even simple pings to the gateway went unanswered. At this time, the firewall on the wireless gateway allowed all traffic to pass. Therefore, the gateway rejected the traffic because it did not use IPsec.

To make IPsec work again, I did this while the ping was still running: see Listing 11.

The first line populates the SPD (Actually, the command populates only the SAD because the file in question contains only add commands. I commented out the spdadd commands) database, based upon the data within the file `/etc/ipsec.conf`. From there, Racoon must negotiate a new key. It took some time (about 26 seconds), but Racoon eventually succeeded. Immediately thereafter, the pings resumed.

DHCP Server

I mentioned above that I could still receive an IP address from my DHCP server that was running on my gateway. I had not previously mentioned it, but I think it might be useful to you. Here are the basics; the rest you should be able to piece together yourself.

Installing dhcpd

To install the dhcp server, I did this:

```
# cd /usr/ports/net/isc-dhcp3-server
# make install clean
```

Starting at boot time

This will install `/usr/local/etc/rc.d/isc-dhcpd.sh`. Remember to add `dhcpd_enable="YES"` to `/etc/rc.conf`, or the script will not start the server. I also added `dhcpd_ifaces="dc0"` so that dhcpd would listen only on the one NIC–the one attached to the same hub as the WAP.

The configuration file

The port installs `/usr/local/etc/dhcpd.conf`. It is full of examples, but here is what I'm using, slightly altered to protect the obvious: see Listing 12.

That fixed-address relates to the following entry in `/etc/dhclient.conf`:

```
send dhcp-client-identifier "laptop.example.org";
```

This allows the laptop to tell the DHCP server who it is, which I use to assign a specific IP address. Note: this method is convenient, but it is not necessarily secure. If you're like me, sometimes using wireless with your laptop and sometimes connecting via wire, then you might want to give it a different IP address depending on where it is. I do that by having two DHCP servers. I'm sure someone will suggest another method.

Is That Enough?

Now you have your wireless laptop connected to your LAN, with encrypted and secured traffic. Nobody else can use your gateway unless they guess the secret key. That's not easy. The key will change from time to time, so even if someone guesses a key, there's a new one coming along soon. The only thing you have to secure is the preshared secret. Don't use what I've supplied. Come up with something odd, even some random values. Pick some text from IRC. That should work.

Are you being paranoid enough? I think for one of my next tasks I will look for any unusual traffic coming on the gateway, from any IP other than my laptop. There are whole books written on intrusion detection, and that topic is well beyond what I can cover here.

Enjoy.

DAN LANGILLE

Dan is a highly experienced computer professional with a wide range of skills. He has worked as a database administrator, software developer, and a system administrator. He has a strong belief in creating good documentation and developing procedures for deployment and maintenance.

Redundant firewalls

with OpenBSD, CARP and pfsync

Firewalls are among the most critical network components, since their failure may cause entire groups of machines to remain offline. The damage may range from the public (web, mail, etc.) servers to become unreachable to the outside world up to being unable to surf this web site!

What you will learn…

- How the CARP and PFSYNC protocol work
- How to configure highly-available firewalls with OpenBSD and PF

What you should know…

- Installing and configuring a base OpenBSD system
- Configuring OpenBSD's Packet Filter

Using firewall clusters can dramatically reduce these risks, making a firewall failure invisible to users. Furthermore, maintenance (patching, upgrading, rebooting...) becomes much easier and faster when relying on a backup machine, thus indirectly increasing systems security and reliability.

On the other hand, it's true that redundancy raises hardware costs and can't solve each and every problem, like transparent transfer of certain protocols (e.g. SSH or IRC) between systems or synchronizing data between clustered machines (in matter of fact, we will rely on two different protocols for failover and synchronization).

The tools we will use to build our failover cluster are:

- OpenBSD (*http://www.openbsd.org/*) – largely considered one of the most secure OSes around, with *only two remote holes in the default install, in a heck of a long time!*;
- Packet Filter (PF *http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*) – OpenBSD's system for filtering TCP/IP traffic and doing Network Address Translation;
- CARP (*Common Address Redundancy Protocol http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*) – the protocol that achieves system redundancy, by having multiple computers creating a single, virtual network interface between them;
- pfsync (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4*) – the protocol that allows PF state tables to be synchronized between multiple firewalls.

A good knowledge of OpenBSD and PF is assumed, since we won't dwell much on topics like PF maintenance commands and rules syntax. Anyway, the appendix contains some useful links for delving into these topics.

Network layout

First, let's take a look at the environment in which our firewall cluster will operate. It's a very simple and *classic* network, made up of:

- a DMZ (172.16.240.0/24), containing the publicly accessible machines (e.g. web and mail servers) and the intrusion detection sensors;
- a LAN (172.16.0.0/24), containing clients and servers not accessible from the public Internet (file server, DHCP server, internal DNS server...);
- a router, in a small subnet (172.16.250.0/24), to connect the network to the Internet.

This environment requires that we setup two firewall clusters: the first separating the DMZ from the Internet

Listing 1. *Services enabled through inetd(8)*

```
$ grep -v ^# /etc/inetd.conf
ident           stream  tcp     nowait  _identd /usr/libexec/identd   identd  -el
ident           stream  tcp6    nowait  _identd /usr/libexec/identd   identd  -el
127.0.0.1:comsat dgram  udp     wait    root    /usr/libexec/comsat   comsat
[::1]:comsat    dgram   udp6    wait    root    /usr/libexec/comsat   comsat
daytime         stream  tcp     nowait  root    internal
daytime         stream  tcp6    nowait  root    internal
time            stream  tcp     nowait  root    internal
time            stream  tcp6    nowait  root    internal
$
```

(we won't take into account any router filtering); the second between the LAN and the DMZ. The network looks roughly like this: see Figure 1.

The utmost merit of this topology is that, needing two groups of firewalls, it will allow us to look over two slightly different cluster configurations. Jokes apart, these are some of its major benefits:

- in case of a firewall compromise, the LAN is protected by an additional layer of filtering (though it would be preferable to use different firewall platforms, to prevent attackers from compromising the internal firewalls with the same technique [MISC17]);
- a single (clustered) firewall, filtering both LAN and DMZ traffic, is a single point of failure;
- on each firewall, rules apply only to LAN or DMZ traffic, thus making PF rulesets cleaner and easier to maintain;

but there are also a few drawbacks:

- besides its own traffic, the DMZ must support the traffic load from the internal network to the Internet;
- double-filtering LAN traffic increases security but (slightly) affects performances;
- the cost of additional hardware may not be irrelevant.

Base configuration

Let's take a brief look at the base system configuration, which applies to all of our firewalls.

We won't go through the installation of the operating system, which is deeply documented on the OpenBSD web site (*http://www.openbsd.org/faq/faq4.html*). The only (obvious) remark is that you should install only the bare minimum, to prevent firewall security and reliability from being compromised by unnecessary software. Therefore, during installation, you only need to select file sets marked as *Required* by the documentation (*http://www.openbsd.org/faq/faq4.html#FilesNeeded*), i.e.:

- `bsd`, the kernel;
- `baseXX.tgz`, the base system;
- `etcXX.tgz`, the configuration files in `/etc`.

There should be no need to install the compiler (`compXX.tgz`), also to avoid providing such a useful tool to

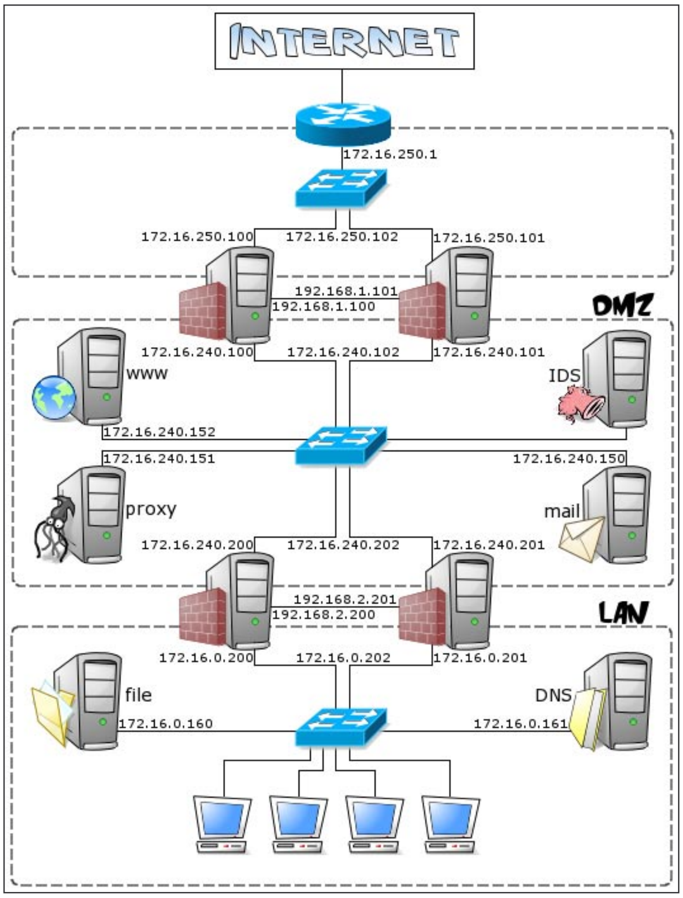


Figure 1. *Network*

possible intruders (see [PUIS] *http://www.oreilly.com/catalog/puis/*).

After the first reboot, we can start setting up some configuration files; by default, OpenBSD comes with very few services enabled through `inetd(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=inetd&sektion=8*): see Listing 1.

The system is considered secure also with these services turned on (see [ABSO] *http://www.absoluteopenbsd.com/*); anyway, disabling them all will make no harm.

It's also a good practice to edit the `/etc/motd` file to give as few information as possible about the system and to warn users, whether legitimate or not, that all access is being logged and that any unauthorized access will be prosecuted (see [PUIS] *http://www.oreilly.com/catalog/puis/*).

You should already have configured the network during installation; anyway, if you need to make some changes, these are the main files to edit:

- `/etc/hostname.if(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=hostname.if&sektion=5*) – containing information regarding the configuration of each network interface (address, netmask, etc.);
- `/etc/mygate(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=mygate&sektion=5*) – containing the address of the gateway host;
- `/etc/myname(5)` (http://www.openbsd.org/cgi-bin/man.cgi?query=myname&sektion=5) – containing the symbolic hostname (FQDN) of the machine;
- `/etc/resolv.conf(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=resolv.conf&sektion=5*) – containing the resolver configuration settings (name servers, local domain name, etc.).

