BSD MAGAZINE

FOR NOVICE AND ADVANCED USERS

FREEBSD: GET UP-TO-DATE

VOL.5 NO.01
ISSUE 01/2012(30)
1898-9144

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

✓ Increased Performance ✓ Impressive Energy Savings

TrueNAS™ Storage Appliance: You are the Cloud

With a rock-solid FreeBSD® base, Zettabyte File System support, and a powerful Web GUI, TrueNAS™ pairs easy-to-manage software with world-class hardware for an unbeatable storage solution.

Expansion Shelves Available

TrueNAS™ 2U System

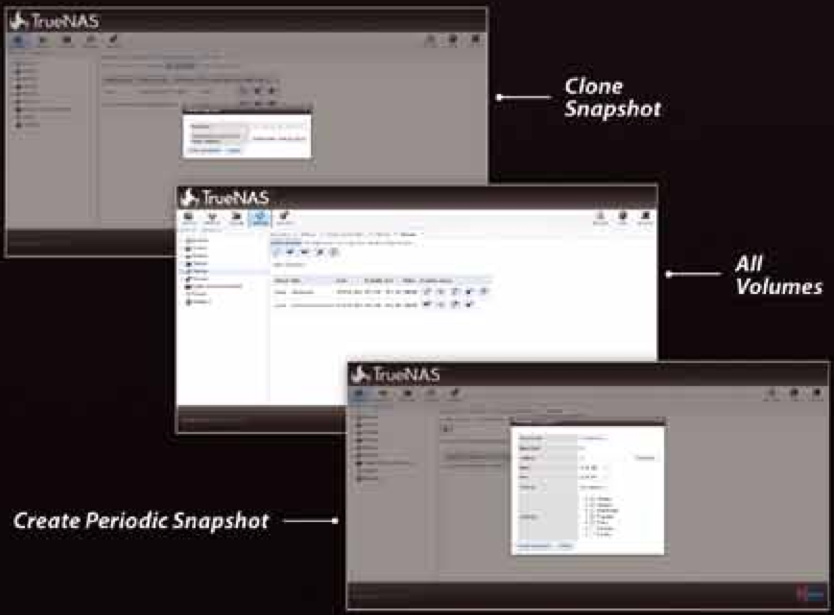
TrueNAS™ 4U System

FreeNAS 8 POWERED

Storage. Speed. Stability.

In order to achieve maximum performance, the TrueNAS™ 2U and 4U Systems, equipped with the Intel® Xeon® Processor 5600 Series, support Fusion-io's Flash Memory Cards and 10GbE Network Cards. Titan TrueNAS™ 2U and 4U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ Systems offer excellent capacity at an affordable price.

For more information on the **TrueNAS™ 2U** and **TrueNAS™ 4U**, or to request a quote, visit: **http://www.iXsystems.com/TrueNAS**.



TrueNAS™ 2U KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 12 Hot-Swap Drive Bays - Up to 36TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to triple parity
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

TrueNAS™ 4U KEY FEATURES

- Supports One or Two Quad-Core or Six-Core, Intel® Xeon® Processor 5600 Series
- 24 or 36 Hot-Swap Drive Bays - Up to 108TB of Data Storage Capacity*
- Periodic Snapshots Feature Allows You to Restore Data from a Previously Generated Snapshot
- Remote Replication Allows You to Copy a Snapshot to an Offsite Server, for Maximum Data Security
- Software RAID-Z with up to triple parity
- 2 x 1GbE Network Interface (Onboard) + Up to 4 Additional 1GbE Ports or Single/Dual Port 10GbE Network Cards

JBOD expansion is available on the 2U and 4U Systems

** 2.5" drive options available; please consult with your Account Manager*

Powerful. Intelligent.

Call iXsystems toll free or visit our website today!

