

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

VPN AND BSD

INSIDE

CLOSED-SOURCE AND UNSUPPORTED DRIVERS WITH FREEBSD

BIULDING VPNS ON OPENBSD

COMMISSIONING FREEBSD

WITH THE DRUPAL CONTENT MANAGEMENT FRAMEWORK – PART 1

I.T. CERTIFICATIONS AND THE VALUE I GOT IN IT...

VOL3 NO.10
ISSUE 10/2010(16)
1898-9144



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



✓ Increased Performance ✓ Impressive Energy Savings

iX-Triton TwinBlade Servers: The Easy-to-Manage, Greener Way to Serve

› AFFORDABLE › ECONOMICAL › SAVINGS



The new Triton TwinBlade Server is the most technologically advanced blade server system in the industry, and the ideal solution for power-efficiency, density, and ease of management.

The Triton TwinBlade Server supports up to 120 DP servers with 240 Intel® Xeon® 5600/5500 series processors per 42U rack, achieving an unmatched 0.35U per DP node. Up to two 4x QDR (40 Gbps) Infiniband switches, 10GbE switches or pass-through modules give the TwinBlade the bandwidth to support the most demanding applications.

With N+1 redundant, high efficiency (94%) 2500W power supplies, the TwinBlade is the Greenest, most energy-efficient blade server in the industry. The

energy saved by the iX-Triton TwinBlade Server will keep the environment cleaner and greener, while leaving the green in your bank account.

Server management is also simple with the Triton Twin Blade Server.

Remote access is available through SOL (Serial Over Lan), KVM, and KVM over IP technologies. A separate controller processor allows all of the Triton's remote management and monitoring to function regardless of system failures, offering true Lights Out Management.

Using the Triton's management system, administrators can remotely control TwinBlades, power supplies, cooling fans, and networking switches. Users may control the power remotely to reboot and reset the Triton TwinBlade Center and individual Twin Blades, and may also monitor temperatures, power status, fan speeds, and voltage.

For more information on the **iX-Triton TwinBlade**, or to request a quote, visit:

<http://www.ixsystems.com/tritontwinblade>

20 Server Compute Nodes in 7U of Rack Space

The iX-TB4X2 chassis holds 10 TwinBlade servers and each TwinBlade supports two nodes. This gives the iX-TB4X2 chassis the ability to house 20 nodes in 7U of rack space. The powerful Triton TwinBlade achieves 0.35U per dual-processor node, and is twice as dense as the previous generation of dual-processor blades.

A fully-loaded iX-Triton TwinBlade supports 40 Intel® Xeon® 5600/5500 series processors and up to 2.5 TB DDR 1333/1066/800MHz ECC Registered DIMM memory. In a 42U rack this translates into 120 nodes with 240 Intel® Xeon® 5600/5500 series processors and 15 TB DDR 1333/1066/800MHz ECC Registered DIMM memory.



- ▶ By replacing 1U servers with TwinBlade servers, the power savings of the iX-TB4X2 can reach more than \$1000* per year, per server with reduced cooling costs added in.



- ▶ Replacing 1U rackmount servers with an iX-TB4X2 Twin Blade can reduce carbon dioxide emissions by over 5.5 metric tons.**



- ▶ The iX-Triton TwinBlade delivers the most energy-efficient blade server in the industry with four N+1 redundant, high efficiency (94%) 2500W power supplies.

* Electricity costs vary by location.

** According to Energy Information Agency (a statistical agency of the U.S. Department of Energy), saving one kilowatt hour of electricity reduces carbon dioxide emissions by 1.43 pounds.



Call iXsystems toll free or visit our website today!
+1-800-820-BSDi | www.iXsystems.com

Key features:

- Up to 10 dual-node TwinBlades in a 7U Chassis, 6 Chassis per 42U rack
- Remotely manage and monitor TwinBlades, power supplies, cooling fans, and networking switches
- Hardware Health Monitor
- Virtual Media Over Lan (Virtual USB, Floppy/CD, and Drive Redirection)
- Integrated IPMI 2.0 w/ remote KVM over LAN/IP
- Remote Power Control
- Supports one hot-plug management module providing remote KVM and IPMI 2.0 functionalities
- Up to four N+1 redundant, hot-swap 2500W power supplies
- Up to 16 cooling fans

Each of the TwinBlade's two nodes features:

- Intel® Xeon® processor 5600/5500 series, with QPI up to 6.4 GT/s
- Intel® 5500 Chipset
- Up to 128GB DDR3 1333/ 1066/ 800MHz ECC Registered DIMM / 32GB Unbuffered DIMM
- Intel® 82576 Dual-Port Gigabit Ethernet
- 2 x 2.5" Hot-Plug SATA Drive Trays
- Integrated Matrox G200eW Graphics
- Mellanox ConnectX QDR InfiniBand 40Gbps or 10GbE support (Optional)



Dear Readers!

Get make yourself comfortable and open this issue. October has welcomed us with the cold weather and nothing reminds us about summer.

You will find inside some advice about Building VPNs written by Daniele. Joshua as always shares with us his thought in Let's Talk.

Rob introuces the first part of his article, to teach you how to perform a bare metal installation of FreeBSD with networking enabled.

Hope you find the articles interesting and useful.

We want to remind you about answering short questionnaires concerning our magazine.

This will certainly help us to improve our magazine!

Thank you and enjoy your reading!

Thank you!

Olga Kartseva
Editor in Chief

olga.kartseva@software.com.pl



MAGAZINE BSD

Editor in Chief:

Olga Kartseva
olga.kartseva@software.com.pl

Contributing:

Rob Somerville, Daniele Mazzocchio, Rashid N. Achilov, Joseba Mendez, Laura Michaels
Lukas Holt, Caryn Holt, Laura Michaels

Special thanks to:

Marko Milenovic, Worth Bishop and Mike Bybee

Art Director:

Ireneusz Pogroszewski

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.lesińska@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

All trade marks presented in the magazine were used only for informative purposes. All rights to trade marks presented in the magazine are reserved by the companies which own them.

The editors use automatic DTP system **AOPUS**

Mathematical formulas created by Design Science MathType™.

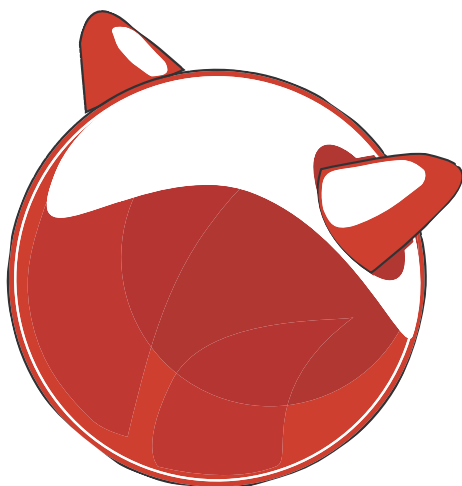
GET STARTED

06 Commissioning FreeBSD with the Drupal Content Management Framework – Part 1

ROB SOMMERVILLE

With nearly 6000 modules and PHP support Drupal offers a sophisticated web development platform as well as a thriving community.

Drupal, originally conceived by Dries Buytaert, has a reputation of being an extremely capable DContent Management System (CMS) albeit with a steep learning curve. While many criticisms concerning the complexity of the interface will be addressed in the forthcoming Drupal 7 release (which is currently in the alpha stage), Drupal 6 excels in stability, flexibility and high quality code. The developers also subscribe to a transparent policy towards security issues, and have a dedicated security team which ensures that core modules remain high quality. Used as the basis of many high profile sites.



HOW TO'S

12 Building VPNs on OpenBSD

Daniele Mazzocchio

A VPN is a network made up of multiple private networks situated at different locations, linked together using secure tunnels over a public (insecure) network, typically the Internet.

VPNs are becoming increasingly popular, as they allow companies to join the LANs of their branches or subsidiaries into a single private network (site-to-site VPNs) or to provide mobile employees, such as sales people, access to their corporate network from outside the premises (remote-access VPNs), thus making accessing and sharing internal information much easier.

34 Closed-source and unsupported drivers with FreeBSD

Anton Borisov

Sooner or later you come to a conclusion that you need to have an enhanced mobility throughout your home place. And you decide to purchase a Wi-Fi card and put it into a home gate-keeper. Do you know about troubles that could bring this simple transaction like WiFi network card purchase? Some might ask – is it necessary to buy a WiFi-card instead of a simple AccessPoint (AP)? At first glance you can figure out that there exist the fine models of ADSL-modems with wireless capabilities and that could work as AP. However, it should be noticed that: a) not all home connections to an Internet-provider go through a „copper” like phone- or cable-line; b) you simply need to add a WiFi-capability to an already working gate; c) a WiFi-card itself costs several times cheaper of AP.



LETS TALK

40 I.T. certifications and the value I got in it

Joshua Ebarvia

Joshua shares his experience with our readers, this time about certifications.

Commissioning FreeBSD

with the Drupal Content Management Framework – Part 1

With nearly 6000 modules and PHP support Drupal offers a sophisticated web development platform as well as a thriving community.

What you will learn...

- How to patch, upgrade and install ports, initially configure Apache, PHP, MySQL and Drupal

What you should know...

- How to perform a bare metal installation of FreeBSD with networking enabled etc.

Drupal, originally conceived by Dries Buytaert, has a reputation of being an extremely capable *Content Management System* (CMS) albeit with

a steep learning curve. While many criticisms concerning the complexity of the interface will be addressed in the forthcoming Drupal 7 release (which is currently in the

Listing 1. Extract from rc.conf file

```
hostname="drupal.merville.intranet"
```

Listing 2. Extract from hosts file

```
192.168.0.117 drupal.merville.intranet drupal
```

Listing 3. Patching FreeBSD to the latest revision

```
pkg_add -r portaudit portupgrade
```

```
freebsd-update fetch
freebsd-update install
```

```
portaudit -Fda
portsnap fetch
portsnap extract
pkgdb -F
portupgrade -avbPR --batch
```

Listing 4. PHP pre-compile set-up

```
cd /usr/ports/lang/php5
make config
cd /usr/ports/lang/php5-extensions
make config
```

Listing 5. Installing the ports

```
cd /usr/ports/databases/mysql55-server
make install BATCH=YES
cd /usr/ports/www/apache22
make install BATCH=YES
cd /usr/ports/lang/php5
make install BATCH=YES
cd /usr/ports/lang/php5-extensions
make install BATCH=YES
```

Listing 6. Ensure the PHP module is present in httpd.conf

```
LoadModule php5_module libexec/apache22/libphp5.so
```

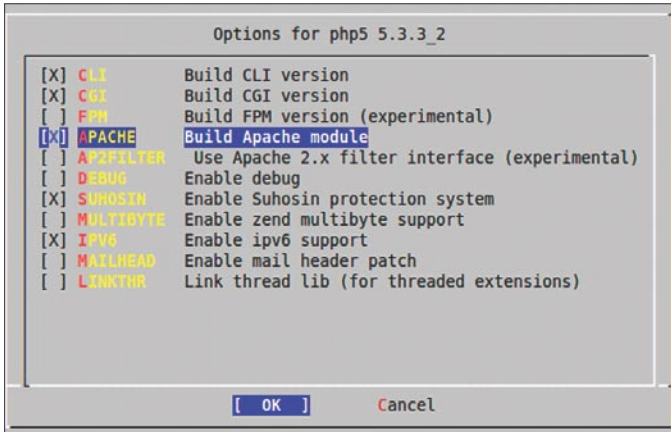



Figure 1. Ensure the Apache module is enabled

alpha stage), Drupal 6 excels in stability, flexibility and high quality code. The developers also subscribe to a transparent policy towards security issues, and have a dedicated security team which ensures that core modules remain high quality. Used as the basis of many high profile sites (Table 2).

Requirements

Drupal requires Apache/MySQL/PHP and may be configured to run in a virtual host environment. In this Howto, we will install Drupal as a stand-alone server. This demo was prepared using Virtualbox 3.28 hosting FreeBSD 8.1 with 1GB RAM and 20GB storage.

Stage 1 – Install FreeBSD

Proceed with a bare metal install of FreeBSD 8.1, and configure user accounts, networking etc. so that the install can download ports from the FreeBSD website. To minimise server bloat, I performed a minimal install without the ports tree etc. which took only a few minutes.