Considering the large amount of DNS-based attacks, it is also preferable, especially on firewalls, not to rely on DNS to resolve names and addresses of the most critical systems, but rather inserting them into the `/etc/hosts(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=hosts&sektion=5*) file. To make sure the `/etc/hosts(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=hosts&sektion=5*) file has a higher priority than DNS, just check that the first line in `/etc/resolv.conf(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=resolv.conf&sektion=5*) is:

```
/etc/resolv.conf
lookup file bind
```

Packet Filter is enabled by default and loads rules from the `/etc/pf.conf(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5*) file; a different path can be specified by assigning it to the pf_rules variable in `/etc/rc.conf.local(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf&sektion=8*).

```
/etc/rc.conf.local
pf_rules=/new/path/to/pf.conf
```

You may also set `pflogd(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pflogd&sektion=8*) flags in the variable `pflogd_flags`. Last, don't forget to enable IP forwarding by issuing the command:

```
# sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
#
```

and to uncomment the following line in `/etc/sysctl.conf(5)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl.conf&sektion=5*) to re-enable it after reboot:

```
/etc/sysctl.conf
net.inet.ip.forwarding=1
```

The CARP protocol

CARP (*Common Address Redundancy Protocol*) is the protocol that achieves system redundancy, by having a group of hosts on the same network segment (*redundancy group*) to share an IP address, so that if any machine fails, another can respond in its place. CARP also allows a degree of load sharing between systems.

Although creating redundant firewalls is one of its most common uses, CARP isn't a firewall-specific protocol. It can be used to ensure service continuity and/or load sharing to a number of network services.

Initially, the OpenBSD team wanted to produce a free implementation of the IETF standard protocols, VRRP (*Virtual Router Redundancy Protocol*), defined in [RFC3768 *http://www.faqs.org/rfcs/rfc3768.html*], and HSRP (*Hot Standby Router Protocol*), defined in [RFC2281 *http://www.faqs.org/rfcs/rfc2281.html*]; but Cisco, claiming patent rights on it, firmly informed the OpenBSD community that Cisco would defend its patents for VRRP implementation (see [CARP] *http://www.openbsd.org/lyrics.html#35* for more details), thus forcing the OpenBSD developers to create a new, competing protocol designed to be fundamentally different from VRRP.

CARP is a multicast protocol, grouping several physical computers together under one or more virtual addresses. Of these, one system is the master and responds to all packets destined for the group; the other systems (backups) just stand by, waiting for any problems to take its place (as it happens among co-workers...).

At configurable intervals, the master advertises its operation on IP protocol number 112. If the master goes offline, the other hosts in the redundancy group begin to advertise. The host that's able to advertise most frequently becomes the new master. When the main system comes back up, it becomes a backup host by default, although it can be configured to try to become master again.

As you can see, CARP only creates and manages the virtual network interface; it's up to the system administrator to synchronize data between applications, using `pfsync(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4* which we'll discuss in the next chapter), rsync (*http://rsync.samba.org/*) or whatever protocol is appropriate for the specific application.

Configuration parameters

CARP configuration is done via the `sysctl(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=8*) and `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*) commands. There are four relevant `sysctl(3)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=3*) variables:

- `net.inet.carp.allow` – it defines whether the host handles CARP packets or not. It is enabled by default;
- `net.inet.carp.log` – it defines whether to log CARP errors or not. It may be a value between 0 and 7, corresponding to the `syslog(3)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=syslog&sektion=3*) priorities, and defaults to 2 (i.e. only CARP state changes are logged);
- `net.inet.carp.preempt` – if set to 0 (default), the host won't try to become master if it receives CARP advertisements from another master. Otherwise, it will try to become master if it is able to advertise more frequently than the current master. This option also enables failing over all interfaces in the event that one interface goes down. In fact, if one physical CARP-enabled interface goes down, CARP will change the `advskew` value (see below) to 240 on all other CARP-enabled interfaces, thus allowing the election of new masters on all subnets. The syntax for configuring CARP with `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*) is: see Listing 2.
- `carpN` – the name of the `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) virtual interface.
- `advbase`, `advskew` – these values determine the interval between two consecutive CARP advertisements. This interval (in seconds) is given by the formula (`advbase + (advskew / 255)`); increasing `advbase` (the default value is 1 second) will decrease network traffic but increase the delay in electing the new master. Small `advskew` values allow a host to advertise more frequently, increasing its probability to become master. `advbase` must be a value between 1 and 255, `advskew` between 0 (the default) and 254;
- `balancing` – sets the load balancing mode (which will be discussed later); valid modes are `arp`, `ip`, `ip-stealth`, and `ip-unicast`;
- `carpnodes` – a comma-separated list of `vhid:advskew` pairs to actually define how the load should be shared among the configured carp nodes (see below for further details);
- `carpdev` – specifies the physical interface that belongs to this redundancy group. By default, CARP uses the physical interface that belongs to the same subnet as the virtual interface;
- `carppeer` – allows you to specify the IP address of the other CARP peer(s), instead of using the default multicast group;

Listing 2. *The syntax for configuring CARP with ifconfig(8)*

```
ifconfig carpN create

ifconfig carpN [advbase n] [advskew n] [balancing mode]   \
[carpnodes vhid:advskew,vhid:advskew,...] [carpdev iface] \
[[-]carppeer peer_address] [pass passphrase] [state state] [vhid host-id]
```

- `pass` – the authentication password to use when talking to other CARP-enabled hosts in the redundancy group. This must be the same on all members of the group;
- `state` – Force a `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) interface into a certain state (`init`, `backup` or `master`);
- `vhid` – the Virtual Host ID. This is a unique number (between 1 and 255) that is used to identify the redundancy group to the other nodes on the network.

The demotion counter

Besides basic configuration, the `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*) command also allows you to tweak the CARP demotion counter, which is *a measure of how "ready" a host is to become master of a CARP group* (the higher the counter, the less ready the host). Let's see it in more detail.

CARP interfaces are divided in groups (by default all `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) interfaces are members of the "carp" interface group) and each group is assigned a demotion counter, whose value can be viewed by running the following command:

```
$ ifconfig -g carp
carp: carp demote count 0
```

The demotion counter comes in handy mainly when:

- you want to momentarily prevent a host from becoming master: for instance, at boot time, the `rc(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=rc&sektion=8*) script increases the demotion counter by 128 before starting the network and decreases it by the same amount once all interfaces have been initialized and all network daemons have been started (the demotion counter can't be set to an absolute value, but only increased or decreased by a certain amount):

```
/etc/rc
ifconfig -g carp carpdemote 128
[ ... ]
ifconfig -g carp -carpdemote 128
```

- you want to gracefully failover only a limited number of a host's `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) interfaces (not all of them, as it happens when an interface goes down and preempt is enabled). In the following example, we will failover the `carp1` and `carp2` interfaces and leave the state of the others unchanged: see Listing 3.

For further details on the CARP demotion counter, please refer to [PFFAQ *http://www.openbsd.org/faq/pf/carp.html#forcefail*].

Load balancing

CARP provides two different methods for load balancing incoming network traffic among a set of CARP-

Listing 3. *The demotion counter*

```
# ifconfig carp1 group morituri
# ifconfig carp2 group morituri
# ifconfig morituri
carp1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        carp: MASTER carpdev sis0 vhid 1 advbase 1 advskew 100
        groups: carp morituri
        inet 1.2.3.4 netmask 0xffffff00 broadcast 1.2.3.255
carp2: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        carp: MASTER carpdev sis1 vhid 2 advbase 1 advskew 100
        groups: carp morituri
        inet 2.3.4.5 netmask 0xffffff00 broadcast 2.3.4.255
# ifconfig -g morituri
morituri: carp demote count 0
# ifconfig -g morituri carpdemote 50
# ifconfig -g morituri
morituri: carp demote count 50
```

Shameless stock web 2.0-like art bug to get your attention.

We used the color red to grab your attention, it's an old trick.

Now, let us introduce ourselves; we're Superb Internet, and we want your business. We sell dedicated servers, hosting, rack space, etc. Oh yeah, and we have an awesome network, it's up 100% of the time.

30-Day Money-Back Guarantee SLA

Award-Winning Customer Service

100% Uptime Guarantee

Man, that istock girl is HOT! We used her photo to make you actually want to talk to us.

Yup, awesome network map. As awesome as our real Tier-1 network, with three amazing datacenters.

WWW.SUPERB.NET/NO_BS

Go here and check us out...you might like it

Superb INTERNET

Listing 4. *Creating the carp* devices and configure them with ifconfig(8)*

```
mickey# ifconfig carp0 172.16.0.202/24 vhid 1 pass password1 advbase 1 advskew 0
mickey# ifconfig carp1 172.16.240.202/24 vhid 2 pass password2 advbase 1 advskew 0

minnie# ifconfig carp0 172.16.0.202/24 vhid 1 pass password1 advbase 1 advskew 100
minnie# ifconfig carp1 172.16.240.202/24 vhid 2 pass password2 advbase 1 advskew 100
```

enabled hosts: ARP balancing and IP balancing. Both methods require that you first create a load balancing group by configuring, on each balanced `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) interface, as many VHIDs as hosts in the balancing group; the `advskew` on each VHID will be configured so that each host will be the master on a separate VHID (see below for a practical example).

ARP balancing works by applying a hash function to the source MAC address of ARP requests to determine which VHID should handle the request. The ARP request will be answered solely by the host whose `carp(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4*) interface is master for that VHID. ARP load balancing can be enabled through `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*) by setting the value of the balancing option to `arp` on all hosts; for instance:

```
# ifconfig carp0 balancing arp carpnodes 1:0,2:100
```

IP load balancing works in a very similar manner to ARP balancing, but uses the hash of the source and destination addresses of the IP packet to determine which VHID (and therefore which host) should accept the packet.

IP balancing requires that traffic destined to the CARP address be received by all CARP hosts. It can be enabled using `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*), by setting the `balancing` option to `ip`; this will cause CARP to use a multicast MAC address, forcing the switch to send incoming traffic to all nodes in the redundancy group. For example:

```
# ifconfig carp0 balancing ip carpnodes 1:0,2:100
```

Alternatively, you can set the `balancing` option to `ip-stealth` (stealth mode), in order to prevent hosts from sending packets with their virtual MAC address as source; this will prevent the switch from learning the virtual MAC address, forcing it to flood the traffic to all its ports. Last, if you're using a hub or a switch that supports some kind of monitoring mode, you can set `balancing` to `ip-unicast`.