1-855-GREP-4-IX | www.iXsystems.com

Dear Readers,

The New Year has come. People are now full of dreams and expectations... again. It happens year after year, so it can be sad that there is nothing to talk about... However, each January we know better what are our goals and how to achieve them. We know more about what we like and what we don't. In this special month when all have minds filed with new ideas, we would like to present you 30th issue of BSD Magazine. We hope you will find it inspiring and entertaining. This time you will have a chance to update your knowledge about FreeBSD.

We start with What's New in FreeBSD 9.0 by Dru Lavigne. Next, the section How To follows. You will have an opportunity to learn more about Puppet on FreeBSD and how to deploy servers by using it. Michael Shirk will discuss the Snort's ability to integrate with ipfw that allow for inline IPS mode on FreeBSD. Toby Richards will show you how to build Captive Portal using OpenBSD's Packet Filter.

For admins this time Paul McMath prepared the piece about malloc(9): The Kernel's General Purpose Memory Allocator. Don't miss that! In Tips & Tricks Slawomir Wojtczak deals with an ungrateful topic of keeping both FreeBSD's base system and installed packages up-to-date. Since the new release is out it's an obligatory position on January list of articles. In section Security you will find part 2 of Anatomy of a FreeBSD Compromise by Rob Somerville. This time he will examine some of the common techniques used to gain control and what we can do to mitigate the risks.

Let's talk will give you a chance to learn something about PostgreSQL and PGDay in Italy. I hope you will find it inspiring somehow... Don't forget to send us your feedback!

Wish you all your dreams come true in 2012!

Patrycja Przybyłowicz
& BSD Team

MAGAZINE BSD

Editor in Chief:
Patrycja Przybyłowicz
patrycja.przybylowicz@software.com.pl

Contributing:
Dru Lavigne, Toby Richards, Rob Sommerville, Michael Shirk, Edward Tan, Sławomir Wojtczak, Paul McMath, Luca Ferrari

Proofreaders:
Robert Nowak, Zander Hill, Sander Reiche, Christopher J. Umina

Special Thanks:
Denise Ebery

Art Director:
Ireneusz Pogroszewski

DTP:
Ireneusz Pogroszewski

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

CEO:
Ewa Dudzic
ewa.dudzic@software.com.pl

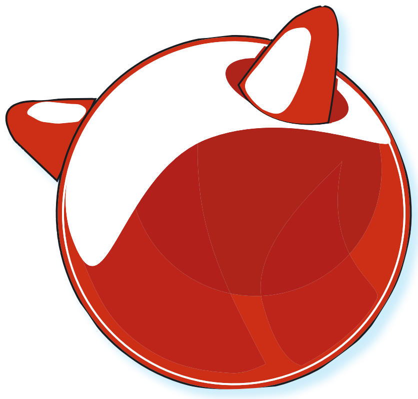
Production Director:
Andrzej Kuca
andrzej.kuca@software.com.pl

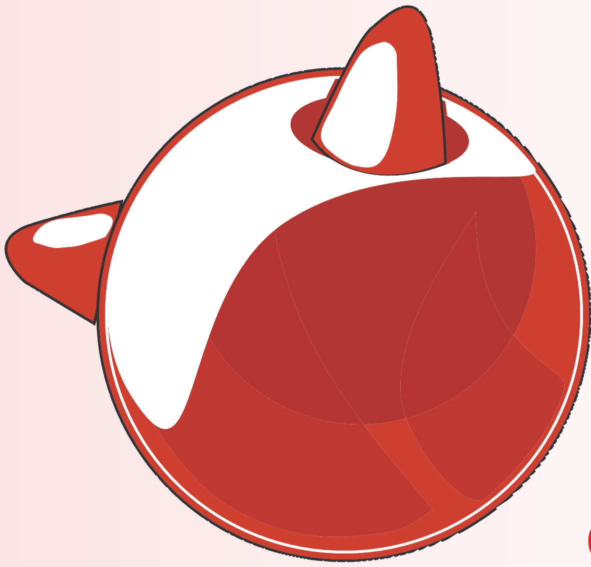
Executive Ad Consultant:
Ewa Dudzic
ewa.dudzic@software.com.pl

Advertising Sales:
Patrycja Przybyłowicz
patrycja.przybylowicz@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*





What's New in FreeBSD 9.0

This article provides an overview of some of the new features available in FreeBSD 9.0.