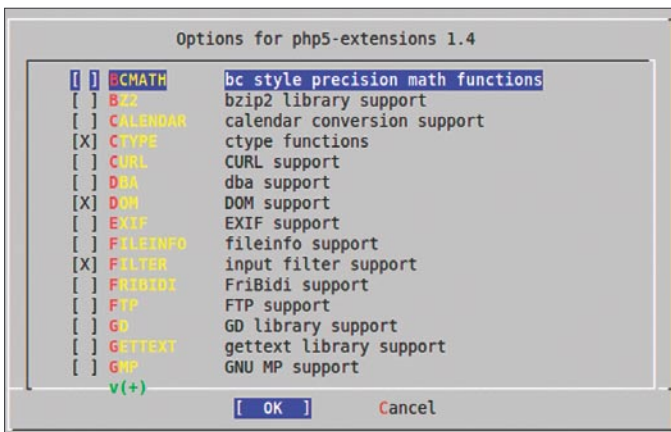


Figure 2. Enable PHP support as required

Listing 7. Setting up the php.ini file

```
cp /usr/local/etc/php.ini-production /usr/local/etc/
    php.ini
```

Listing 8. drupal.conf

```
#
# Apache configuration file for Drupal6
#

DocumentRoot "/usr/local/www/drupal6/"
<Directory "/usr/local/www/drupal6">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

<IfModule dir_module>
    DirectoryIndex index.php
</IfModule>

ErrorLog "/var/log/drupal.log"
```

Listing 9. Drupal log file

```
# Added for PHP support

application/x-httpd-php                php
application/x-httpd-phps                phps
```

Listing 10. Drupal log file

```
touch /var/log/drupal.log
```

Listing 11. MySQL config file

```
cp /usr/local/share/mysql/my-medium.cnf /var/db/mysql/
    my.cnf
```

Listing 12. Securing the root MySQL password

```
/usr/local/etc/rc.d/mysql-server onestart
mysqladmin password
```

Listing 13. Create user

```
mysqladmin -u root -p create drupal6
```

Listing 14. Login to MySQL

```
mysql -u root -p
```

Listing 15. SQL to create Drupal database and user login

```
GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX,
ALTER, LOCK TABLES, CREATE TEMPORARY TABLES ON 'drupal6'.
* TO 'drupal'@'localhost' IDENTIFIED BY '!lgH87i-LL34';
```

Listing 16. Installing Drupal and supporting modules

```
cd /usr/ports/www/drupal6
make install BATCH=YES
cd /usr/ports/www/drupal6-advanced_help
make install BATCH=YES
cd /usr/ports/www/drupal6-cck
make install BATCH=YES
cd /usr/ports/www/drupal6-chaos
make install BATCH=YES
cd /usr/ports/www/drupal6-ckeditor
make install BATCH=YES
cd /usr/ports/www/drupal6-image
make install BATCH=YES
cd /usr/ports/www/drupal6-imce
make install BATCH=YES
cd /usr/ports/www/drupal6-menu_block
make install BATCH=YES
cd /usr/ports/www/drupal6-nodewords
make install BATCH=YES
cd /usr/ports/www/drupal6-page_title
make install BATCH=YES
cd /usr/ports/www/drupal6-panels
make install BATCH=YES
cd /usr/ports/www/drupal6-path_redirect
make install BATCH=YES
cd /usr/ports/www/drupal6-pathauto
make install BATCH=YES
cd /usr/ports/www/drupal6-print
make install BATCH=YES
cd /usr/ports/www/drupal6-seo_checklist
make install BATCH=YES
cd /usr/ports/www/drupal6-views
make install BATCH=YES
cd /usr/ports/www/drupal6-webform
make install BATCH=YES
cd /usr/ports/www/drupal6-wysiwyg
make install BATCH=YES
cd /usr/ports/www/drupal6-zeropoint
make install BATCH=YES
```

Listing 17. Copying the Drupal settings file across

```
cd /usr/local/www/drupal6/sites/default
cp default.settings.php-dist ./settings.php
chown www:www settings.php
```

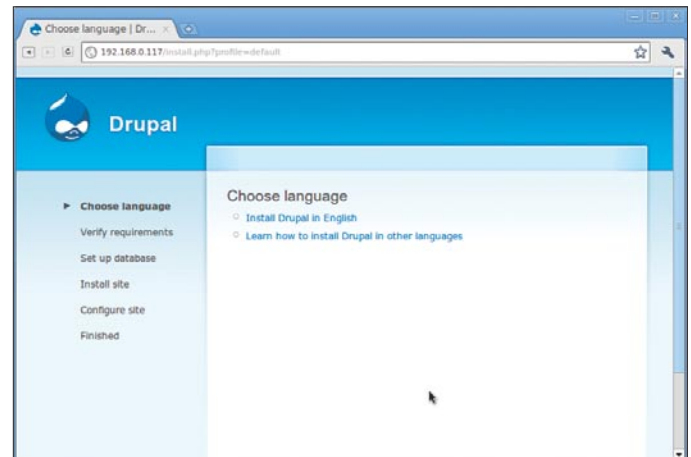
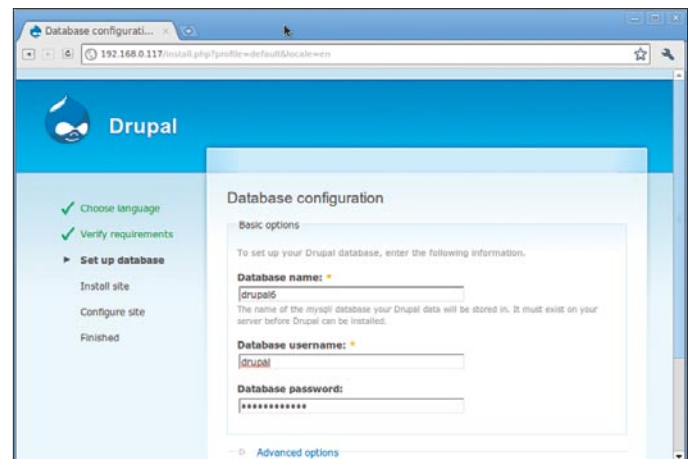
Listing 18. Starting Apache

```
/usr/local/etc/rc.d/apache22 onestart
```

Stage 2**– Post install configuration of FreeBSD, install the latest ports tree and AMP stack**

First of all, ensure that `/etc/rc.conf` and `/etc/hosts` have a valid hostname and IP address respectively, otherwise Apache will not start. Replace with parameters that match your network (Listing 1/2).

As we will require PHP library support for Apache, the AMP stack etc. will have to be installed from ports rather

**Figure 3. Drupal up and running ready for install****Figure 4. Use the settings in Listing 13/14/15**

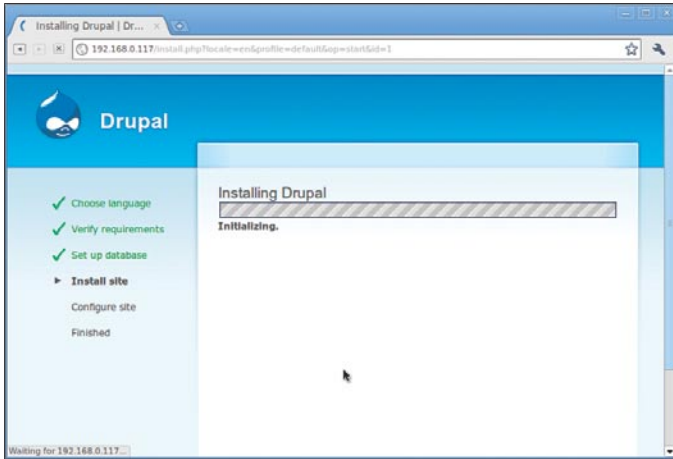


Figure 5. Drupal installing



Figure 7. Installation complete

than packages. Best practice in a production environment is to ensure the server is patched to the latest revision, so we will install `portaudit` (which checks for known vulnerabilities) and `portupgrade` which upgrades the ports to the latest version. The binary `freebsd-update` applies security updates to the base system, while `portsnap` will pull the latest version of the ports tree onto our server. As a precaution, `pkgdb` will be used to check pack age registry database prior to upgrade.

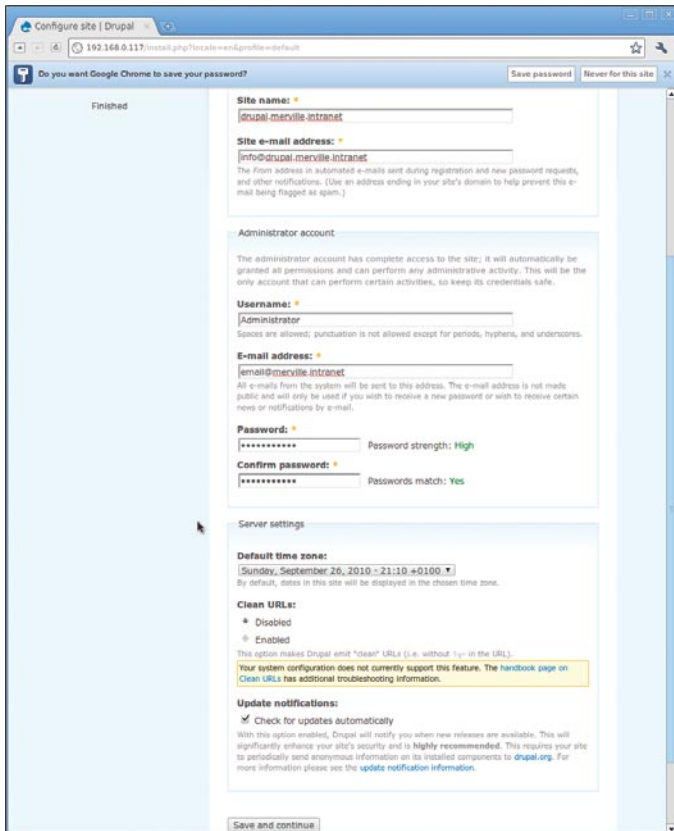


Figure 6. Drupal login and email settings

NOTE

Running `portsnap`, `pkgdb` and `portupgrade` on an existing FreeBSD installation should be done with caution as the ports tree will be updated and may have unforeseen implications – see the man pages and the FreeBSD website for further details and caveats etc.

Depending on your bandwidth and server specification, the entire upgrade may take some time so if you prefer to perform the install without patching the box, the ports tree can either be installed from the FreeBSD DVD / ISO during Stage 1, or the commands `portsnap fetch` and `portsnap extract` can be used to fetch and extract the latest tree which takes a few minutes (see Listing 3).

Now that we have the ports tree installed and updated, we can proceed to install the AMP stack. Prior to the compilation of PHP, we need to ensure that the Apache module is enabled and any additional PHP extensions are installed as required, e.g. `curl` or `bz2` (Listing 4 and Figure 1/2):

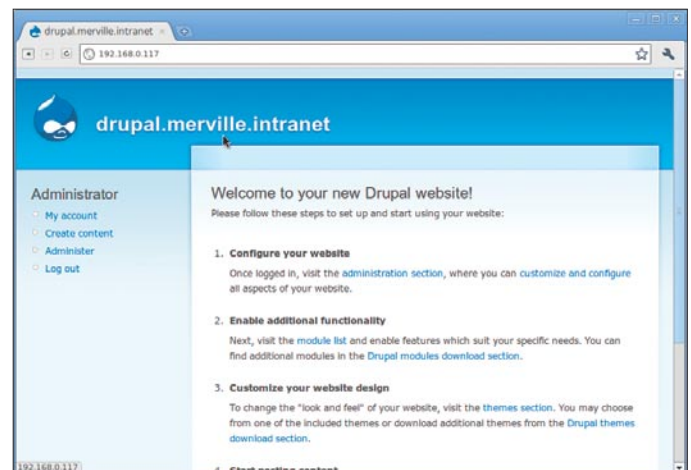


Figure 8. Site up and running

Listing 19. Add these settings to `rc.conf`

```
# Added for Drupal support

sendmail_enable="NONE"
apache22_enable="YES"
mysql_enable="YES"
```

Using `make config` can be repeated for MySQL and Apache as required, but for the Drupal install to proceed the only essential change is support for the Apache module (Figure 1).

Now we need to download the ports and install (Listing 5). Using the `BATCH=YES` switch means we can leave the server to perform an unattended install with the default values if additional ports etc. are downloaded. If fine control of the installation is required, this may be omitted but further intervention will be needed during the installation of various libraries etc. to fine tune any settings.

Stage 3 – Configure Apache, PHP and MySQL

Check that the following line is in DSO Support section of `/usr/local/etc/apache22/httpd.conf` (Listing 6). Copy the `php.ini-production` file to `php.ini` (Listing 7).

Create an Apache configuration file for Drupal `/usr/local/etc/apache22/Includes/drupal.conf` and add the following (Listing 8). Add PHP support to `/usr/local/etc/apache22/mime.types` (Listing 9). Create the error log file for Drupal (Listing 10). Copy the skeleton `my.cnf-medium` file to `/var/db/mysql/my.cnf` (Listing 11). Start MySQL and secure the MySQL root password: (Listing 12). Create the MySQL database `drupal6` (Listing 13).

Table 1. Files modified during install

Check list of files modified
<code>/usr/etc/hosts</code>
<code>/usr/etc/rc.conf</code>
<code>/usr/local/etc/apache22/httpd.conf</code>
<code>/usr/local/etc/apache22/mime.types</code>
<code>/usr/local/etc/apache22/Includes/drupal.conf</code>
<code>/usr/local/etc/php.ini</code>
<code>/var/db/mysql/my.cnf</code>
<code>/usr/local/www/drupal6/sites/default/settings.php</code>

Table 2. Some high-profile Drupal websites

Drupal Websites	
UK government national data	<code>data.gov.uk</code>
The Economist	<code>economist.com</code>
The Mayor of London	<code>london.gov.uk</code>
MTV UK	<code>mtv.co.uk</code>
Sony Music	<code>musicbox.sonybmg.com</code>
The New York State Senate	<code>nysenate.gov</code>
The New Republic	<code>tnr.com</code>
Ubuntu Linux	<code>ubuntu.com</code>
The World Food Program	<code>wfp.org</code>
The US Whitehouse	<code>whitehouse.gov</code>

Set the privileges and drupal MySQL password to `!lgH87i-LL34` for database `drupal6` (Listing 14/15).

Stage 4 – Install and configure Drupal

Install Drupal and supporting modules (Listing 16). Copy the Drupal settings file across (Listing 17). Start Apache (Listing 18).

Install Drupal via web interface – point your browser at the IP address set in hosts in Listing 2 (Figure 3/4/5/6/7).

Add settings to `rc.conf` so daemons start on reboot (Listing 19)

Next article

In Part 2, we will look at setting up templates, adding content and further configuring extending the site functionality. Now is a good time to secure / fine tune the configuration further and get to know the Drupal 6 interface.

ROB SOMERVILLE

*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.*

Creative Data Solutions and Hosting

genioDATA

Data Security

Replicate your databases in high class data centers.

Have an email archive run by **genioDATA** that leaves nothing more to wish for.

Copy your files to several sites to plan for disaster recovery.