The choice between the two load balancing mechanisms mostly depends on the network environment in which the systems will be placed: ARP balancing only works for clients in the local network and cannot balance traffic that crosses a router, as routed traffic always contains the MAC address of the router as its source address. Therefore, if clients are on remote networks, IP balancing is the only option; the only drawback of IP balancing is that traffic destined towards the load balanced IP addresses must be received by all CARP-enabled hosts, resulting in a higher network load.

Table 1. *Recaping the firewalls adresses from the network diagram*

	Mickey	Minnie	Virtual address
LAN	172.16.0.200	172.16.0.201	172.16.0.202
DMZ	172.16.240.200	172.16.240.201	172.16.240.202
pfsync	192.168.2.200	192.168.2.201	
	Donald	Daisy	Virtual address
DMZ	172.16.240.100	172.16.240.101	172.16.240.102
Internet	172.16.250.100	172.16.250.101	172.16.250.102
pfsync	192.168.1.100	192.168.1.101	

Parameters configuration

Now it's time to configure CARP on our firewalls. To examine two slightly different CARP configurations, we will set up the two internal firewalls (Mickey and Minnie, between LAN and DMZ) in active/stand-by mode, with only one system filtering the whole network traffic and the other one acting as a hot spare; the two external firewalls (Donald and Daisy, separating the DMZ from the internet), instead, will be in active/active mode, sharing the traffic load.

So let's recap the firewalls adresses, as we have seen them in the network diagram: see Table 1.

Active/standby configuration

Let's start with Mickey and Minnie: first, we need to create the carp* devices and configure them with `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*): see Listing 4.

We have just created the interfaces, assigned them an IP address, a virtual host ID (1 on the LAN, 2 on the DMZ) and a password (probably not the most secure...) for authentication. We also decided that, whenever possible, Mickey will be the master; this is done by giving Minnie a higher `advskew` value (100), thus making the interval between its advertisements (1+100/255) higher than the interval between Mickey's advertisements (1+0/255). And we've seen before that the host that's able to advertise most frequently, becomes the new master.

Furthermore, by setting `net.inet.carp.preempt` to `1` on Mickey, we ensure that Mickey will always try to become the master:

```
mickey# sysctl net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1
```

To make these settings permanent after reboot, we just need to edit the `/etc/hostname.carp*` and `/etc/sysctl.conf` files on Mickey: see Listing 5 and on Minnie: see Listing 6.

Note: to make the adoption of CARP easier on pre-existing networks, CARP allows using the physical address of a host as the virtual address of the whole redundancy group.

Active/active configuration

Now let's get on to Donald and Daisy and start by configuring their DMZ interfaces. As before, we will create the `carp0` device on each machine, but this time, to enable load balancing, we will use the `carpnodes` option to assign two different Virtual Host IDs to the interface (VHIDs 3 and 4).

Listing 5. *Editing the /etc/hostname.carp* and /etc/ sysctl.conf files on Mickey*

```
/etc/hostname.carp0
inet 172.16.0.202 255.255.255.0 172.16.0.255 vhid 1 pass password1 advbase 1 advskew 0
/etc/hostname.carp1
inet 172.16.240.202 255.255.255.0 172.16.240.255 vhid 2 pass password2 advbase 1 advskew 0
/etc/sysctl.conf
[...]
net.inet.carp.preempt=1
```

Listing 6. *Editing the /etc/hostname.carp* and /etc/ sysctl.conf files on Minnie*

```
/etc/hostname.carp0
inet 172.16.0.202 255.255.255.0 172.16.0.255 vhid 1 pass password1 advbase 1 advskew 100
/etc/hostname.carp1
inet 172.16.240.202 255.255.255.0 172.16.240.255 vhid 2 pass password2 advbase 1 advskew 100
```

Listing 7. *Forcing Daisy to become master for VHID 4*

```
donald# ifconfig carp0 172.16.240.102/24 balancing ip carpnodes 3:0,4:100 \
> pass password3
donald# sysctl net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1

daisy# ifconfig carp0 172.16.240.102/24 balancing ip carpnodes 3:100,4:0 \
> pass password3
daisy# sysctl net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1
```

On VHID 3, we will set the `advskew` of Donald and Daisy to 0 and 100 respectively: this will ensure that Donald becomes master for that VHID; on VHID 4, instead, we will do the opposite, by setting the `advskew` of Donald and Daisy to 100 and 0 respectively, in order to force Daisy to become master for VHID 4: see Listing 7.

We now have two redundancy groups with the same IP address, but each with a different master: see Listing 8.

To make these settings permanent across reboots, we need to edit the startup files on Donald: see Listing 9 and Daisy: see Listing 10. Now we just have to do the same on the external network interfaces: see Listing 11 and edit the startup files on Donald: see Listing 12.

Though the above configuration involves only a couple of machines, it can be easily extended to up to 32 hosts. *One last note:* load sharing won't probably achieve a perfect 50/50 distribution between the two machines, since CARP uses a hash of the source and destination IP addresses to determine which system should accept a packet, not the actual load.

Listing 8. *Two redundancy groups with the same IP address, but each with a different master:*

```
donald# ifconfig carp0
carp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 01:00:5e:00:01:01
        carp: carpdev rl1 advbase 1 balancing ip
                state MASTER vhid 3 advskew 0
                state BACKUP vhid 4 advskew 100
        groups: carp
        inet 172.16.240.102 netmask 0xffffff00 broadcast 172.16.240.255
        inet6 fe80::2c0:a8ff:fe8e:b112%carp0 prefixlen 64 scopeid 0x5

daisy# ifconfig carp0
carp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        lladdr 01:00:5e:00:01:01
        carp: carpdev rl1 advbase 1 balancing ip
                state BACKUP vhid 3 advskew 100
                state MASTER vhid 4 advskew 0
        groups: carp
        inet 172.16.240.102 netmask 0xffffff00 broadcast 172.16.240.255
        inet6 fe80::219:d2ff:fe02:6469%carp0 prefixlen 64 scopeid 0x5
```

Listing 9. *Editing the startup files on Donald*

```
/etc/hostname.carp0
inet 172.16.240.102 255.255.255.0 172.16.240.255 balancing ip carpnodes 3:0,4:100 pass password3
/etc/sysctl.conf
[...]
net.inet.carp.preempt=1
```

Listing 10. *Editing the startup files on Daisy*

```
/etc/hostname.carp0
inet 172.16.240.102 255.255.255.0 172.16.240.255 balancing ip carpnodes 3:100,4:0 pass password3
/etc/sysctl.conf
[...]
net.inet.carp.preempt=1
```

The pfsync protocol

Pfsync is the protocol used by Packet Filter (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*) to manage and update state tables, which allow for stateful inspection and NAT. By default, state change messages are sent out on the synchronization interface using IP multicast packets. The protocol is IP protocol 240 and the multicast group used is 224.0.0.240. We will use it to synchronize state tables among firewalls of the same redundancy group and, in the event of a failover, allow network traffic to flow uninterrupted through the new master firewall.

`Pfsync(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4*) is also the name of the pseudo-device on which PF state table changes pass (except states created by rules marked with the `no-sync` keyword or by `pfsync(4)` *http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4 packets*). `pfsync(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4*) can be configured with a physical synchronization interface, in order to merge the state tables of multiple firewalls.

The physical synchronization interface can be set through `ifconfig(8)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ifconfig&sektion=8*), using the `syncdev` parameter; for example, on our firewalls, we can write:

```
# ifconfig pfsync0 syncdev rl2
```

assuming that the `rl2` interface is, on each host (see picture), the interface on the 192.168.1.0/24 subnet (for Mickey and Minnie) or 192.168.2.0/24 (for Donald and Daisy) and cross-cabled to the *beloved* firewall.

Crossover cables are recommended because the pfsync protocol doesn't provide any cryptography or authentication mechanism; if you don't use a secure network, like a crossover cable, an attacker may use spoofed pfsync packets to alter the firewalls state tables and bypass filter rules.

Alternatively, you can use the `syncpeer` keyword to specify the address of the firewall to synchronize with. The system will use this address, instead of broadcast, as the destination of `pfsync(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4*) messages and you can use `Ipsec` (*http://www.kernel-panic.it/openbsd/vpn/index.html*) to protect the communication. In this case, syncdev must be set to the `enc(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4*) pseudo-device, which encapsulates/decapsulates

Listing 11. *External network interfaces*

```
donald# ifconfig carp1 172.16.250.102/24 balancing ip carpnodes 5:0,6:100 \
> pass password5

daisy# ifconfig carp1 172.16.250.102/24 balancing ip carpnodes 5:100,6:0 \
> pass password5
```

Listing 12. *External network interfaces. Editing the startup files on Donald*

```
/etc/hostname.carp1
inet 172.16.250.102 255.255.255.0 172.16.250.255 balancing ip carpnodes 5:0,6:100 pass password5
and Daisy:
/etc/hostname.carp1
inet 172.16.250.102 255.255.255.0 172.16.250.255 balancing ip carpnodes 5:100,6:0 pass password5
```

Listing 13. *Normal filtering*

```
# External DNS
pass in  on $int_if inet proto { tcp, udp } from $int_if:network to $dns_srv \
     port domain
pass out on $ext_if inet proto { tcp, udp } from $ext_if to $dns_srv \
     port domain
```

Listing 14. *A basic PF ruleset for our external firewalls, Donald and Daisy*

```
/etc/pf.conf
################################################################################
# Macros and lists                                                             #
################################################################################

ext_if  = rl0                           # External interface
int_if  = rl1                           # DMZ interface
pfs_if  = rl2                           # Pfsync interface
carp_if = carp1                         # External CARP interfaces

mail_srv = "mail.kernel-panic.it"                               # Mail server
web_srv  = "{ www1.kernel-panic.it, www2.kernel-panic.it }""    # Web servers
dns_srv  = "{ dns1.isp.com, dns2.isp.com }"                     # DNS servers
int_fw   = "{ mickey.kernel-panic.it, minnie.kernel-panic.it }" # Internal fw

mail_ports = "{ smtp, imap, imaps }"    # Mail server ports
web_ports  = "{ www, https }"           # Web server ports
# Allowed incoming ICMP types
icmp_types = "{ echoreq, timex, paramprob, unreach code needfrag }"

# Private networks (RFC 1918)
priv_nets = "{ 127.0.0.0/8, 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 }"

################################################################################
# Options, scrub and NAT                                                       #
################################################################################

set block-policy drop
set loginterface $ext_if
set skip on lo

# NAT outgoing connections
nat on $ext_if from !$ext_if to any -> $ext_if

# Redirect web services (with load balancing)
rdr on $ext_if inet proto tcp from any to $carp_if port $web_ports -> $web_srv \
    round-robin sticky-address

# Redirect mail services
rdr on $ext_if inet proto tcp from any to $carp_if port $mail_ports -> $mail_srv

################################################################################
# Filtering rules                                                              #
################################################################################
```