FreeBSD 9.0-RELEASE introduces many new features which benefit FreeBSD users, application developers, and companies that use or base their products on FreeBSD. This article provides an overview of some of these features, including references to additional information. It does not list all of the new features as the FreeBSD 9.0 Detailed Release Notes, available from freebsd.org, contains a summary of all the changes introduced in 9.0.

This article discusses features in the following categories: security, compilers and testing frameworks, filesystems and storage, networking, and miscellaneous.

Security

Capsicum Framework

Capsicum is a lightweight framework which extends a POSIX UNIX kernel to support new security capabilities and adds a userland sandbox API. It was originally developed as a collaboration between the University of Cambridge Computer Laboratory and Google, sponsored by a grant from Google, with FreeBSD as the prototype platform and Chromium as the prototype application. FreeBSD 9.0 provides kernel support as an experimental feature for researchers and early adopters. Application support will follow in a later FreeBSD release and there are plans to provide some initial Capsicum-protected applications in FreeBSD 9.1.

Traditional access control frameworks are designed to protect users from each other through the use of permissions and mandatory access control policies. However, they cannot protect the user when an application, such as a web browser, processes many potentially malicious inputs, such as HTML, scripting languages, and untrusted images. Capsicum provides application developers fine-grained control over files and network sockets to provide privilege separation within an application, with minimal code changes. In other words, it provides application compartmentalisation, allowing the application itself to provide many different sandboxes to contain its various elements. As an example, each tab in the Chromium browser has its own sandbox; it is also possible to contain each image in its own sandbox. Creating sandboxes under Capsicum does not require privilege, a key problem with current UNIX sandbox approaches.

As an example, the insecure tcpdump application can be sandboxed with Capsicum in about 10 lines of code and the Chromium web browser can be sandboxed in about 100 lines of code. capsicum(4) provides an overview of the available system calls. More information, including links to technical publications, projects, and a mailing list, can be found at the Capsicum website: *http://www.cl.cam.ac.uk/research/security/capsicum/*.

Resource Limits

rctl(8) has been added to the system, allowing the user to display the current resource limits and to define what action will occur when a process exceeds it limits. Resource rules can be applied to processes, users, login classes, or jails.

The racct API tracks per-process, per-jail, per-loginclass, and per-user resource accounting information. More information about resource limits and rctl can be found at *http://wiki.freebsd.org/Hierarchical_Resource_Limits*.

Compilers and Testing Frameworks

LLVM Compiler Infrastructure

LLVM is a BSD-licensed compiler infrastructure with similar capabilities to the GPL3-licensed GCC compiler collection. Clang is the C, C++, Objective C, and Objective C++ front-end to LLVM and provides an alternative programming environment for developers and companies who prefer to use a BSD-licensed toolchain.

In addition to being BSD-licensed, Clang improves developer productivity with significantly improved error messages and a static code analyzer. The compiler is easily extendable to support research on new language features or code instrumentation.

Beginning with FreeBSD 9.0, the FreeBSD kernel and world can be compiled using Clang on most of the supported architectures. Work is ongoing to migrate the ports infrastructure so that any port can also be compiled with Clang. Details about architecture support, link time optimizations, automatic test generation, and links to additional resources can be found at *http://wiki.freebsd.org/BuildingFreeBSDWithClang*. More information about Clang can be found at *http://clang.llvm.org/* and more information about LLVM is available from *http://www.llvm.org/*.

A video of Brooks Davis describing how the FreeBSD Project has been actively working to incorporate tools from the LLVM project into the base system is available at *http://www.youtube.com/watch?v=yVaNAm8jR_U*. You can follow the status of the ports infrastructure with regards to Clang at *http://wiki.freebsd.org/PortsAndClang*.