Get Your Own World

Operating Systems are the worlds in the IT universe. Get yours:

FreeBSD, NetBSD starting at € 25

CentOS, OpenSuSE starting at € 25

RHEL, SLES starting at € 42

MacOS X Server starting at € 67

Windows Server starting at € 42

Got an idea? Make it live.

In a **genioDATA** Server.

Individual Appliances

Need an ERP environment

(enterprise resource planning)?

Have to operate a web(services)

cluster with 99,999 % availability?

Need an email environment where

not one email gets lost?

genioDATA engineers it.

genioDATA runs it.

You use it.

*genio***DATA**

info@sccon.de | www.geniodata.com/bsdi.html | +49(0)8092 862568

Building VPNs

on OpenBSD

A VPN is a network made up of multiple private networks situated at different locations, linked together using secure tunnels over a public (insecure) network, typically the Internet.

What you will learn...

- A good knowledge of OpenBSD administration

What you should know...

- How to build building VPNs on OpenBSD

Traffic inside VPN tunnels is usually encrypted and authenticated to provide security equivalent to that provided by leased lines, but at a fraction the cost.

A *tunnel* is created by encapsulating a network protocol (e.g. IP) within another network protocol, operating at the same layer of the OSI model (e.g. IP, ICMP) or at a higher layer (e.g. ESP, TLS).

VPNs are becoming increasingly popular, as they allow companies to join the LANs of their branches or subsidiaries into a single private network (*site-to-site VPNs*) or to provide mobile employees, such as sales people, access to their corporate network from outside the premises (*remote-access VPNs*), thus making accessing and sharing internal information much easier.

Though most often associated with Ipsec (<http://www.kernel-panic.it/openbsd/vpn/vpn2.html>), VPNs are a rather broad concept and can be implemented using a number of different tunneling protocols (L2TP, MPLS, PPTP, TLS, among others). In particular, in this document, we will take a look at the three most popular VPN implementations supported by OpenBSD:

IPsec

<http://www.kernel-panic.it/openbsd/vpn/vpn2.html> – a suite of standard protocols, defined in various RFCs (see Appendix), that operate at the network layer of the OSI model; OpenBSD (<http://www.openbsd.org/>) natively

supports IPsec protocols and provides specific tools and daemons to manage IPsec VPNs;

OpenVPN

<http://www.openvpn.net/> – an SSL-based VPN solution, operating at the application layer and probably the strongest contender for IPsec, thanks to its robustness, ease of use and portability;

OpenSSH

<http://www.openssh.org/> – since release 4.3, OpenSSH supports the tunneling of *arbitrary network packets over a connection between an OpenSSH client and server, as a true VPN* (see [OBSD39] <http://www.openbsd.com/39.html>).

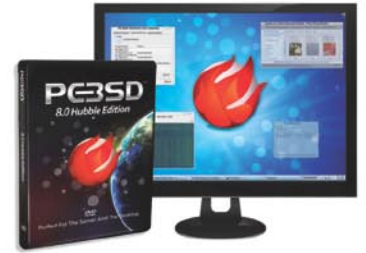
Besides the inherent differences in cryptographic algorithms and authentication mechanisms, these three VPN implementations differ under several aspects; each one has its own advantages and drawbacks and the choice among them must consider not only the ease of installation and administration, but also factors like bandwidth, reliability and scalability. The following are the most relevant differences:

- IPsec runs in kernel space, tightly integrated with the host TCP/IP stack, while OpenVPN and OpenSSH are user-space daemons. The in-kernel architecture



FreeBSD
Mall

Your FreeBSD &
PC-BSD Resource
www.FreeBSDMall.com



FreeBSD 8.1 Jewel Case CD/DVD

Set contains:

- **Disc 1:** Installation Boot (i386)
- **Disc 2:** LiveFS (i386)
- **Disc 3:** Essential Packages (i386)
- **Disc 4:** Essential Packages (i386)

FreeBSD 8.1 CD	\$39.95
FreeBSD 8.1 DVD	\$39.95
FreeBSD 7.3 CDROM	\$39.95
FreeBSD 7.3 DVD	\$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD!

FreeBSD Subscription, start with CD 8.1	\$29.95
FreeBSD Subscription, start with DVD 8.1	\$29.95
FreeBSD Subscription, CD 7.3	\$29.95
FreeBSD Subscription, DVD 7.3	\$29.95

PC-BSD 8 DVD (Hubble Edition)

PC-BSD 8 DVD	\$29.95
PC-BSD Subscription	\$19.95

BSD Magazine

BSD Magazine	\$11.99
BSD Magazine Subscription	\$11.99

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide)	\$39.95
The FreeBSD Handbook, Volume 2 (Admin Guide)	\$39.95
★ Special: The FreeBSD Handbook, Volume 2 (Both Volumes)	\$59.95
★ Special: The FreeBSD Handbook, Both Volumes, & FreeBSD 8.1	\$79.95

The FreeBSD Bundle

Inside the Bundle, you'll find:

- FreeBSD Handbook, 3rd Edition, Users Guide
- FreeBSD Handbook, 3rd Edition, Admin Guide
- FreeBSD 8.1 4-disc set
- FreeBSD Toolkit DVD

★ Special: The FreeBSD CD Bundle	\$89.95
★ Special: The FreeBSD DVD Bundle	\$89.95

The FreeBSD Toolkit DVD

FreeBSD Mousepad

FreeBSD Caps

PC-BSD Caps

For **MORE** FreeBSD & PC-BSD items, visit our website at [FreeBSDMall.com!](http://FreeBSDMall.com)

CALL 925.240.6652 Ask about our software bundles!

t-shirts
\$18 - \$21.99



has the advantage of being faster and more efficient, but may increase the impact of possible vulnerabilities and programming errors on the whole system;

- OpenVPN and OpenSSH have a slightly higher overhead due to the encapsulation of the payload within higher layers of the OSI model;
- IPsec works at the network layer of the OSI model, while both OpenVPN and OpenSSH can operate in either bridging mode (layer 2) or routing mode (layer 3) (please refer to [OVPN-FAQ] <http://www.openvpn.net/index.php/open-source/faq.html#bridge2> for a brief overview of bridging vs. routing); to tunnel ethernet traffic over IPsec, you need the additional layer of tunneling provided by the `gif(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=gif&sektion=4>) interface;
- IPsec interoperability comes from its being a standard, but different vendors' implementations may not be entirely compatible; the interoperability of OpenVPN and OpenSSH, instead, is ensured by their high portability across the most popular OSes.

Despite the many differences, OpenVPN has some common ground with IPsec, since, as stated in [OVPN-SEC] (<http://www.openvpn.net/index.php/open-source/faq.html#security-issues>), OpenVPN's security model is heavily based on the IPsec ESP protocol for secure tunnel transport over UDP.

This document assumes that you are familiar with OpenBSD, since it won't cover topics like base system configuration, packages/ports installation or Packet Filter syntax.

IPsec overview

IPsec configuration on OpenBSD is a pretty easy and straightforward process, especially compared to most other implementations; nevertheless, IPsec is a rather complicated beast and a good working knowledge of its protocols and internals is essential to configure it and get it to

work properly. Therefore, before beginning the configuration, let's take a brief tour of the IPsec protocols and features.

IPsec (IP security) is a suite of standard protocols designed to provide interoperable, high quality, cryptographically-based security [RFC4301] (<http://tools.ietf.org/html/rfc4301>) for protecting communications over IPv4 and IPv6 networks. The main security services offered by IPsec are:

- *Confidentiality* – traffic is encrypted to ensure that only the legitimate receiver is able to access the data transmitted.
- *Connectionless integrity* – ensures that no modifications were made to the data while in transit across the network.
- *Data origin authentication* – the receiver is able to verify that data actually originates from the claimed source.
- *Detection and rejection of replays* – duplicate IP datagrams are detected and processed only once.

These security services are provided at the IP layer (layer 3 of the OSI model), thus protecting all protocols that may be carried over IP, including IP itself.

IPsec protocols

Most of IPsec security services are provided using two traffic security protocols:

- *AH (Authentication Header)* – defined in [RFC4302] (<http://tools.ietf.org/html/rfc4302>), AH is used to provide connectionless integrity, data origin authentication and optional (at the discretion of the receiver) anti-replay protection for IP datagrams.
- *ESP (Encapsulating Security Payload)* – defined in [RFC4303] (<http://tools.ietf.org/html/rfc4302>), ESP offers the same set of services as AH (data origin authentication, connectionless integrity and anti-replay), plus confidentiality.

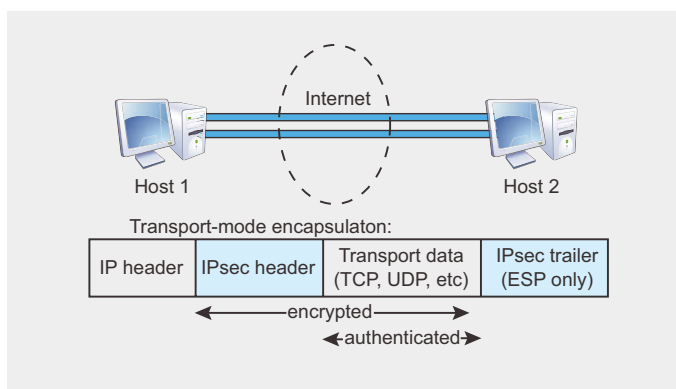


Figure 1. ESP and AH – transport mode

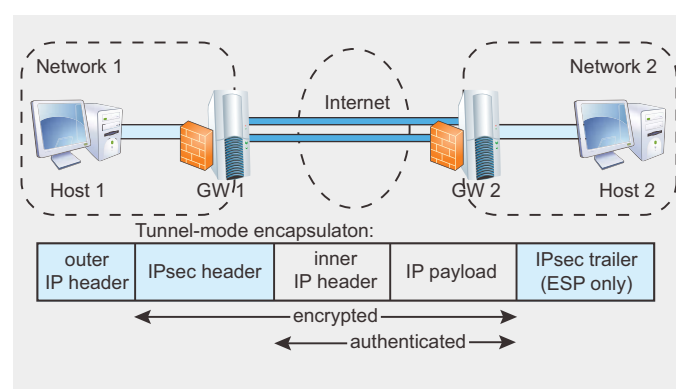


Figure 2. Basic network topology of the VPN

ESP is by far the most popular of the two protocols, since it provides confidentiality by encrypting network traffic, thus protecting transmitted data from passive attacks. On the other hand, AH provides stronger authentication than ESP as it protects part of the outer IP header as well as the next level protocol data, while ESP only protects the inner (encapsulated) IP header; however, this feature, in addition to not being of great use in most cases, also violates the modularization of the protocol stack (see [SCHNEIER] <http://www.schneier.com/paper-ipsec.pdf>, where the AH protocol is proposed for complete elimination).

AH and ESP may also be applied in combination with each other to exploit the strengths of both protocols but, in most real-world scenarios, ESP alone is enough.

Both ESP and AH support two modes of operation:

- transport mode – IPsec protects only the payload of the IP packet (usually the transport layer data, hence its name), leaving the IP header, and thus routing, unchanged; transport mode can be used only for host-to-host communication; (see Figure 1)
- tunnel mode – the entire IP packet is encrypted and/or authenticated and then encapsulated into a new IP packet; tunnel mode is typically used to connect either two remote networks or a host and a network; it is more flexible than transport mode, but imposes more bandwidth overhead; (see Figure 2)

The flexibility of tunnel mode allows it to fully supersede the functionality of transport mode, at the reasonable expense of a slightly higher bandwidth overhead. As a consequence, transport mode is rarely used in real-world VPNs and, just like AH, [SCHNEIER] (<http://www.schneier.com/paper-ipsec.pdf>) suggests that transport mode be eliminated altogether, with the advantage of significantly reducing IPsec complexity.

In a nutshell, while ESP and tunnel mode are by far the most prevalent choice, AH and transport mode can be considered the black sheeps of the IPsec protocol family!

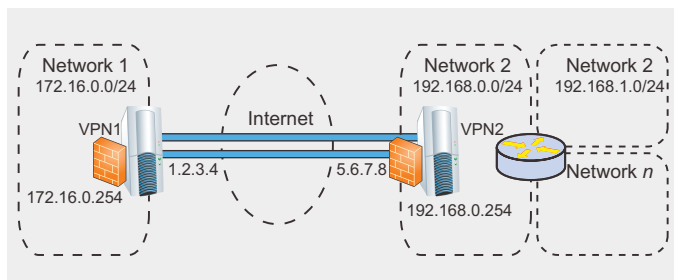


Figure 3. ESP and AH – tunnel mode

SA, SPI, SPD and other acronyms

To actually establish the VPN, the IPsec protocols require that some state data be shared between the VPN endpoints, such as the cryptographic algorithms for encryption and authentication, the keys used as input to the cryptographic algorithms, the current sequence number, the antireplay window and so on.

These data are held in a data structure called a *Security Association (SA)*; SAs are created by a specific protocol, IKEv2 (defined in [RFC4306] <http://tools.ietf.org/html/rfc4306>), which also has the responsibility of mutually authenticating the two communicating parties, setting up the encrypted channel for secure information exchange (these steps are part of the so-called *IKE phase 1*) and negotiating the shared secret from which cryptographic keys are derived (*IKE phase 2*).

A Security Association applies to a single protocol (AH or ESP) and to a single direction of traffic flow; therefore, *to secure typical, bi-directional communication between two IPsec-enabled systems, a pair of SAs (one in each direction) is required. IKE explicitly creates SA pairs in recognition of this common usage requirement [RFC4301] (<http://tools.ietf.org/html/rfc4301#section-4.1>).*

SAs are collected in a *Security Association Database (SAD)*, where they are uniquely identified by the combination of protocol (AH or ESP), destination address and an arbitrary 32-bit value called the *Security Parameter Index (SPI)*. The SPI has the specific task of helping the receiver to identify the SA under which an incoming packet should be processed.