Listing 14. *A basic PF ruleset for our external firewalls, Donald and Daisy*

```
block all                          # Default deny
block in quick from urpf-failed    # Spoofed address protection

# Scrub incoming packets
match in all scrub (no-df)

pass quick on $pfs_if proto pfsync                   # Enable pfsync
pass quick on { $int_if, $ext_if } proto carp        # Enable CARP

block in  quick on $ext_if from $priv_nets to any
block out quick on $ext_if from any to $priv_nets

# Mail server
pass in  on $ext_if inet proto tcp from any to $mail_srv port $mail_ports
pass out on $int_if inet proto tcp from any to $mail_srv port $mail_ports
pass in  on $int_if inet proto tcp from $mail_srv to any port smtp
pass out on $ext_if inet proto tcp from $ext_if to any port smtp modulate state

# Web servers
pass in  on $ext_if inet proto tcp from any to $web_srv port $web_ports \
     synproxy state
pass out on $int_if inet proto tcp from any to $web_srv port $web_ports

# ICMP
pass in  inet proto icmp all icmp-type $icmp_types
pass out inet proto icmp all

# DNS
pass in  on $int_if inet proto { tcp, udp } from $int_if:network to $dns_srv \
     port domain
pass out on $ext_if inet proto { tcp, udp } from $ext_if to $dns_srv \
     port domain

# Internet web servers
pass in  on $int_if inet proto tcp from $int_fw to any port $web_ports
pass out on $ext_if inet proto tcp from $ext_if to any port $web_ports \
modulate state
```

Bibliography

- *http://www.openbsd.org/faq/faq6.html#CARP* – The Common Address redundancy Protocol (CARP)
- *http://www.countersiege.com/doc/pfsync-carp/* – Firewall Failover with pfsync and CARP
- *http://home.nuug.no/~peter/pf/* – Firewalling with OpenBSD's PF packet filter, Peter N. M. Hansteen, 2008
- *http://www.nostarch.com/pf.htm* – The Book of PF, Peter N. M. Hansteen, No Starch Press, 2008
- *http://www.reedmedia.net/books/pf-book/* – The OpenBSD PF Packet Filter Book, Jeremy C. Reed, Reed Media Services, 2006

References

- [MISC17] – Filtrage applicatif: le cas des clients web, mail et p2p (MISC Nº17)
- [PUIS] *http://www.oreilly.com/catalog/puis/* – Practical UNIX and Internet Security, Simson Garfinkel and Gene Spafford, O'Reilly, 2003
- [ABSO] *http://www.absoluteopenbsd.com/* – Absolute OpenBSD, Michael W. Lucas, No Starch Press, 2003
- [CARP] *http://www.openbsd.org/lyrics.html#35* – "CARP License" and "Redundancy must be free"
- [PFFAQ] *http://www.openbsd.org/faq/pf/index.html* – PF: The OpenBSD Packet Filter
- [RFC3768] *http://www.faqs.org/rfcs/rfc3768.html* – RFC 3768, Virtual Router Redundancy Protocol (VRRP)
- [RFC2281] *http://www.faqs.org/rfcs/rfc2281.html* – RFC 2281, Cisco Hot Standby Router Protocol (HSRP)

`ipsec(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec&sektion=4*) traffic. E.g.:

```
# ifconfig pfsync0 syncpeer 192.168.1.101 syncdev enc0
```

To make these settings permanent after reboot, we need to edit the `/etc/hostname.pfsync0` file on each firewall:

```
/etc/hostname.pfsync0
up syncdev rl2
```

PF rules!

The impact of CARP and `pfsync(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4*) on Packet Filter rules is really minimal. First, you need to let the PFSYNC and CARP protocols pass on their own interfaces:

```
pass quick on rl2 proto pfsync keep state (no-sync)
pass on { rl0, rl1 } proto carp keep state (no-sync)
```

Then, when writing firewall rules, keep in mind that, from `pf(4)`'s (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*) point of view, all traffic passes through the physical interface. Thus, in cases like:

```
pass in on $ext_if [...]
```

you can keep referring to the physical, not the virtual interface. On the other hand, the virtual address is associated to the CARP interface; thus, you need to refer to it if the firewall offers any services on its virtual address:

```
# SSH on the virtual interface
pass in on $int_if inet proto tcp from $int_if:network to
                    carp0 port ssh
```

or on a NATed server, through traffic redirection:

```
# Mail server accessible from the internet
rdr on $ext_if inet proto tcp from any to carp2 port
                    $mail_ports -> $mail_srv
```

In all other cases, CARP is perfectly transparent to `pf(4)` (*http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4*), as for services offered by the firewall on its physical addresses:

```
# SSH on the physical address
pass in on $int_if inet proto tcp from $int_if:network to
                    $int_if port ssh
```

or for normal filtering: see Listing 13.

As an example, let's see a basic PF ruleset for our external firewalls, Donald and Daisy: see Listing 14.

DANIELE MAZZOCCHIO

Latest version: http://www.kernel-panic.it/openbsd/pdc/

genioDATA

Creative Data Solutions

genioDATA Multiverse

Exactly the IT systems you need

genioDATA hosts every important operating system (NetBSD, FreeBSD, MacOS X Server) and many others as well (Linux, Solaris, Windows, HPUX) – on real hardware or virtualized.

All of them run in premier grade data centers (e.g. Level3).
This is what you get:
- clustered virtual machines,
- SAN-Storage (clustered or single),
- redundant routing and switching paths,
- assistance or complete management for your system (optional),
- several backup strategies including regional desaster recovery options.

Ask other providers if they can keep up:
- professional UNIX administrators with various certifications (RHCE, ACSA, ...),
- no long-term contracting (very low customer quit rate: no ties involved),
- individual setups for elaborated requirements,
- reliable platforms for those who do everything on their own,
- SLAs for up to 99,999 % availability (depending on the setup),
- very competitive prices.

Special discount:
10 % discount for readers of the BSD Magazine!

Get in contact:
- write an email to sales@geniodata.de,
- use our web contact form: www.geniodata.com/contact.

genioDATA | Wiesenweg 12 | 85567 Grafing | Germany
Data Centers in Munich, Nuremberg and hidden
www.geniodata.com | sales@geniodata.de

Easier WINE

Installation on amd64 FreeBSD

A short article on easier and faster method of installing WINE on an amd64 FreeBSD system.

What you will learn…

- You will learn how to install WINE on a amd64 FreeBSD system without needless (and long) compilation process.

What you should know…

- Installing and configuring a base OpenBSD system
- Configuring OpenBSD's Packet Filter

For those of You for whom WINE term does not ring a bell, here is a little quote from Wikipedia: *Wine is a free software application that aims to allow computer programs written for Microsoft Windows to run on Unix-like operating systems.' (...) Wine is not a full emulator, but is instead a compatibility layer, providing alternative implementations of the DLLs that Windows programs call, and a process to substitute for the Windows NT kernel.* [1]

Unfortunelly *FreeBSD Handbook* [2] in all its glory and usefullness does not contain information on how to add WINE to an amd64 system, even the WINE port in the Ports tree is only limited to i386 architecture. This may look not very optimistic for us, but there is other way of making it work on amd64 system, we will have to use whole i386 FreeBSD environment under `/compat/i386` directory to provide 32-bit libraries for WINE and 32-bit WINE itself.

Not so long ago official *FreeBSD Wiki* [3] started its activity and contain more and more useful entries, one of them is covering WINE on amd64 system, but it requires You to download all source of the FreeBSD base system, compile it and then fetch all sources for WINE and its dependencies and of course compile it. This of course takes a lot of time, even on fast machines with fast network connection and a lot of horsepower under the hood.

Down to Work

In this guide You will learn how to install WINE on amd64 system without compiling a single file, everything will be achieved using already built packages and FreeBSD base system data sets.

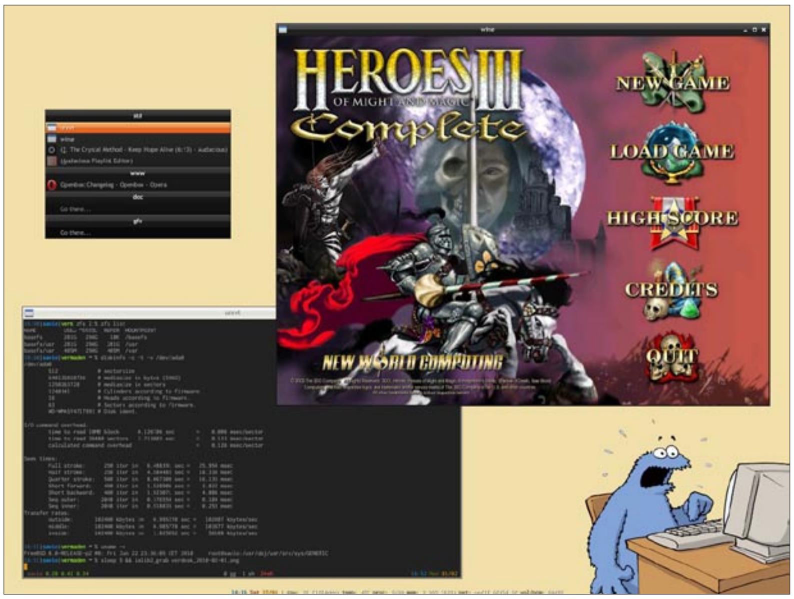


Figure 1. *Working WINE installation on amd64 FreeBSD system*

Listing 1. *Putting gevfspseudo filesystem*

```
# mount -t devfs devfs /compat/i386/dev
You will have to use these aliases to use WINE on amd64 system:
% alias winecfg32="LD_32_LIBRARY_PATH=/compat/i386/usr/local/lib \
                   PATH=/compat/i386/usr/local/bin:$PATH \
                   /compat/i386/usr/local/bin/winecfg"

% alias wine32="LD_32_LIBRARY_PATH=/compat/i386/usr/local/lib \
                PATH=/compat/i386/usr/local/bin:$PATH \
                /compat/i386/usr/local/bin/wine"
```

Below are the instructions to add both i386 complete environment under `/compat/i386` and WINE packages along with appreciate dependencies. First You will have to switch to `root` account.

```
% su -
Password: (will not be prompted)
```

If you do not have `wget` installed, then install it now, it will be needed later.

```
# pkg_add -r wget
```

Then set some variables that we will use later, like version of the installed FreeBSD system, where to get FreeBSD data sets and target installation directory.

```
# setenv UNAME $( uname -r | egrep -o ".*-[A-Z]+" )
# setenv URL ftp.freebsd.org/pub/FreeBSD/releases/i386/
                   ${UNAME}/base
# setenv DESTDIR /compat/i386
```

Now create appreciate destination directory, fetch needed FreeBSD datasets from the server and install them right away.

```
# mkdir -p ${DESTDIR}
# wget -r -c ftp://${URL}
# cd ${URL}
# chmod +x install.sh
# ./install.sh (answer 'y' here)
```

You will also need to copy your DNS information to the `/compat/i386` environment, just copy the `resolv.conf` file.

```
# cp /etc/resolv.conf ${DESTDIR}/etc/
```

This step involves using the `chroot(8)` mechanism, then setting appreciate variables and finaly, adding WINE packages with the dependencies.

```
# chroot ${DESTDIR} /bin/sh
> /etc/rc.d/ldconfig start
> export MACHINE=i386
> export UNAME_p=i386
> export UNAME_m=i386
> pkg_add -r wine
> exit
#
```

Done

If you are reading this text and performed all instructions provided above, You should have already working WINE on your amd64 system. I have tested this method with*Heroes]I[Complete* game and it works great. The devfspseudo filesystem is not needed for it, but in case some other game/application will require it, just mount it as shown below, or even put it permanently into your `/etc/fstab` file see Listing 1.