Userland Ttrace

DTrace is a general purpose, lightweight tracing framework that allows administrators, developers, and users to investigate causes of system failure or performance bottlenecks. FreeBSD introduced kernel-level DTrace support in FreeBSD 8.0. The addition of user-level DTrace suppport in 9.0 allows inspection of userland software and its correlation with the kernel, thus providing a much better picture of what exactly is going on behind the scenes.

http://wiki.freebsd.org/DTrace provides examples for using both kernel- and user-level DTrace on FreeBSD, as well as links to other DTrace resources.

Filesystems and Storage

Highly Avaliable Storage (HAST)

The Highly Available Storage framework allows for synchronous, block-level replication of any storage media across several physically separated machines connected by a TCP/IP network. HAST can be understood as a network-based mirror, similar to Linux DRBD. When combined with FreeBSD's carp(4), HAST makes it possible to build a highly available storage cluster that is resistant to hardware failures.

HAST is file system and application independent and can be combined with any existing GEOM class. In case of a primary node failure, the cluster will automatically switch to the secondary node, check and mount the UFS file system or import the ZFS pool, and continue to work without missing a single bit of data.

The FreeBSD Handbook describes how to configure HAST: *http://www.freebsd.org/doc/handbook/disks-hast.html*.

SU+J

Journaled softupdates for UFS is now the default filesystem type. It adds a light version of journaling to soft updates as described in this technical paper: *http://www.mckusick.com/BSDCan/bsdcan2010.pdf*. This significantly reduces boot time after an improper shutdown as a background fsck only needs to be run if there is a corruption of the journal log.

ZFSv28

FreeBSD 9.0 ships with ZFSv28. This version of ZFS adds the following features:

- *deduplication*: the process of eliminating duplicate copies of data. When enabled on datasets with duplicate data (for example, virtual images or jails), deduplication saves space and increases performance because less data is written and stored.
- *triple parity RAIDZ*: RAIDZ3 offers three parity drives and can operate in degraded mode if up to three drives fail with no restrictions on which drives can fail.
- *zfs diff*: command which describes which file system level changes have occurred between two snapshots.
- *zpool split*: allows an administrator to extract one disk from each mirrored top-level vdev and use them to

create a new pool with an exact copy of the data. The new pool can then be imported on any machine.

- *snapshot holds*: permit users or applications to place holds on snapshots to prevent them from being deleted.
- *zpool import -F*: allows the administrator to rewind a corrupted pool to an earlier transaction group.
- the ability to import zpool as read-only.

Generic GEOM I/O Scheduler Framework

This framework supports scheduling disk I/O requests in a device independent manner in order to support multiple disk I/O schedulers to be used on different I/O providers. The framework provides a couple of sample scheduling algorithms that use the framework and implements two forms of anticipatory scheduling.

The ability to create different I/O schedulers allows users to select the I/O scheduler best suited to the task. This can increase responsiveness in certain kinds of I/O workloads, such as a mix of sequential and random I/O. Examples of how to use the provided schedulers can be found at *http://svnweb.freebsd.org/base/head/sys/geom/sched/README?view=markup&pathrev=206497*.

Changes to CAM and AHCI SATA

The new ATA/SATA driver supports AHCI-compliant hardware, port multipliers, and NCQ (tagged queueing) for increased performance on modern SATA drives. Performance has been greatly increased, larger data transfers are supported, and hot-plugging support is much improved. ATA/SATA drives can now can be enumerated and manipulated via camcontrol(8), just like SCSI drives.

The cam(4) subsystem is now modularized and the addition of the ATA/SATA modules allows the CAM subsystem to grow into a framework for arbitrary transports and protocols. It also allows drivers to be written to support discrete hardware without jeopardizing the stability of non-related hardware.