But how does IPsec decide which datagrams to send through the VPN and which not? For instance, in a typical site-to-site VPN scenario, the IPsec gateway will usually tunnel and/or protect only traffic between the remote

Listing 1. Adding the variables to the `/etc/sysctl.conf`

```

/etc/sysctl.conf
[ ... ]
net.inet.esp.enable=1      # Enable the ESP IPsec
                           protocol
net.inet.ah.enable=1      # Enable the AH IPsec
                           protocol
net.inet.ip.forwarding=1  # Enable IP forwarding for
                           the host. Set it to '2' to
                           # forward only IPsec
                           traffic
net.inet.ipcomp.enable=1  # Optional: compress IP
                           datagrams
    
```

LANs, leaving all other traffic unaffected. Well, IPsec makes such decisions based on *policies*, i.e. user-defined rules stating which packets should be protected using IPsec security services, which should be allowed to bypass IPsec protection and which should be discarded. IPsec policies are applied based on some specific fields in the datagram headers, called *selectors*, which include: source and destination addresses, Next Layer Protocol, source and destination ports (if used by the next layer protocol).

As with Security Associations, IPsec policies are held in a database, called the *Security Policy Database (SPD)*, which must be consulted during the processing of all traffic (inbound and outbound), including traffic not protected by IPsec, that traverses the IPsec boundary.

The life of an IPsec packet

To recap, let's have a look at what the (brief) life of an IPsec packet looks like; we will consider the most common case: an ESP tunnel-mode VPN between two remote networks (see picture above). The story begins when the first gateway (GW1) receives an outbound packet from a host (Host1) within its internal network and destined for a host (Host2) on the remote network:

- the gateway first compares the datagram's selector fields against the SPD to find the first matching policy;
- the policy may specify one of three possible processing choices:
 - DISCARD, the packet is not allowed to traverse the IPsec boundary and is dropped;
 - BYPASS, the packet is allowed to cross the IPsec boundary without IPsec protection and will be routed normally;
 - PROTECT, the packet must be afforded IPsec protection and the policy will point to zero or more SAs in the SAD;
- in the present case, the gateway has a policy specifying that the datagram must be encapsulated with tunnel-mode ESP and sent to GW2;
- if no SA exists for this policy, IKE will be invoked to negotiate the SAs with the appropriate peer;
- the first matching SA(s) will be applied, providing the requested security services to the datagram;
- the IP datagram will be encapsulated in ESP and the outer IP header will have the addresses of GW1 and GW2 as source and destination addresses respectively;

After a brief walk around the Internet, the encapsulated packet hits the second gateway (GW2):

- the datagram is checked to see whether it contains an IPsec header; if not, the datagram is forwarded normally;
- using the destination address, the SPI and the type of IPsec header of the incoming datagram, the gateway determines which SA to use; if no matching SA is found, the packet is dropped;
- if antireplay is activated, the sequence number is checked for validity;
- the packet is decrypted and/or authenticated as specified by the SA;
- the gateway locates the SPD entry that applies to the datagram based on its selectors and verifies that the SA(s) applied in the previous steps match with SA(s) specified by the policy;
- the packet is decapsulated and forwarded to next hop or to the appropriate transport protocol.

Listing 2. *The first step in setting up the PKI is the creation of the root CA certificate and private key on the signing machine using openssl*

```
CA# openssl req -x509 -days 365 -newkey rsa:1024 \
> -keyout /etc/ssl/private/ca.key \
> -out /etc/ssl/ca.crt
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to '/etc/ssl/private/ca.key'
Enter PEM pass phrase: <passphrase>
Verifying - Enter PEM pass phrase: <passphrase>
-----
You are about to be asked to enter information that
will be incorporated
into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN.
There are quite a few fields but you can leave some
blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) []: Kernel Panic Inc.
Organizational Unit Name (eg, section) []: IPsec
Common Name (eg, fully qualified host name) []:
CA.kernel-panic.it
Email Address []: danix@kernel-panic.it
CA#
```

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



BSD

BSDMAGAZINE

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

www.bsdmag.org

Ipsec on OpenBSD

Now that we have an adequate working knowledge of the IPsec architecture and protocols, we are finally ready to move from theory to practice and start having some fun with OpenBSD! OpenBSD ships by default with full IPsec support in the stock kernel and provides a set of user-space daemons and tools for managing IPsec configuration, dynamic key exchange and high availability; and the great thing is that, as you'll see, setting up an IPsec VPN on OpenBSD is an incredibly simple and fast task, especially compared to most other IPsec implementations out there.

But before proceeding to edit configuration files and run system commands, let's take a brief look at the basic network topology of the VPN that we are going to set up in this document; it's a very simple site-to-site VPN, with a couple of multi-homed security gateways (VPN1 and VPN2) linking two remote private networks (172.16.0.0/24 and 192.168.0.0/24) see Figure 3.

In this chapter, we will set up the VPN using IPsec: to be more precise, we will configure it in tunnel mode (the only

option for network-to-network VPNs) and use the ESP protocol in order to encrypt the VPN traffic as it traverses the Internet; we will also consider the case of redundant IPsec gateways with `carp(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4>). Then, in the next chapters, we will see how the same VPN can be implemented using alternative solutions, in particular OpenVPN and OpenSSH.

Preliminary steps

Before proceeding to configure IPsec, we have to perform a few preliminary steps to make sure the systems are correctly set up for IPsec to work properly. The IPsec protocols are enabled or disabled in the OS's TCP/IP stack via two `sysctl(3)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=3>) variables: `net.inet.esp.enable` and `net.inet.ah.enable`, both enabled by default; you can check this by running the `sysctl(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=8>) command:

```
# sysctl net.inet.esp.enable
net.inet.esp.enable=1
# sysctl net.inet.ah.enable
net.inet.ah.enable=1
```

Since our VPN gateways will have to perform traffic routing, we also need to enable IP forwarding, which is turned off by default. This is done, again, with `sysctl(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl&sektion=8>), by setting the value of the `net.inet.ip.forwarding` variable to `1` if you want any kind of traffic to be forwarded or `2` if you want to restrict forwarding to only IPsec-processed traffic:

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

Optionally, you may also want to enable the IP Payload Compression Protocol (IPComp) to reduce the size of IP datagrams for higher VPN throughput; however, bear in mind that the reduction of bandwidth usage comes at the expense of a higher computational overhead (see [RFC3173] <http://tools.ietf.org/html/rfc3173> for further details):

```
# sysctl net.inet.ipcomp.enable=1
net.inet.ipcomp.enable: 0 -> 1
```

To make these settings permanent across reboots, add the following variables to the `/etc/sysctl.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sysctl.conf&sektion=5>) file: see Listing 1.

Listing 3. The creation of a Certificate Signing Request (CSR) on each of the IKE peers

```
VPN1# openssl req -new -key /etc/isakmpd/private/
      local.key \
> -out /etc/isakmpd/private/1.2.3.4.csr
You are about to be asked to enter information that
      will be incorporated
into your certificate request.
What you are about to enter is what is called a
      Distinguished Name or a DN.
There are quite a few fields but you can leave some
      blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) []: IT
State or Province Name (full name) []: Italy
Locality Name (eg, city) []: Milan
Organization Name (eg, company) []: Kernel Panic Inc.
Organizational Unit Name (eg, section) []: IPsec
Common Name (eg, fully qualified host name) []: 1.2.3.4
Email Address []: danix@kernel-panic.it

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []: <enter>
An optional company name []: <enter>
VPN1#
```

Finally, we need to bring up the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) virtual network interface. This interface allows you to inspect outgoing IPsec traffic before it is encapsulated and incoming IPsec traffic after it is decapsulated; this is primarily useful for filtering IPsec traffic with PF and for debugging purposes.

```
# ifconfig enc0 up
```

To make the system automatically bring up the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) interface at boot, create the `/etc/hostname.enc0` (<http://www.openbsd.org/cgi-bin/man.cgi?query=hostname.if&sektion=5>) configuration file:

```
/etc/hostname.enc0
up
```

Setting up the PKI

OpenBSD's IKE key management daemon, `isakmpd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=isakmpd&apropos=0&sektion=8>), relies on public key certificates for authentication and therefore requires that you first set up a *Public Key Infrastructure* (PKI) for managing digital certificates.

The first step in setting up the PKI is the creation of the root CA certificate (`/etc/ssl/ca.crt`) and private key (`/etc/ssl/private/ca.key`) on the signing machine (which doesn't have to be necessarily one of the VPN gateways) using `openssl(1)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=openssl&sektion=1>); e.g.: see Listing 2.

The next step is the creation of a *Certificate Signing Request* (CSR) on each of the IKE peers; for instance, the following command will generate the CSR (`/etc/isakmpd/private/1.2.3.4.csr`) for the VPN1 machine (the IP address, in this case `1.2.3.4`, is used as unique ID): see Listing 3.

Next, the CSRs must be sent to the CA, which will generate the signed certificates out of the certificate requests. For instance, assuming the CSR file is in the current directory: see Listing 4.

Finally, you need to copy the newly-generated certificates (the files ending in `.crt`) to the respective machines in the `/etc/isakmpd/certs/` directory, as well as the CA certificate (`/etc/ssl/ca.crt`) in `/etc/isakmpd/ca/`.

Configuration

So we have conveniently set up the system for IPsec use and generated all the required certificates for IKE peer authentication; now we're finally ready to configure our VPN connection. On OpenBSD, all the configuration for IPsec takes place in a single file, `/etc/ipsec.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>), which uses a very compact syntax, similar to `pf.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5>), to define almost every characteristic of the VPN; the basic format of the file is as follows:

`www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5`), which uses a very compact syntax, similar to `pf.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5>), to define almost every characteristic of the VPN; the basic format of the file is as follows:

- comment lines begin with a hash character (`#`) and extend to the end of the line;
- rules may span across multiple lines using the backslash character (`\`);
- network addresses can be specified in CIDR notation, as symbolic host names, interface names, or interface group names;
- to simplify the configuration file, macros can be used; macro names must start with a letter, may contain letters, numbers and underscores and must not be reserved words;
- certain parameters (such as IP addresses) can be expressed as lists; lists are comma-separated and enclosed in curly braces.

Listing 4. CSRs must be sent to the CA, assuming the CSR file is in the current directory

```
CA# env CERTIP=1.2.3.4 openssl x509 -req \
> -days 365 -in 1.2.3.4.csr -out 1.2.3.4.crt \
> -CA /etc/ssl/ca.crt -CAkey /etc/ssl/private/ca.key \
> -CAcreateserial -extfile /etc/ssl/x509v3.cnf -
    extensions x509v3_IPAddr
Signature ok
subject=/C=IT/ST=Italy/L=Milan/O=Kernel Panic Inc./
    OU=IPsec/CN=1.2.3.4/emailAddress=d
    anix@kernel-panic.it
Getting CA Private Key
Enter pass phrase for /etc/ssl/private/ca.key:
    <passphrase>
CA#
```

Listing 5. The syntax

```
ike [mode] [encap] [tmode] [proto protocol] \
    from src [port sport] [(srcnat)] to dst [port
    dport] \
    [local localip] [peer remote] \
    [mode auth algorithm enc algorithm group group] \
    [quick auth algorithm enc algorithm group group] \
    [srcid string] [dstid string] \
    [psk string] [tag string]
```

There are different types of `ipsec.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) rules, depending on whether you want IPsec flows and SAs to be set up automatically (using `isakmpd(8)` <http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) or manually; we will only consider the former case (which is usually what you want), so please refer to the documentation (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) for further details on manual setups. The syntax is as follows: see Listing 5.

Though it may look rather complex at first, actual rules are usually very short and simple because most of the parameters can be omitted, in which case the default values are used. But let's examine the rule syntax in detail:

- `ike [mode] [encap] [tmode]` – the `ike` keyword specifies that `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) must be used to automatically establish the Security Associations for this flow; `mode` can be either `active` (`isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) will immediately start negotiation of this tunnel), `passive` (to wait for an incoming request from the remote peer to start negotiation) or `dynamic` (to be used for hosts with dynamic IP addresses) and defaults to `active`; `encap`

specifies the encapsulation protocol and can be either `esp` (default) or `ah`; `tmode` is the transport mode to use, i.e. `tunnel` (default) or `transport`.

- `proto protocol` – Restrict the flow to a specific IP protocol (e.g. TCP, UDP, ICMP); by default all protocols are allowed.
- `from src [port sport] [(srcnat)] to dst [port dport]` – Specify the source and destination addresses of the packets that this rule applies to; you may also specify source and/or destination ports, but only in conjunction with the TCP or UDP protocols. The `srcnat` parameter can be used to specify the actual source address in outgoing NAT/BINAT scenarios.
- `local localip peer remote` – Specify the local and remote endpoints of the VPN; the local endpoint is required only if the machine has multiple addresses; the remote endpoint can be omitted if it corresponds to the `dst` parameter.
- `mode auth algorithm enc algorithm group group` – Specify the mode (`main` or `aggressive`) and cryptographic transforms to be used for IKE phase 1 negotiation; please refer to the documentation (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) for a complete list of the possible values and their defaults.
- `quick auth algorithm enc algorithm group group`
- Specify the cryptographic transforms to be used for IKE phase 2 negotiation; please refer to the

Listing 6. The configuration files for the site-to-site VPN we're setting up

```
/etc/ipsec.conf
# Macros
ext_if      = "r10"                                #
                External interface (1.2.3.4)
local_net   = "172.16.0.0/24"                       #
                Local private network
remote_gw   = "5.6.7.8"                            #
                Remote IPsec gateway
remote_nets = "{192.168.0.0/24, 192.168.1.0/24}" #
                Remote private networks

# Set up the VPN between the gateway machines
ike esp from $ext_if to $remote_gw
# Between local gateway and remote networks
ike esp from $ext_if to $remote_nets peer $remote_gw
# Between the networks
ike esp from $local_net to $remote_nets peer $remote_gw
```

Listing 7. The configuration files for the site-to-site VPN we're setting up

```
/etc/ipsec.conf
# Macros
ext_if      = "r10"                                #
                External interface (5.6.7.8)
local_nets  = "{192.168.0.0/24, 192.168.1.0/24}" #
                Local private networks
remote_gw   = "1.2.3.4"                            #
                Remote IPsec gateway
remote_net  = "172.16.0.0/24"                       #
                Remote private network

# Set up the VPN between the gateway machines
ike esp from $ext_if to $remote_gw
# Between local gateway and remote network
ike passive esp from $ext_if to $remote_net peer
                $remote_gw
# Between the networks
ike esp from $local_nets to $remote_net peer $remote_gw
```


documentation (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) for a complete list of the possible values and their defaults.