Now, to start some application with WINE You will type `% wine32` app.exe and to change preferences of WINE, `% winecfg32`. Of course You need to put these aliases into your shell startup files, so they will be 'remembered'.

References

- [1] *http://en.wikipedia.org/wiki/Wine_(software)*
- [2] *http://freebsd.org/handbook*
- [3] *http://wiki.freebsd.org/Wine*

SLAWOMIR WOJTCZAK (VERMADEN)

Configuring IP-Based SSL

on multiple hosts with Apache and FreeBSD

I have a very large PHP/MySQL application running on FreeBSD 6.0. Before we go any further let's get it out of the way: yes, I should upgrade to a more current version, but when you have a big system running a lot of sites in a live environment, well, you often end up a couple of versions behind the most recent stable release.

What you will learn…

- Before following the instructions a person would need to know basic shell commands and be familiar with virtual hosting entries in the Apache configuration file

What you should know…

- After reading the article the person will have a working knowledge of how to configure multiple SSL certificates using IP-based virtual hosting

There are currently about thirty sites on the system, all used by non-profit groups around the US. More recently I added on line ticket sales to the system for theatrical groups, and donations as well. This new feature meant, of course, that I needed secured processing of credit card information, so I implemented an SSL certificate to process those transactions.

The Bartell Theatre Company was the first to require SSL for their ticket sales so I installed an SSL certificate and added the following to the Apache configuration file see Listing 1.

This worked great for the Bartell, but then Mercury Players wanted to sell tickets too. So I replicated a VirtualHost entry for them and included appropriate references to their SSL certificate files. But when I tried to access the secured page to process a ticket sale I got a warning from the browser instructing me that the certificate in use was for the Bartell Theater's domain name.

I checked the Apache configuration again to make sure that correct file names were in the SSLCertificate... entries in the VirtualHost section of the file. Everything looked fine. I began to get a sinking feeling. Could it really be that I couldn't use multiple SSL certificates on the same Apache install? That made no sense. I was sure that of all the big installations out there running all kinds of e-commerce and secured connections this had to be possible.

I found the explanation, predictably, on the Apache web site itself:

Why can't I use SSL with name-based/non-IP-based virtual hosts?

The reason is very technical, and a somewhat *chicken and egg* problem. The SSL protocol layer stays below the HTTP protocol layer and encapsulates HTTP. When an SSL connection (HTTPS) is established `Apache/mod_ssl` has to negotiate the SSL protocol parameters with the client. For this, `mod_ssl` has to consult the configuration of the virtual server (for instance it has to look for the cipher suite, the server certificate, etc.). But in order to go to the correct virtual server Apache has to know the Host HTTP header field. To do this, the HTTP request header has to be read. This cannot be done before the SSL handshake is finished, but the information is needed in order to complete the SSL handshake phase. Bingo! (*http://httpd.apache.org/docs/2.0/ssl/ssl_faq.html#vhosts*)

The page went on to explain that a potential workaround was to use different IP addresses per SSL hosts. Needing this configuration to be up and working fast, and knowing this might be a little over my head as a developer who only plays system administrators on TV when asked (and I've never been asked), I enlisted the help of Bob Martin of BBM Hosting down in Texas, the only guy I know with a thick southern drawl who really knows his stuff about

FreeBSD and most all things *nix. Non-US natives might not know what I mean by that, but try to imagine, say, T. Boone Pickens talking competently about the details of web server performance or MySQL database configuration... but I digress.

Bob connected to the server and echo'd all the commands to a terminal on my laptop so I could follow through the steps to see exactly how he configured IP based SSL hosting.

Step 1

First, for each new IP address that will support a VirtualHost entry in the Apache configuration file, create an entry like the following, that accompanys see Listing 2.

...where the IP address in the snippet above is the public address of the web site to process the secured transactions. Port 443 ties the transactions to the secured SSL port.

Step 2

In `/etc/rc.conf` create an entry like this:

```
ifconfig_em0_alias0="inet 208.66.132.91 netmask
                    255.255.255.255"
```

Listing 1. *SSL and Apache Configuration file*

```
<VirtualHost *:443>
 DocumentRoot /usr/home/penguinsites/public_html
 ServerName bartelltheatre.org
 ServerAlias www.bartelltheatre.org
 SSLEngine on
 SSLCertificateFile /etc/ssl/penguinsites/bartell.crt
 SSLCertificateKeyFile /etc/ssl/penguinsites/bartell.key
 SSLCertificateChainFile /etc/ssl/penguinsites/gd_bundle.crt
...more directives here...
</VirtualHost>
```

Listing 2. *Configuration file. Creating the entry*

```
<VirtualHost 192.168.1.15:443>
 DocumentRoot /usr/home/penguinsites/public_html
 ServerName bartelltheatre.org
 ServerAlias www.bartelltheatre.org
 SSLEngine on
 SSLCertificateFile /etc/ssl/penguinsites/bartell.crt
 SSLCertificateKeyFile /etc/ssl/penguinsites/bartell.key
 SSLCertificateChainFile /etc/ssl/penguinsites/gd_bundle.crt
...more directives here...
</VirtualHost>
```

...where `ifconfig_em0` is the name of your NIC interface and `_alias0` is the unique fragment that identifies each consecutive alias for a new SSL certificate. The numeric part on the end of `_alias` must be consecutive following in increments of 1 for each new entry, as in the subsequent entries below.

```
ifconfig_em0_alias1="inet 208.66.132.92 netmask
                    255.255.255.255"
ifconfig_em0_alias2="inet 208.66.132.93 netmask
                    255.255.255.255"
ifconfig_em0_alias3="inet 208.66.132.94 netmask
                    255.255.255.255"
```

The two steps above will configure your server for multiple IP based SSL certificates, but the machine would need to be rebooted to activate them. To make each interface active immediately – following an Apache restart to reread its configuration – one additional step is needed. Make the interface live in the current environment by executing the following command as root in a shell.

```
ifconfig em0 inet ipaddress netmask 255.255.255.255 alias
```

...where the `0` in `em0` is the alias number, and must match the same incremental value in the rc.conf entries and `<ipaddress>` is the IP address of the new interface.

And that's it! You now have a simple, step by step guide to implementing IP based Virtual Hosting for multiple SSL certificates. I currently have six running on my server and hope to have about thirty by the end of the year.

SKIP EVANS

Skip has been programming since the first Radio Shack TRS-80's came off Noah's Ark and began repopulating the world with microcomputers after the Biblical flood. He's been a developer in industries including banking, accounting and telecommunications. He gambled away huge percentages of his bloated dot-com salary shooting pool in New York City in the late 1990s, eventually starting his own development shop in 2004. He nows resides in in Madison, Wisconsin, where he builds and maintains PenguinSites.com, a large multi-site hosting platform for managing on line ticket sales for theatrical groups.

BOB MARTIN

BBM Hosting, Diboll, Texas USA

BSD

File Sharing – Part 4. SSH

Last time I wrote on FTP and mentioned its security weakness, this time I intend to write on sharing and transfering files using its encrypted alternatives sftp, scp, and on fuse-sshfs.

What you will learn…

- how to configure and start OpenSSH server on various BSD's
- how to use various SSH based filesharing tools, namely: SCP, SFTP, FUSE-SSHFS

What you should know…

- have basic system admin skills such as „how to edit a file as root, how to add a user"
- be familiar with work in terminal

Although I test everything on OpenBSD and NetBSD and some things on FreeBSD personally, it all may change with versions and releases so be careful not to blindly copy and paste what I write.

Sharing files using SSH – a little bit of theory

SSH is neither a file system nor a file transfer protocol nor a sharing tool itself, it is an encrypted protocol, way of secure communication, used by transfering protocols and sharing tools, such as SCP or SFTP and the most interesting and inovative – file sharing system SSH file system (SSHFS) based on the FUSE framework. I am talking about the two transfer protocols and the SSH filesharing system in the same article since the server side is always the same. It is the sshd server. To be more precise I will speak about OpenSSH and the OpenSSHd from the OpenBSD project, that is the proper name of the free secure shell implementation that majority of *NIX systems are using nowadays, however for practical reason I will refer to it as SSH only.

SSH

In the beginning there was rlogin and telnet, ways how to reach a remote shell, that was for security reasons replaced with ssh, the reason was, that when you access a remote terminal with telnet, an attacker can access the packets with information on the way and get access to your login and password and to the data too. While when you login into the remote shell with OpenSSH the whole communication is encrypted and then sent so whoever may get access to your data on the way can very hardly read anything.

SCP and SFTP

To transfer files, rcp (remote copy from the BSD project) and ftp (file transfer protocol, also from Berkeley) were used and as we can see, the ftp still is widely used. However there are useful replacements for the two transfer protocols, called scp (secure copy) and sftp (secure ftp). For a regular user, especially when using a sofisticated GUI client, there is practicaly very little difference in using them two. Technically there is a difference, however. They use a different algorythm. Scp uses a faster transfer as it does not wait for confirmation of each package before sending another one which sftp does. However sftp is more reliable and more sofisticated on the contrary.