- `srcid string dstid string` – Define the unique ID that `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) will use as the identity of the local (`srcid`) and remote (`dstid`) peer; if omitted, the IP address is used.
- `psk string` – Use a pre-shared key for authentication instead of `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>).
- `tag string` – Add a `pf(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pf&sektion=4>) tag to IPsec packets matching this rule.

So let's write the configuration files for the site-to-site VPN we're setting up; as you'll see, it's a really trivial task and a few rules will do. On the VPN1 host, the `/etc/ipsec.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) file will look like this: see Listing 6 and on VPN2: see Listing 7.

Now we are ready to start the `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) daemon on both gateways; we will make it run in the foreground (`-d` option) in order to easily notice any errors:

```
# isakmpd -K -d
```

Then, again on both gateways, we can parse `ipsec.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) rules (`-n` option of `ipsecctl(8)` <http://www.openbsd.org/cgi-bin/man.cgi?query=ipsecctl&sektion=8>) and, if no errors show up, load them:

```
# ipsecctl -n -f /etc/ipsec.conf
# ipsecctl -f /etc/ipsec.conf
```

You can check that IPsec flows and SAs have been correctly set up by running `ipsecctl(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsecctl&sektion=8>) with the `-s all` option; for example: see Listing 8.

Well, since everything seems to be working fine, we can configure the system to automatically start the VPN at boot by adding the following variables in `/etc/rc.conf.local(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8>) on both security gateways:

```
/etc/rc.conf.local
isakmpd_flags="-K" # Avoid keynote(4) policy checking
ipsec=YES # Load ipsec.conf(5) rules
```

Listing 8. You can check that IPsec flows and SAs have been correctly set up by running `ipsecctl(8)`

```
VPN1# ipsecctl -s all
FLOWS:
flow esp in from 192.168.0.0/24 to 1.2.3.4 peer 5.6.7.8
           srcid 1.2.3.4/32 dstid 5.6.7.8/32
           type use
flow esp out from 1.2.3.4 to 192.168.0.0/24 peer 5.6.7.8
           srcid 1.2.3.4/32 dstid 5.6.7.8/32
           type require
flow esp in from 192.168.1.0/24 to 1.2.3.4 peer 5.6.7.8
           srcid 1.2.3.4/32 dstid 5.6.7.8/32
           type use
flow esp out from 1.2.3.4 to 192.168.1.0/24 peer 5.6.7.8
           srcid 1.2.3.4/32 dstid 5.6.7.8/32
           type require
[ ... ]

SAD:
esp tunnel from 5.6.7.8 to 1.2.3.4 spi 0x027fa231 auth
           hmac-sha2-256 enc aes
esp tunnel from 1.2.3.4 to 5.6.7.8 spi 0x13ebc203 auth
           hmac-sha2-256 enc aes
esp tunnel from 1.2.3.4 to 5.6.7.8 spi 0x25da85ac auth
           hmac-sha2-256 enc aes
esp tunnel from 5.6.7.8 to 1.2.3.4 spi 0x891aa39b auth
           hmac-sha2-256 enc aes
[ ... ]
VPN1#
```

Listing 9. The sample configuration file

```
/etc/sasyncd.conf
# carp(4) interface to track state changes on
interface carp0
# Interface group to use to suppress carp(4)
preemption during boot

group carp
# sasyncd(8) peer IP address or hostname. Multiple
'peer' statements are allowed
peer 172.16.0.253
# Shared AES key used to encrypt messages between
sasyncd(8) hosts. It can be
# generated with the openssl(1) command 'openssl rand
-hex 32'
sharedkey 0x115c413529ba5ac96b208d83a50473b3e6ade60e66
c59a10a944ad3d273148dd
```

Packet filtering

IPsec traffic can be filtered on the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) interface, where it appears unencrypted before encapsulation and after decapsulation. The following are the main points to keep in mind for filtering IPsec traffic:

- IPsec protocols (<http://www.kernel-panic.it/openbsd/vpn/vpn2.html#vpn-2.1>) (AH and/or ESP) must be explicitly allowed on the external interface; e.g.:

```
# Allow ESP encapsulated IPsec traffic on the external
interface
pass in on $ext_if proto esp from $remote_gw to $ext_if
pass out on $ext_if proto esp from $ext_if to $remote_gw
```

- `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) requires that UDP traffic on ports 500 (`isakmp`) and 4500 (`ipsec-nat-t`) be allowed on the external interface; e.g.:

```
# Allow isakmpd(8) traffic on the external interface
pass in on $ext_if proto udp from $remote_gw to $ext_if \
port {isakmp, ipsec-nat-t}
pass out on $ext_if proto udp from $ext_if to $remote_gw \
port {isakmp, ipsec-nat-t}
```

- if the VPN is in tunnel mode, IP-in-IP traffic between the two gateways must be allowed on the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) interface; e.g.:

```
# Allow IP-in-IP traffic between the gateways on the enc(4)
interface
pass in on enc0 proto ipencap from $remote_gw to $ext_if
keep state \
(if-bound)
pass out on enc0 proto ipencap from $ext_if to $remote_gw
keep state \
(if-bound)
```

- as stated before, IPsec traffic filtering is done on the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) interface, where it appears unencrypted. State on the `enc(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=enc&sektion=4>) interface should be interface `bound` (<http://www.openbsd.org/faq/pf/options.html#state-policy>); e.g.:

Listing 10. Initializing parameters in the vars file with your organization's data to avoid being prompted for the same information every time you create a new certificate

```
/usr/local/share/examples/openvpn/easy-rsa/2.0/vars
export EASY_RSA="'pwd'"

export OPENSSSL="openssl"
export PKCS11TOOL="pkcs11-tool"
export GREP="grep"

export KEY_CONFIG="$EASY_RSA/openssl.cnf"
export KEY_DIR="$EASY_RSA/keys"

echo NOTE: If you run ./clean-all, I will be doing a
rm -rf on $KEY_DIR

export PKCS11_MODULE_PATH="dummy"
export PKCS11_PIN="dummy"

export KEY_SIZE=1024
export CA_EXPIRE=3650
export KEY_EXPIRE=3650

export KEY_COUNTRY="IT"
export KEY_PROVINCE="Italy"
export KEY_CITY="Milan"
export KEY_ORG="Kernel Panic Inc."
export KEY_EMAIL="danix@kernel-panic.it"
```

Listing 11. Initializing the PKI by building the Diffie-Hellman parameters and creating the root CA certificate and key

```
# cd /usr/local/share/examples/openvpn/easy-rsa/2.0/
# ./vars
NOTE: when you run ./clean-all, I will be doing a rm
-rf on /usr/local/share/example/
opevpn/easy-rsa/2.0/keys

# ./clean-all
# ./build-dh
Generating DH parameters, 1024 bit long safe prime,
generator 2

This is going to take a long time
[ ... ]
# ./pkitool --initca
Using CA Common Name: Kernel Panic Inc. CA
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'ca.key'
-----
#
```

```
# Filter unencrypted VPN traffic on the enc(4) interface
pass in on enc0 from $remote_nets to $int_if:network keep
    state (if-bound)
pass out on enc0 from $int_if:network to $remote_nets keep
    state (if-bound)
```

Redundant VPNs with sasyncd(8)

One of the most interesting features of OpenBSD's implementation of the IPsec protocol is the possibility to set up multiple VPN gateways in a redundant configuration, allowing for transparent failover of VPN connections without any loss of connectivity.

Typically, in OpenBSD, redundancy at the network level is achieved through the `carp(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4>) protocol, which allows multiple hosts on the same local network to share a common IP address. Redundancy at the logical VPN layer, instead, is provided by the `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) daemon, which allows the synchronization of IPsec SA and SPD information between multiple IPsec gateways.

We have already covered the `carp(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4>) protocol in a previous document (<http://www.kernel-panic.it/openbsd/carp/index.html>) about redundant firewalls, so we won't come back to this topic now; therefore, I assume that you already have a working `carp(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=carp&sektion=4>) setup and that you have modified your configuration accordingly (in particular the `ipsec.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsec.conf&sektion=5>) and `pf.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pf.conf&sektion=5>) files).

Please note that, as stated in the documentation (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>), for SAs with replay protection enabled, such as those created by `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>), the `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) hosts must have `pfsync(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4>) enabled to synchronize the in-kernel SA replay

Listing 12. Creating the certificate and key for the VPN server

```
# ./pkitool --server vpn1.kernel-panic.it
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'vpn1.kernel-panic.it.key'
-----
Using configuration from /usr/local/share/examples/
    openvpn/easy-rsa/2.0/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName             :PRINTABLE:'IT'
stateOrProvinceName    :PRINTABLE:'Italy'
localityName            :PRINTABLE:'Milan'
organizationName        :PRINTABLE:'Kernel Panic Inc.'
commonName               :PRINTABLE:'vpn1.kernel-
    panic.it'
emailAddress             :IA5STRING:'danix@kernel-
    panic.it'
Certificate is to be certified until Jun  2 08:41:51
    2019 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
#
```

Listing 13. Using the pkitool utility to generate as many client certificates as we need

```
# ./pkitool vpn2.kernel-panic.it
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'vpn2.kernel-panic.it.key'
-----
Using configuration from /usr/local/share/examples/
    openvpn/easy-rsa/2.0/openssl.cnf
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName             :PRINTABLE:'IT'
stateOrProvinceName    :PRINTABLE:'Italy'
localityName            :PRINTABLE:'Milan'
organizationName        :PRINTABLE:'Kernel Panic Inc.'
commonName               :PRINTABLE:'vpn2.kernel-
    panic.it'
emailAddress             :IA5STRING:'danix@kernel-
    panic.it'
Certificate is to be certified until Jun  2 08:47:25
    2019 GMT (3650 days)

Write out database with 1 new entries
Data Base Updated
#
```


counters (for a detailed discussion of the `pfsync(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=pfsync&sektion=4>) protocol, please refer to [CARP] <http://www.kernel-panic.it/openbsd/carp/carp5.html>).

The `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) daemon is configured through the `/etc/sasyncd.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd.conf&sektion=5>) file, which has a rather self-explanatory syntax; below is a sample configuration file: see Listing 9. Since `sasyncd.conf(5)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd.conf&sektion=5>)

contains the shared secret key used to encrypt data between the `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) hosts, it should have restrictive permissions (600) and belong to the `root` or `_isakmpd` user:

```
# chown root /etc/sasyncd.conf
# chmod 600 /etc/sasyncd.conf
```

Well, now we're ready to run the `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>)

Listing 14. The sample configuration file

```
/etc/openvpn/server.conf
# Transport protocol to use. Available protocols are
    udp and tcp-server

proto udp
# TCP/UDP port to bind to
port 1194
# Name of the tun(4) device to use
dev tun0

# Uncomment to enable the management interface on port
    1195. The password file
# only contains the management password on a single
    line.
#management 127.0.0.1 1195 /etc/openvpn/private/
    mgmt.pwd

# Path to the CA certificate
ca /etc/openvpn/ca.crt
# Path to the server's certificate file
cert /etc/openvpn/vpn1.kernel-panic.it.crt
# Path to the private key file
key /etc/openvpn/private/vpn1.kernel-panic.it.key
# Path to the file containing the Diffie-Hellman
    parameters
dh /etc/openvpn/dh1024.pem

# Address range for the tun(4) interfaces
server 10.0.1.0 255.255.255.0
# Uncomment to allow clients to dynamically change
    address (useful for
# road-warriors)
#float

# Send periodic keepalive messages
keepalive 10 120

# Use lzo compression to reduce network utilization
comp-lzo

# User the OpenVPN daemon should run as
user _openvpn
# Group the OpenVPN daemon should run as
group _openvpn
# Make the server daemonize after initialization
daemon openvpn

# Don't re-read key files upon receiving a SIGUSR1
    signal
persist-key
# Don't close and reopen the tun(4) device upon
    receiving a SIGUSR1 signal
persist-tun

# Add a route to the local network to the client's
    routing table
push "route 172.16.0.0 255.255.255.0"
# Add routes to the remote networks to the server's
    routing table
route 192.168.0.0 255.255.255.0
route 192.168.1.0 255.255.255.0
# Directory for client-specific configuration files
client-config-dir /etc/openvpn/ccd

# Uncomment to periodically write status information
    to the specified file
#status /var/log/openvpn-status.log
# Uncomment to raise verbosity level for debugging
#verb 11
```

NYCBSDCon '10

We are proud to announce that the bi-annual **NYCBSDCon 2010** will be held at Manhattan's prestigious Cooper Union on **November 12-14, 2010**. This year's conference is sure to build on the successes of previous years featuring a great array of speakers and topics, with an exciting and diverse crowd representing all the BSD projects. The topics for talks include:

- BSD Professional Certification Update: The Lab Exam – **Jim Brown**
- Practical Security Event Auditing in FreeBSD – **Christian Brueffer**
- BSD Firewalling with pfSense – **Chris Buechler**
- BSD: Choose Your Own Adventure – **Jason Dixon**
- Managing Multiple Machines - FreeBSD, radmind and LDAP – **Michael Graziano**
- Isilon and FreeBSD – **Zachary Loafman**
- Escaping the Database Doldrums – **James K. Lowden**
- BSD Needs Books – **Michael Lucas**
- The Automated Testing Framework – **Julio Merino**
- PC-SYSINSTALL: A new system installation backend for PC-BSD and FreeBSD – **Kris Moore**
- Standing on the Shoulder's of Giants: Key Players and Events in BSD History – **Jeremy C. Reed**
- A Provider's Perspective on IPv6: Getting Beyond the User Experience – **Massimiliano Stucchi**
- Managing 600 OpenBSD-Based Firewalls in Microsoft-centric Small and Medium Business Networks – **Lawrence Teo**

Tickets:

\$95.00 for early registration ending November 1st

\$125 from November 2nd to November 11th

\$150 to buy tickets at the door



Students:

Free for all Cooper Union students and staff – please ask local acm/ieee group for registration code

\$75.00 Full-time students - must show a current ID upon entry

<http://www.nycbsdcon.org>



`sektion=8`) daemon on the redundant gateways; but first we need to restart `isakmpd(8)` (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) with the `-s` option, which is mandatory on redundant setups (remember to add it also to `isakmpd_flags` in `/etc/rc.conf.local(8)` <http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8>):

```
# pkill isakmpd
# isakmpd -S -K
# sasyncd
```

You can use `ipsecctl(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ipsecctl&sektion=8>) to verify that SAs are correctly synchronized between the IPsec gateways. Finally, if everything is working fine, we only have to add the following variable to the `/etc/rc.conf.local(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=rc.conf.local&sektion=8>) file to automatically start `sasyncd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) on boot.

```
/etc/rc.conf.local
sasyncd_flags=""
```

Listing 15. Make sure that the configuration matches the server configuration

```
/etc/openvpn/client.conf
# Act as a client
client
# IP address (or hostname) and port of the OpenVPN
server. You may specify
# multiple 'remote' options for redundancy.
remote 1.2.3.4 1194

# Transport protocol to use. Available protocols are
udp and tcp-client
proto udp
# Name of the tun(4) device to use
dev tun0

# Uncomment if you connect through an HTTP proxy. The
authfile must contain
# user and password on 2 lines. The authentication
type can be 'none', 'basic'
# or 'ntlm'
#http-proxy proxy_addr proxy_port /etc/openvpn/
private/authfile auth_type
# Make the server daemonize after initialization
daemon openvpn
# Send periodic keepalive messages
keepalive 10 120

# Don't bind to the local address and port, i.e. don't
wait for incoming
# connections
nobind

# User the OpenVPN daemon should run as
user _openvpn
# Group the OpenVPN daemon should run as
group _openvpn

# Directory to chroot to after initialization
chroot /var/empty

# Don't re-read key files upon receiving a SIGUSR1
signal
persist-key
# Don't close and reopen the tun(4) device upon
receiving a SIGUSR1 signal
persist-tun

# Path to the CA certificate
ca /etc/openvpn/ca.crt
# Path to the client's certificate file
cert /etc/openvpn/vpn2.kernel-panic.it.crt
# Path to the private key file
key /etc/openvpn/private/vpn2.kernel-panic.it.key

# Require that the peer certificate has the nsCertType
field set to 'server'
ns-cert-type server
# Use lzo compression to reduce network utilization
comp-lzo

# Uncomment to periodically write status information
to the specified file
#status /var/log/openvpn-status.log
# Uncomment to raise verbosity level for debugging
#verb 11
```


Note

`sasyncd`(8) (<http://www.openbsd.org/cgi-bin/man.cgi?query=sasyncd&sektion=8>) must be manually restarted every time `isakmpd`(8) (<http://www.kernel-panic.it/cgi-bin/man.cgi?query=isakmpd&sektion=8>) is restarted.

OpenVPN

OpenVPN (<http://www.openvpn.net/>) is a full-featured SSL VPN which implements OSI layer 2 or 3 secure network extension using the industry standard SSL/TLS protocol, supports flexible client authentication methods based on certificates, smart cards, and/or username/password credentials, and allows user or group-specific access control policies using firewall rules applied to the VPN virtual interface [OVPN-HOWTO] (<http://www.openvpn.net/index.php/open-source/documentation/howto.html>). Its cross-platform portability, renown security and ease of use have made OpenVPN one of the most popular VPN solutions today.

Listing 16. Creating and configuring the `tun(4)` network device and setting up the appropriate routes to the remote network(s) and hosts

```
VPN1# ifconfig tun0 create
VPN1# ifconfig tun0 10.0.0.1 10.0.0.2 netmask 0xffffffff
VPN1# route add 192.168.0.0/24 10.0.0.2
VPN1# route add 192.168.1.0/24 10.0.0.2

VPN2# ifconfig tun0 create
VPN2# ifconfig tun0 10.0.0.2 10.0.0.1 netmask 0xfffff0
VPN2# route add 172.16.0.0/24 10.0.0.1
```

Listing 17. Creating an RSA authentication key for the user with the `ssh-keygen`(1)

```
VPN2# ssh-keygen -b 2048 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_
rsa): <enter>
Enter passphrase (empty for no passphrase): <enter>
Enter same passphrase again: <enter>
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
cd:9c:3b:3a:c0:92:7f:c2:9b:6e:3a:48:dc:50:a4:2a
root@vpn2.kernel-panic.it
VPN2#
```

Unlike IPsec, OpenVPN is not tightly integrated into the Operating System's kernel, but runs as a user-mode daemon and communicates with the TCP/IP stack via a `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) pseudo-device. Please refer to [OVPN-SEC2] (<http://www.openvpn.net/index.php/open-source/documentation/security-overview.html>) for a detailed overview of the OpenVPN protocol and security model.

In the next paragraphs, we will implement the same VPN topology as in the previous chapter, though replacing IPsec with OpenVPN. The VPN1 machine will act as the server and wait for incoming connections from VPN2.

Installation and configuration

OpenVPN installation simply requires adding a couple of packages (<http://www.openbsd.org/faq/faq15.html#PkgInstall>) on both server and client(s):

- `lzo-xx.tgz`
- `openvpn-x.x.tgz`

Setting up the PKI

The first step in configuring OpenVPN is to set up the Public Key Infrastructure, by creating:

- a root CA certificate and private key;
- a certificate and private key for the OpenVPN server;
- a separate certificate and private key for each client that will connect to the VPN.

The CA private key will be used to sign the server and client certificates; this will allow the two VPN endpoints to mutually authenticate each other simply by verifying the CA signature of the other party's certificate, without having to previously know any other certificate but their own (see [OVPN-PKI] (<http://www.openvpn.net/index.php/open-source/documentation/howto.html#pki>) for further details).

OpenVPN provides a set of scripts, located in `/usr/local/share/examples/openvpn/easy-rsa/2.0/`, that greatly simplify the process of creating and managing the PKI. These scripts require, as a preliminary step, that you initialize a bunch of parameters in the `vars` file with your organization's data, to avoid being prompted for the same information every time you create a new certificate: see Listing 10.

Now, after sourcing the `vars` file, you can initialize the PKI by building the Diffie-Hellman parameters and creating the root CA certificate and key: see Listing 11.

The next step is creating the certificate and key for the VPN server: see Listing 12.

Next, we will use the `pktool` utility to generate as many client certificates as we need: see Listing 13.

So we have generated all the certificates and keys we need; you can find them in the `/usr/local/share/examples/openvpn/easy-rsa/2.0/keys` directory, ready to be copied to the appropriate machines. But before proceeding to copy the key files, we need to create, on both server and clients, the directory (`/etc/openvpn/private`) that will contain the private keys and assign it restrictive permissions to prevent unauthorized access.

```
# mkdir -p /etc/openvpn/private
# chmod 700 /etc/openvpn/private
```

The following are the files that must be copied from the CA-signing machine to the OpenVPN hosts:

- the `ca.crt` file (the CA certificate) must be copied to the `/etc/openvpn` directory of all the machines (server and clients);
- the `ca.key` file (the CA private key) must reside only on the key-signing machine; if you want the OpenVPN server to act also as the CA, just move this file to the `/etc/openvpn/private/` directory of the server machine;
- the `dh1024.pem` file (the Diffie Hellman parameters) must be placed in the `/etc/openvpn` directory of the server machine;
- the remaining `.crt` and `.key` files (i.e. the certificates and private keys of the server and the clients) must be copied to the respective machines; private keys must be stored in `/etc/openvpn/private` and certificates should reside in `/etc/openvpn`.

Finally, remember to delete all the files in `/usr/local/share/examples/openvpn/easy-rsa/2.0/keys/`:

```
# ./clean-all
```

Server configuration

OpenVPN supports a number of configuration parameters, allowing you to deeply customize its behaviour. These parameters can be either passed from the command-line or in a configuration file. Omitted parameters take the default value.

Below is a sample configuration file (see [OVPN-MAN] (<http://www.openvpn.net/man.html>) for a complete list of all the available parameters): see Listing 14.

The `client-config-dir` directive in the server configuration file allows you to specify a directory containing client-specific configuration files. These files must have the same name as the client's X509 Common Name, specified during the creation of the certificates. In this case, we will create a file named `/etc/openvpn/ccd/vpn2.kernel-panic.it`,

which will specify which private networks can be reached through the OpenVPN client:

```
/etc/openvpn/ccd/vpn2.kernel-panic.it
iroute 192.168.0.0 255.255.255.0
iroute 192.168.1.0 255.255.255.0
```

Though very similar, both the `route` and `iroute` directives are necessary, because `route` controls the routing from the kernel to the OpenVPN server (via the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) interface) while `iroute` controls the routing from the OpenVPN server to the remote clients [OVPN-HOWTO] (<http://www.openvpn.net/index.php/open-source/documentation/howto.html>).

Client configuration

The client-side configuration is pretty similar to server-side configuration. The address and port of the server are specified via the `remote` directive. Make sure that the configuration matches the server configuration, in particular that they both use the same protocol, device type and that they both enable or disable lzo compression (see Listing 15).

Starting the VPN

Before starting the VPN, we have to enable IP forwarding on both gateways, since they will have to perform routing of network traffic:

```
# sysctl net.inet.ip.forwarding=1
```

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 IP forwarding after reboot:

Listing 18. Making sure that this file has restrictive permissions

```
VPN1# cat authorized_keys
ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAQEAoWqL6wpgL5j3dlFtdY
      WT+cc72F/FtMhmTBLUEcCMQY8
/V9CptSn7yCC+1R5xhZD8WO3d1lc7R8pUHP77A3omFruEpk4pREui
      sHnMtA6XyVFoxshhVlosyoQ/HJ
w6BhTmmGDCCyNsPmQyAPi9V7rL4NNS1116mFXqLDNth6wf0qjo33BU
      RsyKR6xxmt5QBhDpCBDe13EwLh
gE2Jy06XJZKa62/WU6ofbnXZWWGX8ZsbCPxqu3E0BhMw1UgA1Igks
      GfOcB4rgV+qpcPUf13fQM67Mc7
Nwhh7jqkaCTpu/vs40pBft6j9eVxMgRGylg4a9tBcZY2588wPZZThp
      x/sw== root@vpn2.kernel-pa
nic.it
VPN1# chmod 600 /root/.ssh/authorized_keys
```



BSD day 2010

Argentina

5 - 6 Nov.

Faculty of Exact and Natural Sciences,
University of Buenos Aires
Buenos Aires City, Argentina

BSDday Argentina is a series of technical conferences and talks made by and for developers, sysadmins and users interested in the BSD operating systems, and related free software projects.

Call for Papers!

Call for Papers is open and you can send us your proposal to llamcha@bsdday.org

Complete info of CFP:

<http://www.bsdday.org.ar/consola-en/cfp.txt>

Registration Open!

<http://www.bsdday.org.ar/consola-en/>



Follow us!
[@bsdday](https://twitter.com/bsdday)




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

So we're ready to start the VPN! Just run the following command on the server:

```
vpn1# openvpn --config /etc/openvpn/server.conf
```

and the following on the client:

```
vpn2# openvpn --config /etc/openvpn/client.conf
```

To finish, we just have to create the configuration file for the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) interface on the server (starting OpenVPN from this file improves compatibility with PF):

```
/etc/hostname.tun0
up
```

```
!/usr/local/sbin/openvpn --daemon --config /etc/openvpn/
server.conf
```

and on the client:

```
/etc/hostname.tun0
up
!/usr/local/sbin/openvpn --daemon --config /etc/openvpn/
client.conf
```

OpenSSH

OpenSSH (<http://www.openssh.org/>) is a *FREE* version of the SSH connectivity tools developed by the OpenBSD project (<http://www.openbsd.org/>). It certainly needs no introduction as it has now grown into the de facto standard for secure console access over the Internet, widely supplanting the infamous `r` commands.

Beginning with version 4.3 (<http://www.openssh.com/txt/release-4.3>), OpenSSH also provides secure VPN tunneling capabilities at both layer 2 and layer 3 of the OSI model, by using the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) pseudo-device to encapsulate network traffic within SSH packets.

Of the VPN solutions we've seen so far, OpenSSH-based VPNs are by far the simplest to use and the fastest to implement; however, they also imply a considerable overhead. As a consequence, the documentation (<http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1&format=html#SSH-BASED+VIRTUAL>) warns that OpenSSH VPNs *may be more suited to temporary setups, such as for wireless VPNs*, and recommends the use of IPsec (<http://www.kernel-panic.it/openbsd/vpn/vpn2.html>) for more permanent VPNs.