FUSE/SSHFS

Finally there is the most inovative project of SSH file system. Based on FUSE framework it provides an opportunity for every user to mount a remote filesystem on the local machine and use it as if it was a local file system. Nor special settings on server side, neither root permissons on the client side are needed – theoretically. As this is a rather new product not all BSD's are fully acquainted with this tool (if I may call it so), the most advanced in incorporating fuse into its system is FreeBSD which seems to have it included already in FreeBSD 6. I have never tried it on FreeBSD but I suppose it should work well by now, after 5 years of testing. In NetBSD project it found its way to the wip (testing pkgsrc part) two years ago, if I am not mistaken, and it works. I would not say flawlessly, as I tested it myself and I found some irritating issues, e.g. the terminal hangs after mounting a remote file, so I had to open another one to get to the files, so it probably still needs some tuning before it finds the way to the PKGSRC stable. And to the surprise of users there is complete absence of the sshfs in the OpenBSD project, I have tried to search even in the unofficial threads, but have not found anything.

Starting SSHD

Starting OpenBSD SSH server is easy. On FreeBSD, NetBSD and OpenBSD it is part of the basic system, so you just confirm the option in the rc script and here we go. Here follow the details:

FreeBSD

```
edit /etc/rc.conf:
sshd_enable="YES"
```

then you can specify the options by

```
sshd_flags="your_options"
```

to start the service from command line:

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

to restart the ssh service run:

```
# /etc/rc.d/sshd restart
```

NetBSD

```
edit /etc/rc.conf:
sshd="YES"
```

then you can specify the options by

```
ftpd_flags="your_options"
```

to start the service from command line:

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

to restart the ssh service run:

```
# /etc/rc.d/sshd restart
```

if you do not have the sshd option in rc.conf set to yes, the server will not start even manualy unless you specify *onestart* option

```
# /etc/rc.d/sshd  onestart
```

OpenBSD

edit /etc/rc.conf:

```
sshd_flags=""
```

and if you wish you can put down your options in the double quotes.

to start the service from command line:

```
# /usr/sbin/sshd &
```

to restart the ssh service run:

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

Configuring OpenSSH server

Configuring sshd is quite easy, all the configuration happens in one file and that is

```
/etc/ssh/sshd_conf
```

Among the most frequent options to alter belong login permissions and X permissions.

Unless you have a clear reason to do otherwise, X forwarding should be disabled by default, therefore:

```
X11Forwarding no
```

I would always recommend disabling root login:

```
PermitRootLogin no
```

If there are a lot of users using your machine, you may have a reason to allow only some of them distant login, after you use the following option, no one else but the chosen people will be able to login or share files through ssh:

```
AllowUsers mike goofy dirk
```

similarly you can allow a group of users by setting

```
AllowGroup wheel
```

or apply the opposit approach by banning only a few

```
DenyUsers daisy bunny
```

If you decide to use a passwordless login with the ssh key, you may need to specify that you allow it and where to look for the key, especially in case the remote system has different defaults:

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

For security reasons and attacks on port 22, some people prefer to change the port:

```
Port 5218
```

There are many other options, but in my seven years of admin practise I have never have to change them, besides a section I will consult in the *chroot* section below. But, of course, it all depends on what you are doing with the machine, so feel free to consult:

```
$ man sshd_config
```

Chrooted user SSH server

(the following example will work for OpenSSH)

In `/etc/ssh/sshd_config` set up the following options section:

```
Subsystem     sftp   internal-sftp
Match Group sftp
    ChrootDirectory %h
    ForceCommand internal-sftp
    AllowTcpForwarding no
```

For any users that you wish to chroot, add them to the sftp group by using:

```
# usermod -G sftp david
# usermod -s /bin/false david
# chown root:root /home/david
# chmod 0755 /home/david
```

Client side of SSH

The range of ssh clients is wide. I have chosen to talk more about the SCP, SFTP and SSHFS.

The command line syntax is easy and more or less the same with all BSD systems:

SCP

They syntax is very similar to the `cp` command. Just one easy selfexplanatory example: The following line will copy a directory nixcraft from robin's home to david's home preserving permissions and dates of the transfered directory and its contents. See that the robin user does not have to exist on the remote machine, since you can login as david to the 192.168.1.7 machine. The collon at the end of the line defines it is going to fall to the home of the loging-in user.

```
$ scp -rp  /home/robin/nixcraft david@192.168.1.7:
```

another example is moving robin's nixcraft directory to a different directory on a remote machine, logging in as robin himself:

```
$ scp -v /home/robin/nixcraft 10.0.0.3:/var/special/robins_stuff
```

SFTP

The syntax is almost the same like the `ftp` command syntax, here is an example where the user is logging in with his username to the `192.168.0.1` machine using a non standard port 5938 – the standard is port 22. In this example a file `gig.txt` from the user's home is uploaded.

```
$ sftp -o "Port=5938" 192.168.0.1
Connecting to 192.168.0.1...
petr@192.168.0.1's password:
$ sftp> ls
Documents
Download
Desktop
mydoc.txt
$ sftp> put gig.txt
Uploading gig.txt to /home/petr/gig.txt
gig.txt                     100%     0      0.0KB/s    00:00
$ sftp> exit
$
```

On the 'Net

- *http://nixcraft.com/all-about-freebsd-openbsd-netbsd/*
- *http://serverfault.com/questions/138362/openssh-sftp-chrooted-user-with-access-to-other-chrooted-users-files*
- *http://www.cyberciti.biz/faq/openbsd-restart-reload-openssh-without-rebooting/*
- *http://www.pubbs.net/200908/freebsd/34504-fusefs-sshfs.html*
- *http://fuse4bsd.creo.hu/*
- *http://www.coaly.com/support/howtos/41-unixhowtos/77-mountlocallyfreebsd.html* – OpenBSD, FreeBSD, NetBSD man pages

FUSE-SSHFS

Installation:

NetBSD (make sure you have *wip* section of pkgsrc in your pkgsrc tree)

```
# cd /usr/pkgsr/wip/fusefs-sshfs
# make install
```

(tested, works, but does not work comfortably)

FreeBSD

```
# cd /usr/ports/sysutils/fusefs-sshfs
# make install
# kldload /usr/local/modules/fuse.ko
# /usr/local/etc/rc.d/fusefs start
```

(not tested personally)

Accesssing remote system

`sshfs user@host:remotedir` mountpoint, so you run (on BSD it seems that you have to do that as root):

```
# sshfs john@192.168.0.1:/home/john /home/robin/johns_files
```

since that moment the files will become accessbile in the local directory `johns _ files` in robin's home. If you want to unmount the filesystem, you can do that by:

```
#   fusermount -u /home/robin/johns_files
```

Summary

SCP nad SFTP are very secure and reliable ways how to transfer files. SCP can be faster, SFTP can have more options, but they have similar system overheads and use SSH for encryption. SSHFS (FUSE) is a secure, simple, inovative and easy tool allowing users to mount remote filesystems using SSH protocol for encrypting the transfer, the advantage is that it does not require any special settings on the servers side (compared e.g. to NFS or SAMBA). Although OpenBSD project is the *father* of the OpenSSH protocol, SSHFS has not been implemented in OpenBSD (yet?). All the above tools are the client side of the process where OpenSSHD is the server side.

PETR TOPIARZ

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.

BSD Opinion

Reflections of a Sys-admin

A sentimental Rob Somerville reflects on his experiences with IT over the years and concludes that the BSD family is yet to be recognised as the shining example of good engineering and innovation we know it is.

First, a bit of history. When I was young lad, I assembled some scrap electronics – a few transistors, a transformer, some paper capacitors, an old HT coil from a car and a sturdy 12 volt power supply – and then build a Heath Robinson contraption that was capable of generating enough KV to generate a spark about 2 inches long. Not satisfied with this, I persuaded my father to stand on a rubber mat and hold on to the *hot* end while I placed a fluorescent tube on his (bald) head and it illuminated without damage to either the tube – or indeed – my father. Apart from demonstrating a natural bent towards technology and electronics, it taught me a valuable lesson. Often the best systems are based on true innovation and design and surprising results can be achieved by thinking *outside the box*.

The past 25 years have seen a major sea change in IT. We have moved away from the specialised arena of in-house analysts / programmers, innovators, coding and proprietary systems design towards the model that IT is more or less considered mass market, off the shelf, bolt on *blackboxes* and technical resource is considered external and disposable rather than an integral part of the organisation. In short, the shallow consumerist model (buy it, bust it, bin it) has now penetrated the corporate market, instead of ethos of investing in the long term by re-engineering, integrating or repairing. Ultimately, this leads to compromise, and as desktop software is mostly inherently fragile, people are shocked when systems fail, data quality is poor and the accusation is often made *Why can't my computer be as straightforward as my car ?* To summarise, sometimes the mass marketplace is not a good judge of quality, as the Betamax / VHS, Ipod / Mp3 player, Zippo/disposable lighter etc. arguments prove.

The 80's

Those were the days. DOS, Wordperfect, sneaker net (or Token Ring if you were lucky). Lots of bulletin boards on 1200 Baud connections if you knew where to look, but no *World Wide web*. Email was available to mainframe users and some BBS users. No viruses, trojans or spam. Hardware was manually configured with interrupt and DMA jumpers, and once it was up and running, systems would generally only break when the hardware or communications failed. Software was expensive, and required the explicit approval and comprehensive testing by IT departments or it was not commissioned. If you dropped a PC on your foot, you needed a new foot. The command line was king, and if you couldn't do it with a batch file, edlin, a hex editor or a basic compiler, you were in trouble. Diagnostic programs were extremely expensive (if available) and computer forensics were unheard of. One of my quick and dirty tests to see if an IBM XT had a hardware fault was to install DOS and Wordperfect and if it ran OK, it was a good bet that the PC iteslf was OK. Most of the time, unless there was a really nasty intermittent fault that was an excellent benchmark. I even remember people putting dust covers over their PC's when they went home for the night. If you worked in IT, you needed to know what you were doing or else there was going to be tears before bedtime. Systems were disparate, and if you wanted to link 2 branch offices together, weeks if not not months of planning were required before the project could be rolled out. The big players, IBM, Compaq, HP, Xerox and Apple were where the action was being played out. Microsoft was still in it's infancy, Xenix was the new kid on the block and BSD was on version 3.0. While IT was complicated and often unfriendly from the user perspective, it quickly became clear who knew their stuff and who didn't. The age of the guru had arrived, and end users were heavily dependent on the sysadmin for anything out of the ordinary. Few users had a PC at home.