Configuration

We will configure the same VPN topology (<http://www.kernel-panic.it/openbsd/vpn/vpn3.html#vpn>) as in the previous chapters; the VPN1 machine will act as the OpenSSH server, waiting for connections from VPN2.

First off, we need to enable tunneling support on the OpenSSH server, since this feature is disabled by default. This is achieved by setting the `PermitTunnel` parameter in `/etc/ssh/sshd_config(5)` (http://www.openbsd.org/cgi-bin/man.cgi?query=sshd_config&sektion=5) to `ethernet` or `point-to-point`, depending on whether you want the VPN to operate, respectively, at layer 2 of the OSI model, layer 3 or both.

```
/etc/ssh/sshd_config
[ ... ]
# Enable layer-3 tunneling. Change the value to 'ethernet'
```

References

- [OBSD39 <http://www.openbsd.com/39.html>] – The OpenBSD 3.9 Release
- [OVPN-FAQ <http://www.openvpn.net/index.php/open-source/faq.html#bridge2>] – OpenVPN FAQ, What is the difference between bridging and routing?
- [OVPN-SEC <http://www.openvpn.net/index.php/open-source/faq.html#security-issues>] – Are there any known security vulnerabilities with OpenVPN?
- [RFC4301 <http://tools.ietf.org/html/rfc4301>] – RFC 4301, Security Architecture for the Internet Protocol
- [RFC4302 <http://tools.ietf.org/html/rfc4301>] – RFC 4302, IP Authentication Header
- [RFC4303 <http://tools.ietf.org/html/rfc4301>] – RFC 4303, IP Encapsulating Security Payload (ESP)
- [SCHNEIER] – A Cryptographic Evaluation of IPsec, N. Ferguson and B. Schneier
- [RFC4306 <http://www.schneier.com/paper-ipsec.pdf>] – RFC 4306, Internet Key Exchange (IKEv2) Protocol
- [RFC3173 <http://tools.ietf.org/html/rfc3173>] – RFC 3173, IP Payload Compression Protocol (IPComp)
- [CARP <http://www.kernel-panic.it/openbsd/carp/>] – Redundant firewalls with OpenBSD, CARP and pfsync
- [OVPN-HOWTO <http://www.openvpn.net/index.php/open-source/documentation/howto.html>] – OpenVPN 2.0 HOWTO
- [OVPN-SEC2 <http://www.openvpn.net/index.php/open-source/documentation/security-overview.html>] – OpenVPN Security Overview
- [OVPN-PKI <http://www.openvpn.net/index.php/open-source/documentation/howto.html#pki>] – Setting up your own Certificate Authority (CA) and generating certificates and keys for an OpenVPN server and multiple clients
- [OVPN-MAN <http://www.openvpn.net/man.html>] – OpenVPN 2.0.x Man Page

Carry the card that supports BSD events around the world



BSD Fund is proud to sponsor of BSDCan 2010 and meetBSD California 2010 thanks to revenue from the BSD Fund Visa. A donation is made every time you use the card and simply charging your travel to an event can help sponsor that event.

BSD Fund also raises money through direct donations on behalf of BSD projects such as the pcc compiler.

Find our more at www.bsdfund.org

```

        for layer-2 tunneling
PermitTunnel point-to-point

```

On the client side, the `Tunnel` parameter, in `/etc/ssh/ssh_config(5)` (http://www.openbsd.org/cgi-bin/man.cgi?query=ssh_config&sektion=5), must be set to the same value as `PermitTunnel` on the OpenSSH server:

```

/etc/ssh/ssh_config
[ ... ]

# Enable layer-3 tunneling. Change the value to 'ethernet'
        for layer-2 tunneling
Tunnel point-to-point

```

Next, we need to enable IP forwarding on both VPN gateways, since they will have to perform routing of network traffic:

```
# sysctl net.inet.ip.forwarding=1
```

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

```

And the configuration phase is over: how could it be easier? Now we only have to force `sshd(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=sshd&sektion=8>) to reread its configuration file by sending it a `SIGHUP` signal:

```
VPN1# pkill -HUP sshd
```

Starting the VPN

Before actually firing up the VPN, we will carry out a couple of preliminary steps on both the OpenSSH server and the client, i.e. creating and configuring the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) network device and setting up the appropriate routes to the remote network(s) and hosts (see Listing 16).

Well, we're finally ready to initiate the `ssh(1)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1>) connection and establish the VPN tunnel. The `-f`

option requests `ssh(1)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ssh&sektion=1>) to go to background after prompting for the password, and the `-w` option specifies the numerical ID of the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) device in charge of forwarding VPN traffic; in our setup, we're using `tun0` on both client and server, so we will set this option to `0:0`.

```

VPN2# ssh -f -w 0:0 1.2.3.4 true
root@VPN1's password: pAssWOrd

```

Finishing touches

To finish, we will configure the client machine to automatically start the VPN on boot. To prevent the system from hanging during startup until the user enters the password, we need to create an RSA authentication key for the user with the `ssh-keygen(1)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=ssh-keygen&sektion=1>) utility: see Listing 17, and add the newly-generated key, contained in `/root/.ssh/id_rsa.pub`, to the authorized keys in `/root/.ssh/authorized_keys` on the server; please make sure that this file has restrictive permissions (`600`): see Listing 18.

Next, on the server side, we need to create the configuration file for the `tun(4)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=tun&sektion=4>) pseudo-device, `/etc/hostname.tun0`, which will also include the necessary static routes:

```

/etc/hostname.tun0
10.0.0.1 10.0.0.2 netmask 0xffffffff
!route add 192.168.0.0/24 10.0.0.2 >/dev/null 2>&1
!route add 192.168.1.0/24 10.0.0.2 >/dev/null 2>&1

```

Similarly, on the client side, we will create the `/etc/hostname.tun0` configuration file :

```

/etc/hostname.tun0
10.0.0.2 10.0.0.1 netmask 0xffffffff
!route add 172.16.0.0/24 10.0.0.1 >/dev/null 2>&1

```

but also add the VPN start command in `/etc/rc.local(8)` (<http://www.openbsd.org/cgi-bin/man.cgi?query=rc.local&sektion=8>).

```

/etc/rc.local
[ ... ]
echo -n ' OpenSSH-VPN'
/usr/bin/ssh -f -w 0:0 1.2.3.4
true

```

DANIELE MAZZOCCHIO

Latest version: <http://www.kernel-panic.it/openbsd/vpn/>

Bibliography

VPNs Illustrated: Tunnels, VPNs, and IPsec, Jon C. Snader, Addison Wesley, 2006

HAKING

PRACTICAL PROTECTION

APC
by Schneider Electric

PROTECT YOUR COMPUTER,
THE ENVIRONMENT, AND YOUR WALLET

HAKING

PRACTICAL PROTECTION HARD CORE IT SECURITY MAGAZINE

MOBILE EXPLOITATION


PRIVACY KEEPING AND EXPLOITATION METHODS

EXPLOITING NULL POINTER DEREFERENCES
MOVEMENT ON THE MOBILE EXPLOIT FRONT
METHODS OF SECRECY
BRUTE FORCING USER NAMES
DATA MINING AS A TOOL FOR SECURITY

**MOBILE WEB:
PRIVACY KEEPING AND
EXPLOITATION METHODS**

**INTELLIGENCE REPORT:
ANALYSIS OF A SPEAR
PHISHING ATTACK**

**VIDEOJACKING:
HIJACKING IP VIDEO CALLS**

APPLICATIONS ON THE CD 

CERTIFIED WIRELESS NETWORK
ADMINISTRATOR TRAINING BY SEQRIT.ORG
DOUBLE ANTI-SPY PRO TRIAL

Vol.5 No.2 Price USD 14.99
Issue 2/2010(27) ISSN: 1733-7186



PLUS

A LOOK AT THE MALWARE TRENDS
EXPECTED IN 2010 BY JULIAN EVANS

**CWNA
TRAINING**

IT SECURITY MAGAZINE

Closed-source and

unsupported drivers with FreeBSD

Sooner or later you come to a conclusion that you need to have an enhanced mobility throughout your home place. And you decide to purchase a Wi-Fi card and put it into a home gate-keeper.

What you will learn...

- How to perform a bare metal installation of FreeBSD with networking enabled etc.

What you should know...

- How to patch, upgrade and install ports, initially configure Apache, PHP, MySQL and Drupal)

Do you know about troubles that could bring this simple transaction like WiFi network card purchase?

Trivia

Some might ask – is it necessary to buy a WiFi-card instead of a simple *AccessPoint* (AP)? At first glance you

Listing 1. My home router runs a stable release of FreeBSD 6.2

```
$ uname -a
FreeBSD bridge2 6.2-RELEASE FreeBSD 6.2-RELEASE #3: Mon Aug  4 17:28:07 MSD 2008   anton@bridge2:/usr/obj/usr/
src/sys/bridge2  i386
```

Listing 2. A new WiFi card isn't identified correctly by FreeBSD

```
$ pciconf -lv
xl0@pci1:4:0:   class=0x020000 card=0x100010b7 chip=0x920010b7 rev=0x78 hdr=0x00
  vendor      = '3COM Corp, Networking Division'
  device      = '3C905C-TX Fast EtherLink for PC Management NIC'
  class       = network
  subclass    = ethernet
rl0@pci1:5:0:   class=0x020000 card=0x813910ec chip=0x813910ec rev=0x10 hdr=0x00
  vendor      = 'Realtek Semiconductor'
  device      = 'RT8139 (A/B/C/810x/813x/C+) Fast Ethernet Adapter'
  class       = network
  subclass    = ethernet
none0@pci1:10:0: class=0x028000 card=0x3a711186 chip=0x03021814 rev=0x00 hdr=0x00
  vendor      = 'Ralink Technology, Corp'
  class       = network
```

Listing 3. *It seems that no wireless driver has been found for a new card*

```
$ kldstat
Id Refs Address      Size      Name
 1   33 0xc0400000 72aba0    kernel
 2    1 0xc0b51000 59f20     acpi.ko
 3    1 0xc2409000 6000      linprocfs.ko
 4    2 0xc241c000 16000     linux.ko
 5    1 0xc2456000 2a000     ipl.ko
 6    1 0xc25ad000 3000      ng_iface.ko
 7    1 0xc25b8000 6000      ng_ppp.ko
 8    1 0xc271e000 2000      star_saver.ko
 9    1 0xc2745000 2000      rtc.ko
11    1 0xc2a29000 4000      ng_pptpgre.ko
11    1 0xc2a2d000 4000      ng_ksocket.ko
12    1 0xc2a33000 4000      ng_vjc.ko
13    1 0xc2a39000 2000      ng_tcpmss.ko
14    1 0xc2a3b000 3000      ng_mppc.ko
15    1 0xc2a3e000 2000      rc4.ko
```

Listing 4. *Grepping through a kernel sources can give us a hint*

```
$ cd /usr/src/sys/i386/conf/
$ cat m-gw | egrep ral
# SCSI peripherals
device          agp          # support several AGP chipsets
# Parallel port
device          ppbus        # Parallel port bus (required)
device          plip         # TCP/IP over parallel
device          ppi         # Parallel port interface device
# If you've got a "dumb" serial or parallel PCI card that is
device          ral         # Ralink Technology RT2500 wireless NICs.
device          ural        # Ralink Technology RT2500USB wireless NICs
```

Listing 5. *It's always advisable to have kernel sources installed so you can figure out what exactly is supported (IDs are marked with bold)*

```
$ cd /usr/src/sys/dev/ral
$ cat if_ral_pci.c | grep 0x1814
{ 0x1814, 0x0201, "Ralink Technology RT2560" },
```

Listing 6. *Kernel from a vanilla FreeBSD 6.2 doesn't have a clue about our card (ID is marked with bold)*

```
none0@pci1:10:0: class=0x028000 card=0x3a711186 chip=0x03021814 rev=0x00 hdr=0x00
```

can figure out that there exist the fine models of ADSL-modems with wireless capabilities and that could work as AP. However, it should be noticed that:

- not all home connections to an Internet-provider go through a *copper* like phone- or cable-line;
- you simply need to add a WiFi-capability to an already working gate;
- a WiFi-card itself costs several times cheaper of AP.

Okay, you've crawled through hardware specifications available onsite, pros and contras of different models from different manufacturers (D-Link, ASUS, TrendNet, Edimax, etc.). And eventually come to a simple conclusion – although there exist several independent NIC manufactures, the most important about the WiFi-card is an WiFi-chip that used inside – it doesn't matter how your WiFi-card is labeled actually. Quite possibly they might be having the same WiFi chip. So you decide to skip that fancy feature like a guaranteed speed of 108 Mbit/s, and 801.11n specification and bought, for example, a budget card – D-Link DWA-510. Luckily, it

Listing 7. Conversion process with *ndisgen* utility

```

=====
----- Windows(r) driver converter -----
=====

This script is designed to guide you through the process
of converting a Windows(r) binary driver module and .INF
specification file into a FreeBSD ELF kernel module for use
with the NDIS compatibility system.
The following options are available:
1] Learn about the NDIS compatibility system
2] Convert individual firmware files
3] Convert driver
4] Exit

=====
----- Windows(r) driver converter -----
=====

                Driver file conversion

The script will now try to convert the .INF and .SYS files
using the ndiscvt(1) utility. This utility can handle most
.INF files; however, occasionally it can fail to parse some files
due to subtle syntax issues: the .INF syntax is very complex,
and the Windows(r) parser will sometimes allow files with small
syntax errors to be processed correctly which ndiscvt(1) will
not. If the conversion fails, you may have to edit the .INF
file by hand to remove the offending lines.
Press enter to try converting the files now:
Conversion was successful.
Press enter to continue...