The current scenario

There has been a total revolution not only with technology, but also attitudes towards IT. Everyone is an IT "expert" from the cleaner to the Chairman, and demands and expectations are high. As the user interface is now simple to use, many have fallen into the trap that IT itself is simple, and not a complex system. Systems security is both demanded and despised, as the spread of Internet technologies has allowed the unscrupulous to exploit the naive. PC's have become consumer items, and some households have 2 or processors. Hardware is much more reliable, and costs have fallen dramatically. A high specification PC that cost Ł10,000 to build in the 80's is a mere calculator in comparison to the Ł500 hardware bundle available in the high street today. Drop a PC on your foot today, and you will need a new PC. IT has become a commodity, with outsourcing and delegation to those who are interested (but sometimes blissfully unaware of the belly of the beast) and in-house IT departments have lost a lot of their kudos, sometimes relegated to box changers. Those who prefer the CLI to to pretty graphics are teased. Regularly, sub-standard products are marketed at a premium, written to the *bottom-line cost* instead of being designed with care, attention, extensibility and fitness for purpose. Those sysadmins who understand the complexity of systems pull their hair out in frustration as users demand – and get – low quality software and support that adds to the support burden and generates even more fire-fighting in the long term. Proprietary, closed source applications and operating systems *lock in* customers in a never ending spiral of bug, feature, upgrade and payment, rarely without other hidden costs or long term detriment. Salesmen in shiny suits persuade those vulnerable to a good pitch that this is good model, as *Nobody ever got fired for buying XYZ*. The desktop marketplace is dominated by one player, yet those who understand what lies behind the LCD display realise much of this is an illusion, as the underlying core systems – routers, storage, web-servers, databases, firewalls – are far more disparate and resilient than most desktops will ever be. IT – at least as far as the desktop is concerned – has lost its gravitas and become another victim of the consumer society where form is more important than function, style more important than substance.

The next phase

So where does the BSD family fit into all of this? I believe *BSD and Open Source has been quietly gathering strength in the background, and with the current desire to cut costs even further, eventually there will be a *flight to quality* and a return to long-term strategy and engineering, rather than the current consumerism and fire-fighting. The BSD licence, is one of the most legally liberal documents in existence, will free any organisation with a desire to be rid of red-tape when it comes to software development and regulation. Those that wish to maintain the status quo will argue that *BSD (or whatever) will not run current desktop applications, but they, like the big players in the 80's have missed the point – software is moving back on to the server again (and indeed the web) with the rise of the internet, virtualisation, cloud computing and thin-client technologies. The drive for this will eventually cut the desktop down to size, rationalise the GUI even further and once again the focus will be on reliability, security, redundancy and flexibility – all key features of the BSD family. Install *BSD and you have everything available to you on one DVD – Compilers, Diagnostics, Applications – and this is attractive to the genuine power user, where a *workstation* is preferable to a glorified word processor. The browser will become preeminent, and a move away from *proprietary* to *standardised* will raise the importance of the original strong RFC standards. The front end will become more *thin* and the server will become more *dedicated*, and indeed the PC or hardware interface (in whatever form it eventually evolves into) will become as standardised and reliable as the car. Look at mobile phones – how often do they crash? The move will be away from *generalist* to *specialist* as organisations seek to leverage IT to minimise costs and increase profits and once again back room IT will play a pivotal role in the way the organisation innovates.

To conclude

It is somewhat ironic that the X-windows GUI we are all so familiar with was alive and well in 1984 before Bill Gates launched Microsoft Windows 1.0 in 1985. This innovative thread can be carried through much of the *BSD family, the TCP/IP stack, the filesystem, the kernel etc. In hundreds of installations of Open Source and *BSD installations for testing or production I can count on the fingers of one hand installations that were *difficult* due to hardware incompatibilities. On the fingers of the other hand, I can count on the number of times the O/S or applications have crashed – more often than not due to flaky hardware (and once due to a badly compiled set of libraries). I have an elderly FreeBSD server here that is not on a UPS – and we have frequent power failures here. Only once in 7 years have I had to run FSCK after a crash. If I was using other O/S's – running on unfiltered power would be madness. True, it can be argued that server software is in a different league from the desktop, but little confidence can be gained when a supplier who is cognisant of design flaws exploits this loophole for profit. As a colleague so wryly stated, *Don't knock XYZ, it's what keeps me in a job*. Personally, I would rather make myself redundant in that area, and use my talents to innovate for the organisation, to help them gain the edge over their competitors and share the technological *Wow factor*. I believe the *BSD family – both on the desktop and especially the server has now matured to the point that it can no longer be ignored. The question is do we keep the secret to ourselves or shout it from the rooftops?

ROB SOMMERVILL

Rob Somerville has been passionately involved with technology both as an amateur and professional since childhood. A passionate convert to *BSD, he stubbornly refuses to shave off his beard under any circumstances. Fortunately, his wife understands him (she was working as a System/36 operator when they first met). The technological passions of their daughter and numerous pets are still to be revealed.

11th Libre Software Meeting

Free Admission

Bordeaux - Pessac - Talence

July 6th - 11th, 2010

http://rmll.info

Tuesday 6 to Friday 9: Enseirb-Matmeca and Université Bordeaux 1
Saturday 10 and Sunday 11: Central Bordeaux
And also lots of events all around Bordeaux metropolitan area and Aquitaine

Accessibility - Sustainable development
Administration and public policies
Free art and culture - Science and education - Technology - Corporate
ARDOUR

Manifestation Responsable — Gironde Conseil General

Live on... Radio RMLL
http://radio.rmll.info

SAP

over BSD

As I did describe in the introduction BSD in the Industry last month, several applications are necessary to support all the Industry areas despite of traditional IT services and can be a big step in the BSD implementation into this world.

One of the most important applications are ERP, Enterprise Resource Planning and PLM, Product Life Cycle Management and CRM Customer Relationship Management because support the finances, human resources and all those activities related economically with the projects. Even when there are few ERP projects under free software like Compiere, Open Bravo, OpenERP, webERP, etc, that can be founded in Source Forge and Free Software Foundation, most of them must be highly customized by the end users and the infrastructure support is not very good.

Best of ERP under privative licences is SAP. This Company together with Oracle is covering almost all big companies in all kind of markets, banks and so on. In my case, as Industrial Engineering Company and in order to offer a real Turn Key project to our Clients, you cannot offer a technological package with a price of around 20 millions of Euros and at the time; offer them an ERP with not a clear infrastructure, with bugs, etc.

As per the above sentence SAP is the best choice, so now we must know how to improve SAP with our BSD. To do this, let's describe basically what is the infrastructure offered by SAP:

mySAP ERP is an ERP solution designed for midsize to large companies and focuses on optimizing internal business processes and collaboration/ integration across the extended Enterprise. mySAP ERP is a complete, separately orderable ERP solution that can be converted into a mySAP Business Suite family of business solutions license at any time to include the full suite of software solutions covering mySAP Customer Relationship-Management (mySAP CRM), mySAP Supply Chain Management (mySAP SCM), mySAP Product Lifecycle Management (mySAP PLM), and mySAP Supplier Relationship Management (mySAP SRM). For customers licensing the mySAP Business Suite, the mySAP ERP solution is included.

mySAP ERP comprises:

- SAP R/3 Enterprise (default until end of ramp-up of SAP ERP central component)
- SAP ERP central component
- SAP Business Information Warehouse (as part of SAP NetWeaver)
- SAP Enterprise Portal (as part of SAP NetWeaver)
- SAP Exchange Infrastructure (as part of SAP NetWeaver)
- SAP Supplier Relationship Management (self-service procurement, classic scenario only)
- SAP Strategic Enterprise Management
- SAP E-Recruiting
- SAP Learning Solution
- SAP Financial Supply Chain Management
- SAP Employee Self-Service and SAP Manager Self-Service
- Collaborative Projects (cProjects)
- SAP Internet Sales R/3 Edition (Web application component)

The mySAP All-in-One solution is designed for smaller companies. Based on SAP Business Suite, mySAP All-in-One runs on a single database and offers a preconfigured system with various scenarios incorporating, for example, additional functionality in the areas of customer relationship management and supply chain management.

The SAP Business One solution is PC software for small companies requiring core finance and logistic functions, or in the case of subsidiaries, seamless integration of business software with the parent company's IT system.

Is not intended to do an extensive description of the SAP infrastructure in this article, mainly because in SAP webpage *http://www.sap.com/index.epx* is available all literature about his products. I want just to introduce the core of the infrastructure to start to know the possibilities of this product.

To do this we will be focused in one of the systems, SAP Business One (Figure 1).

Accounting and Financials

SAP Business One helps you manage your general ledger, journals, budgets, and accounts payable and receivable.

You can conduct all your banking activities including processing payments by check, cash, credit card, bank transfer, and bill of exchange as well as reconcile various accounts and create financial reports for profit and loss, cash flow, balance sheet, and aging. You can also update account postings at the exact time relevant business events occur.

Sales and Customers

With SAP Business One, you can:

Create quotes, enter orders, and provide better customer service, track sales opportunities and activities from first contact to the close of sale.

Initiate marketing campaigns by using templates for mass e-mails, provide support for customer service, service contracts, and warranties. The application also lets you manage and maintain customer contacts with full Microsoft Outlook synchronization, which results in increased sales effectiveness and stronger customer relationships.

Purchasing and Operations

Every small business needs a systematic approach to managing the procurement process, from creating purchase orders to paying vendors.

SAP Business One helps manage the complete order to: pay cycle, including receipts, invoices, and returns. You can also plan material requirements for production, control bills of materials, and replenish inventory automatically.

And using integration with Crystal Reports, you can analyze your vendors' performance and adjust your procurement strategy accordingly.

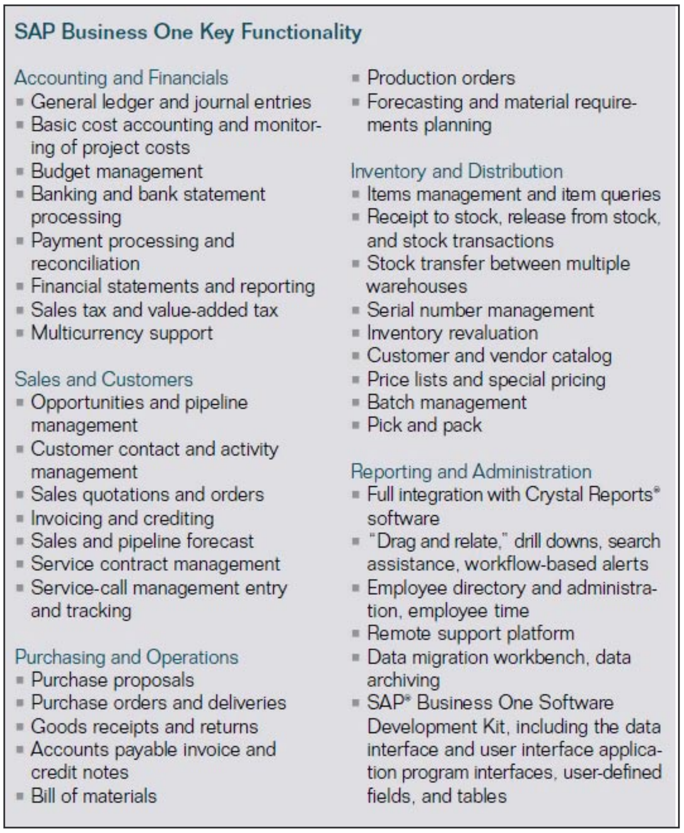
Inventory and Distribution

SAP Business One also lets you readily manage your inventory and operations, including picking, packing, delivery, and billing.

You can perform inventory valuation using different methods such as standard costing, moving average, and FIFO; monitor stock levels; and track transfers in real time and across multiple warehouses. In addition, you can run real-time inventory updates, availability checks, and manage pricing and special pricing, which allows you to automatically apply volume, cash, and account discounts to transactions with vendors and customers.

Reporting and Administration

SAP Business One provides powerful, integrated analytic and reporting tools to help you access the critical business information you need. With SAP Business One,



SAP Business One Key Functionality

Accounting and Financials
- General ledger and journal entries
- Basic cost accounting and monitoring of project costs
- Budget management
- Banking and bank statement processing
- Payment processing and reconciliation
- Financial statements and reporting
- Sales tax and value-added tax
- Multicurrency support

Sales and Customers
- Opportunities and pipeline management
- Customer contact and activity management
- Sales quotations and orders
- Invoicing and crediting
- Sales and pipeline forecast
- Service contract management
- Service-call management entry and tracking

Purchasing and Operations
- Purchase proposals
- Purchase orders and deliveries
- Goods receipts and returns
- Accounts payable invoice and credit notes
- Bill of materials
- Production orders
- Forecasting and material requirements planning

Inventory and Distribution
- Items management and item queries
- Receipt to stock, release from stock, and stock transactions
- Stock transfer between multiple warehouses
- Serial number management
- Inventory revaluation
- Customer and vendor catalog
- Price lists and special pricing
- Batch management
- Pick and pack

Reporting and Administration
- Full integration with Crystal Reports® software
- "Drag and relate," drill downs, search assistance, workflow-based alerts
- Employee directory and administration, employee time
- Remote support platform
- Data migration workbench, data archiving
- SAP® Business One Software Development Kit, including the data interface and user interface application program interfaces, user-defined fields, and tables

Figure 1. *SAP Bussiness One Key Funcionality*

together with fully integrated Crystal Reports, you can gather data from multiple sources and generate timely and accurate reports based on critical company data across financials, sales, customers, inventory, service, production, and operations. Focused on data security, Crystal Reports allows you to choose from a variety of report formats and to control access to information displayed.

You can also use additional functionalities that are an integral part of SAP Business One such as *drag and relate* and interactive drilling down through multiple levels of relevant data to get complete information instantly.

System Reliability and Performance

SAP offers a remote support platform for SAP Business One to help you maintain your software system more easily and proactively prevent potential issues from impacting your business activities. As an automated monitoring tool, this remote support platform helps identify system bottlenecks by enabling SAP support services to collect information on your system status and check the system against known support issues. By sending regular status e-mails and automatic fixes, it allows you to avoid issues from happening and decrease the time you would otherwise spend on IT support. In addition, the tool provides several other services including automated database backups, pre upgrade evaluations, inventory valuation checkups, and system installation health checks.

Is difficult to goes deep in such big software solution but the above description can show an overview of quantity and quality of services that SAP infrastructure can offer.

If we take a look to the SAP origins, they always were linked to privative software but from the last years SAP start to open the doors to others solutions that even when are not free, at least are UNIX based. The choice of SAP involves to be supported by a solid, reliable and well supported OS and this is one of the reasons because they did start to offer also support for OS like Novell SuSE Linux Enterprise Server or Red Hat. In these OS can be founded the critical aspects to consider. Both are solid and reliable and at time offer a really good technical support. The decision from SAP to goes in this direction, open the doors to OS like BSD that offer same benefits that GNU/Linux OS to going into.

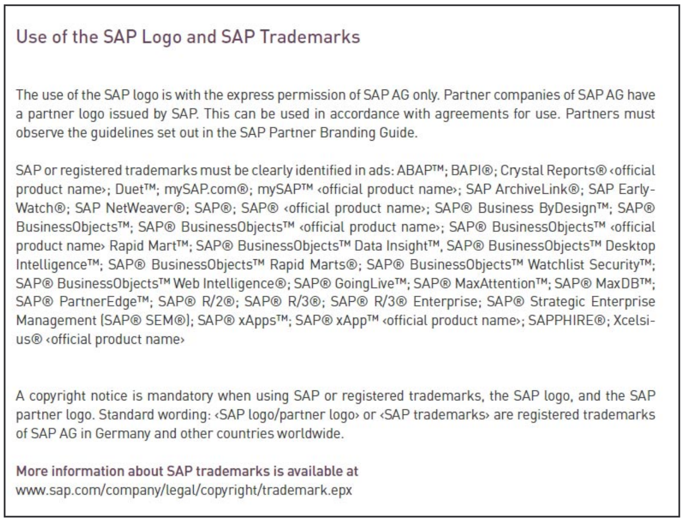
So let's see what are the advantages that SAP watch in SuSE/Novell to know what we can improve to be able to offer a competitive platform over BSD.

One of my favourites is the virtualization. Is clear that when a Industrial Engineering Company decide to offer SAP implementation to his Clients, must have an servers infrastructure to support services like backups online and all services that can meaning a added value to our Clients and at the same time, be an economical benefit to us. Actually the virtualization help you to improve this services in such way that you do not want to have a huge IT infrastructure but a well organized. Also virtualization gives you possibility to improve the reliability of your Services, and offer to your clients a real 24/7 service.

At the same time, you will have a group of physical servers that trough virtualization allows you to implement all the necessary SAP and IT infrastructure despite of your physical infrastructure and get the different Clients data apart ones to the others without need different physical servers to all your Clients.

BSD flavours have all the aspects that do it a good choice to run with SAP, UNIX features like reliability, solid like a rock, fast, high availability ...By other side, OpenBSD could be a perfect choice to banking systems because his proven security.

I guess that the Companies that offer BSD implementations must start to be focus in improve online technical services in order to get in and also persist in a standardization of BSD and by sure early we will enjoy SAP running over any BSD flavours.



Use of the SAP Logo and SAP Trademarks

The use of the SAP logo is with the express permission of SAP AG only. Partner companies of SAP AG have a partner logo issued by SAP. This can be used in accordance with agreements for use. Partners must observe the guidelines set out in the SAP Partner Branding Guide.

SAP or registered trademarks must be clearly identified in ads: ABAP™; BAPI®; Crystal Reports® ‹official product name›; Duet™; mySAP.com®; mySAP™ ‹official product name›; SAP ArchiveLink®; SAP Early-Watch®; SAP NetWeaver®; SAP®; SAP® ‹official product name›; SAP® Business ByDesign™; SAP® BusinessObjects™; SAP® BusinessObjects™ ‹official product name›; SAP® BusinessObjects™ ‹official product name› Rapid Mart™; SAP® BusinessObjects™ Data Insight™, SAP® BusinessObjects™ Desktop Intelligence™; SAP® BusinessObjects™ Rapid Marts®; SAP® BusinessObjects™ Watchlist Security™; SAP® BusinessObjects™ Web Intelligence®; SAP® GoingLive™; SAP® MaxAttention™; SAP® MaxDB™; SAP® PartnerEdge™; SAP® R/2®; SAP® R/3®; SAP® R/3® Enterprise; SAP® Strategic Enterprise Management (SAP® SEM®); SAP® xApps™; SAP® xApp™ ‹official product name›; SAPPHIRE®; Xcelsius® ‹official product name›

A copyright notice is mandatory when using SAP or registered trademarks, the SAP logo, and the SAP partner logo. Standard wording: ‹SAP logo/partner logo› or ‹SAP trademarks› are registered trademarks of SAP AG in Germany and other countries worldwide.

More information about SAP trademarks is available at
www.sap.com/company/legal/copyright/trademark.epx

Figure 2. *SAP Trademarks*

JOSEBA MENDEZ

meetBSD
2010 Conference

Starting from May 1st, 2010 users and developers of UNIX systems (in particular the BSD family) can register for the 7th edition of the meetBSD Conference. This time the event will take place in a beautiful city of Krakow, Poland, on July 2nd and 3rd, 2010 (Friday and Saturday). Similar to previous editions, conference speakers will include people actively involved in BSD development, with international audience and guests, including representatives from companies cooperating with the Open Source community. During this year's edition it will be possible for participants to take the BSD Certification Group exam and obtain the well recognized and valued certificate.

meetBSD is an annual event dedicated to UNIX-based operating systems, especially from the BSD family. It also supports projects related to the BSD systems, people behind them and Open Source communities. meetBSD is a strictly technical event with focus on high quality content, with opportunities for both seasoned professionals and those only planning to learn the BSD way. The truly open world of BSD is waiting for you.

Besides the conference proper, meetBSD has always been a great social event, full of attractions and surprises. Hurry up and register for the conference yet today, as the registration deadline is approaching shortly. See you all in Krakow, July 2-3!

meetBSD 2010 Conference
July 2nd and 3rd 2010, Kraków, Poland

www.meetBSD.org

organizers:

@outsourceme SEMIHALF EMBEDDED SYSTEMS

MAGAZINE BSD

In the next issue:

-My personal BSD
-Ten Years (and More) of FreeBSD
- and Other !

Next issue is coming in July!

NOW FEATURING THE INTEL® XEON® PROCESSOR 5600 SERIES

Orion II iX-N4236

Powerful 4U Orion II Storage Series

- ✓ Outstanding Performance
- ✓ Excellent Cooling Efficiency
- ✓ Up to 24 Processing Cores with Hyper-Threading
- ✓ Up to 72TB in 4U, Unparalleled Storage Density
- ✓ Up to 432TB in 20U of Rack Space Utilizing Optional Orion II JBOD Expansion Units

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

Notable features include:

- Dual Intel® 64-Bit Socket 1366 Six-Core, Quad-Core, or Dual-Core, Intel® Xeon® Processor 5600/5500 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 192GB 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.

The Intel® Xeon® Processor 5600 Series (Six/Quad-Core) and Intel® Xeon® Processor 5500 Series (Quad/Dual-Core) have a light energy 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 to 192GB 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.