=====
----- Windows(r) driver converter -----
=====

                Kernel module generation

The script will now try to generate the kernel driver module.
This is the last step. Once this module is generated, you should
be able to load it just like any other FreeBSD driver module.
Press enter to compile the stub module and generate the driver
module now:
Generating Makefile... done.
Building kernel module... done.
Cleaning up... done.
The file rt61_sys.ko has been successfully generated.
You can kldload this module to get started.
Press return to exit.

```

can be put into a MiniPC slot (comes as low profile card) – so it must be the right thing for a home router. D-Link claims that the card is compatible with 802.11b/g standard and drivers for Windows/Linux operating systems only are offered. But let's hope it will work with FreeBSD as well. Because my home router runs FreeBSD and I don't feel comfortable with changing it onto another OS.

Moreover, I don't think seriously about upgrading this version of FreeBSD at my home router. It is stable, quick in performance and does all necessary home network chores. Next step I do – power off the system in order to plug-in the new WiFi card and after that I switch on FreeBSD box. Unfortunately, no new network interface has been found.

Well, it's obvious that unknown card mapped as `none0@pci1:10:0` is our D-Link DWA510. The only thing we know – the WiFi-chip manufacturer. This is a Ralink company. Better than nothing. Our next step is to figure out what's wrong. Are there all drivers loaded during a bootup process?

Grepping the kernel

Nothing like the `ral0` or `ath0` is loaded. And although almost all WiFi-cards from a consumer market (according to a statistics) are based on chips manufactured by Ralink, Atheros and Marvell (Intel and Broadcom aren't taken into account as they operate in a hi-end market segment) – we hit the wrong turn. Let's make a more loose search in kernel sources – but now we know what we're looking for – string Ralink.

Grepping through a kernel configuration gives us the following: see Listing 4.

It seems that there is a support for Ralink-based cards. But apparently, our new card is a bit *new*. But what exactly cards are supported in this version of FreeBSD?

Yes, it's true. The only cards that can be initialized properly for this FreeBSD release, are based on chips that identified as RT2560 (`ID = 0x0201`). Compare this string with `pciconf` output: see Listing 6.

Should we give up? Don't panic!

Convert Windows XP driver and use NDISulator

We know that FreeBSD and Linux share the same capability – to load binary drivers from

Windows XP by means of designed and implemented *Network Driver Interface Specification* (NDIS). So it might be a magic stick for our need.

In Linux it is known as `ndiswrapper`, while in FreeBSD it appears as `NDISulator`. Being first introduced by Bill Paul in FreeBSD 5.3.

So, there are two ways to generate a kernel driver for FreeBSD from a binary PE-driver for Windows XP. First one – is to use `ndiscvt` (old method), or `ndisgen` (for FreeBSD versions 6.0 and higher).

Before actual conversion process we need to download Windows XP drivers from Ralink site – section *Support-*

Listing 8. It is possible to use `ndiscvt` utility as well

```
# cd /usr/src/sys/modules/if_ndis
# cp ~/driver_ralink/*INF ./
# cp ~/driver_ralink/*sys ./
# ndiscvt -i NetRt61G.INF -s rt61.sys -o ndis_driver_data.h
# make
```

Listing 9. Windows XP converted module is detected as `ndis0` interface

```
# kldload ./rt61_sys.ko
# dmesg | grep ndis
ndis0: <D-Link Wireless G DWA-510 Desktop Adapter> mem 0xe5000000-
          0xe5007fff irq 5 at device 10.0 on pci1
ndis0: NDIS API version: 5.0
ndis0: Ethernet address: 00:21:91:22:9f:20
```

Listing 10. Available working modes via NDIS interface

```
# ifconfig -m ndis0

supported media:
media OFDM/48Mbps mode autoselect mediaopt adhoc
media OFDM/48Mbps mode autoselect
media OFDM/24Mbps mode autoselect mediaopt adhoc
media OFDM/24Mbps mode autoselect
media OFDM/12Mbps mode autoselect mediaopt adhoc
media OFDM/12Mbps mode autoselect
```

Listing 11. The native backported driver is successfully loaded into a kernel space

```
ral0: <Ralink Technology RT2561> mem 0xe5000000-0xe5007fff irq 5
          at device 10.0 on pci1
ral0: MAC/BBP RT2561C, RF RT2527
ral0: Ethernet address: 00:21:91:22:9f:20
```

>Windows. We are only interested in archive for PCI/mPCI/CB (RT256x/RT266x).

Okay, we have download and unpacked the archive. Let's get started with *cooking* the driver. Run `ndisgen`.

We step through a step 3) and finally the kernel driver for FreeBSD is here. Alternatively, we could use the second way, i.e. converted using `ndiscvt` utility (see Listing 8).

Back to our console – we got `if_ndis.ko`, and we need to reboot a system in order for this kernel driver being loaded by `ndis-module`.

But first, let's load it into a memory and see whether the card works.

Looks like our card is detected properly at last. Now we can list modes list that can be applied to this WiFi NIC.

As we can see, the mode `AccessPoint` is missing. Although it isn't listed we can force it to be used – run `ifconfig` command as follows:

```
# ifconfig ndis0 ssid my_net media OFDM/48Mbps mode 11g
mediaopt hostap up
```

Listing 12. We have all modes supported by native driver

```
# ifconfig -m ral0
ral0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet 10.10.1.1 netmask 0xffffffff broadcast 10.10.1.255
ether 00:21:91:22:9f:20
media: IEEE 802.11 Wireless Ethernet OFDM/48Mbps mode 11g
      <hostap>
status: associated
supported media:
  media OFDM/54Mbps mode autoselect mediaopt monitor
  media OFDM/54Mbps mode autoselect mediaopt hostap
  media OFDM/54Mbps mode autoselect mediaopt adhoc
  media OFDM/54Mbps mode autoselect
```

Listing 13. xxxxxxxx

```
# ifconfig ral0
ral0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
inet 10.10.10.1 netmask 0xff000000 broadcast 10.255.255.255
ether 00:21:91:22:9f:20
media: IEEE 802.11 Wireless Ethernet OFDM/48Mbps mode 11g
      <hostap>
status: associated
ssid my_net channel 1 bssid 00:21:91:22:9f:20
authmode OPEN privacy OFF txpowmax 100 bmiss 7 protmode CTS
dtimperiod 1 bintval 100
```

Here we set our network name as `my_net`, chose to use high-speed connection mode – substring `media OFDM/48Mbps mode 11g`, and as a last option we force network interface to be used as `AccessPoint` – parameter `hostap`.

Well, that's pretty a long way. Praps the easy way is to install a modern version of FreeBSD, because we know that this D-Link DWA-510 card is supported there? Alas, you can't always change a working and stable environment onto something else, even if you quite sure that some subsystems are really outdated.

Is there a backport driver?

Anyway, we can continue working with `ndis0` interface without any problem. But there exist a little hope – it is known fact that the structure of the drivers, and the kernel itself does not change drastically with near releases, for instance 6.0 and 7.0.

So there is a little chance that someone made a backport of this driver from FreeBSD 7 to FreeBSD 6. Let's ask Google.

Indeed, after a long search we could find a patch [4]. Continue? Sure!

Copy it to `/usr/src/sys` location and apply the patch:

```
# patch -p0 <if_ral.diff
```

Now we need to compile `ral-driver`.

```
# cd /usr/src/sys/modules
# make ral
```

Driver is ready. By default this `ral-driver` is included into a kernel. That's why we need to comment it in a kernel configuration, and afterthat recompile kernel. Only after you have installed a kernel into `/boot` directory you can test driver with `kldload` command:

```
# kldload if_ral
```

Tracing the logs give us the following message: see Listing 11.

That's very nice, because we have now the full range of supported modes: see Listing 12.

So we can easily start the network interface `ral0`:

```
# ifconfig ral0 ssid my_net media OFDM/48Mbps
mode 11g mediaopt hostap 10.10.10.1 up
```


Visit our website

On the 'Net

- <http://www.thejemreport.com/content/view/293/> [1]
- <http://brainstorm.name/archives/33> [2]
- <http://www.freebsd.org/doc/en/books/handbook/config-network-setup.html> [3]
- http://samodelkin.net/~fjoe/if_ral.diff [4]

Aloha! The wireless network is here! (see Listing 13)

Before actual reboot we need to include the following string into `/boot/loader.conf`:

```
if_ral_load = "YES"
```

And apply several security features, for example, start daemons that will perform WEP- or WPA2-encryption.

Conclusion

Whilst we choose FreeBSD for its known stability and performance, there are chances that not all hardware devices are supported. And you'll be facing a problem – whether to stay with it, kick an upgrade process, or simply move onto another operating system. Undoubtedly, the evolution even with operating systems is great. But sometimes, you have no ability to change a thing – and you will need to figure out how to run unmaintained or even closed-source drivers. In this case, using the drivers from Windows XP with NDIS emulator can be a solution.

ANTON BORISOV

The very first Anton's experience with UNIX was FreeBSD. It was TWM, wget and Netscape Communicator. Many things have changed greatly since then, but a true simplicity remained unchanged – The Power to Serve. That's why the author prefers to delegate several network functions to FreeBSD)

You will find here:

➤ materials for articles-listings, additional documentation, tools

➤ the most interesting articles to download

➤ current information on the upcoming issue

www.bsdmag.org

I.T. certifications

and the value I got in it

After graduating college, I have created an account for an online resume publishing site.

Upon creating my online resume, I stumbled on a field asking for IT certifications. At that point, I have nothing to type in that field. I wasn't even aware that I.T. certification(s) would be something to put on a resume. So I left the field blank and completed my online resume.

A couple of years passed after my resume has been uploaded, I haven't got any emails or calls from employers using the online resume publishing site. I think there's something missing. I need to stand out and my resume should be browsed by potential employers for consideration in their job postings.

I have searched and looked at I.T. certifications from different vendors and technologies and I decided I would want to get one. Since the company I worked for uses different distributions of Linux, a certification in Linux must be the first I should get. I have used many resources studying the Linux systems, from online and printed materials to blogs and tutorials. I also used online practice tests to see my familiarity and mastery of the topics. When I was ready, I have signed up for LPI 101 examination and I passed it. Then I took LPI 102 and passed it too. This time, I was awarded with the LPIC-1 certification.

Upon receiving my certificate, I immediately updated my online resume to put my LPIC-1 in the IT certification field. After a few days, I have received a couple of emails from the site, which contains a list of employers viewing my resume. Indeed my IT certification caught the attention of employers and that I am now gaining value.

I wasn't satisfied with one certification and I went on to take SCJP from Sun Microsystems. I passed the examination and updated my online resume. As

expected, I received emails about job posting. I also now get a lot of phone calls from employers saying that *We have viewed your online resume, would you like to consider an interview for the position.....?* This has been very fulfilling for me.

Having a certification is very rewarding personally and professionally. It is some form of self-accomplishment. First of all, you learn a lot of things by studying, practicing, and making your way through the exam objectives. I for myself, learned a lot from the preparations/reviews I did for my exams. The things I learned were not day-to-day topics. Instead, they were advanced topics ranging from the internals, concepts, and applications. I was a Linux user before my certification, and I became a Linux *Power* user after I achieved it.

The skills I learned from LPIC-1 were my very foundations for studying and using the FreeBSD operating system. Although Linux and FreeBSD have their differences, they have something in common, and that is their UNIX roots (made to act, and based on UNIX, respectively).

I read new study materials from time to time, as to keep my skills *fresh*. I know for a fact that one could get *rusty* if one does not use the skills gained. So being certified in one technology does not mean you are a *master* of that particular technology. One should update his/her skills by reviewing the topics and studying the advancements in that technology.

I'm looking forward to take the BSDA examination next. But according to them, BSDA is available at events and other conferences as for the time being, they have not tied up with Prometric and VUE for examination delivery. I'm looking forward to make my skills and knowledge in FreeBSD go deeper and improve. And I

hope one day, I could take the BSDA examination and pass it.

Certification alone is NOT enough (my personal opinion) to be productive and competitive. In today's highly competitive market, you have to be highly skilled and experienced. Having certification(s) does not guarantee you on landing on a good high paying job, but it makes your chance higher than other job seekers. In my point of view, I.T. certification(s), proven skills, and experience are the pieces that will give you the *edge* in today's job market.

JOSHUA EBARVIA

Joshua Ebarvia is a java programmer, systems administrator and college lecturer. His passion is working and using operating systems specially UNIX-based and UNIX-cloned systems. You can reach him at joshua.ebarvia@gmail.com

a d v e r t i s e m e n t

RootBSD

PREMIERE VPS HOSTING

Latest FreeBSD

Full Root Access

Starting at \$20/mo

VPS and Dedicated

Multiple Datacenter Locations

Friendly, Knowledgeable Support Staff

WWW.ROOTBSD.NET

MAGAZINE

BSD

In the next issue:

- Commissioning FreeBSD with the Drupal Content Management Framework – Part 2
- FreeBSD applications
- and Other !

Next issue is coming in November!

Customize Your Server Flavors

Made to Order
Server Systems

High-Density
Storage Solutions

Intel® Xeon®
5600/5500 Series
Processors

Open Source
Configurations



Served **Exactly** How You Like!

Tired of being able to choose from only **chocolate**, **strawberry**, or **vanilla**? At iXsystems, we understand your need for custom-made servers.

“Open Source Hardware Design” is the iXsystems trademark. iXsystems provides an assortment of pre-configured servers and storage solutions, but our true pride rests on our ability to customize our products to meet your specific tastes and needs. iXsystems mixes in the raw power of Intel® Xeon® 5600/5500 Series Processors for a truly delicious treat. Our Professional Enterprise Service Level packages and desktop support offering also enables us to ensure you get the most from your FreeBSD® and PC-BSD® systems, adding the perfect toppings to your order.

Call iXsystems toll free or visit our website today!

+1-800-820-BSDi | www.iXsystems.com



Powerful.
Intelligent.