Changes to Event Timer Infrastructure

The new event timers infrastructure provides unified APIs for writing event timer drivers and for choosing the best possible drivers by machine independent code. It provides support for both per-CPU and global timers in periodic and one-shot modes for the i386 and amd64 architectures.

To improve performance in virtual machines and power usage in laptops, dynamic tick mode is enabled by default, replacing the periodic hardware timer interrupt ticking with one-shot variable-time ticks. This saves CPU time which would otherwise be spent handling timer interrupts which have no work assigned to them. Tickless mode can be turned off by setting the sysctl value of kern.eventtimer.periodic to 1. Technical details about dynamic tick mode can be found at *http://permalink.gmane.org/gmane.os.freebsd.architecture/13276*.

Networking

Five New TCP Congestion Control Algorithms

The Centre for Advanced Internet Architectures at Swinburne University of Technology, with the support of the Cisco University Research Program Fund at Community Foundation Silicon Valley and the FreeBSD Foundation, delivered enhancements to FreeBSD's TCP stack in order to support newer congestion control algorithms. These enhancements included a modular framework for adding future algorithms as well as new modular implementations of the H-TCP, CUBIC, Vegas, HD, and CHD algorithms.

Each congestion control algorithm is implemented as a loadable kernel module. Algorithms can be selected to suit the application/network characteristics and requirements of the host's installation. The modular framework makes it much easier for developers to implement new algorithms, allowing FreeBSD's TCP stack to be at the forefront of advancements in this area, while still maintaining the stability of its network stack.

Links to technical papers regarding the framework and algorithms can be found at *http://caia.swin.edu.au/freebsd/5cc/*.

"IPv6-Only"

FreeBSD has been on the leading edge of IPv6 development ever since FreeBSD 4.0 was released in 2000 with the KAME reference implementation of IPv4/IPv6 networking support. In addition, the FreeBSD Project has been serving releases from IPv6-enabled servers for more than 8 years and FreeBSD's website, mailing lists, and developer infrastructure have been IPv6-enabled since 2007.

Beginning with FreeBSD 9.0, no-IPv4 snapshots of FreeBSD are available. By completely decoupling IPv6 from IPv4, early adopters and developers can determine if "IPv6-ready" applications really are ready for IPv6 or if bugs were hidden due to the ability to fallback on IPv4. Providing an implementation of an IPv6-only kernel without IPv4 support provides the FreeBSD Project with the ability to test and fix such regressions while encouraging other software developers to improve their code for true IPv6 readiness. More information about no-IPv4 versions of FreeBSD is available from *http://www.freebsd.org/ipv6/*.

To support IPv6-only, rtadvd(8) and rtsold(8) were completely overhauled to support RFC 6106. rtsold can now update `/etc/resolv.conf` using the openresolv DNS management framework (*http://roy.marples.name/projects/openresolv*). An optional kernel module is available to provide *Secure Neighbor Discovery* protocol (SeND) support; SeND is described in RFC 3971.

Continuing earlier efforts, more global options can now be controlled on a per-interface base, such as the ability to accept router advertisements on one interface while still forwarding. This is needed to effectively run FreeBSD as an IPv6 CPE device. The single `/etc/rc.conf` option `ipv6_cpe_wanif` will correctly set all sysctls and interface options to make creating a CPE as easy as possible.

High Performance SSH (HPN-SSH)

OpenSSH is network performance limited by statically defined internal flow control buffers. These buffers often end up acting as a bottleneck for network throughput of SCP, especially on long and high bandwith network links. HPN-SSH adds support for dynamically adjusted buffers to allow the full use of the bandwidth of long fat pipes such as 100Mbps or greater, trans-oceanic, or trans-continental links. Bandwidth-delay products up to 64MB are also supported. This implementation includes a multithreaded cipher implementation which makes such bandwidth sustainable on the CPU side.

HPN is enabled by default in FreeBSD 9.0's sshd and several HPN options have been added to `/etc/ssh/sshd_config`. These options, as well as some performance tips, are described in *http://svnweb.freebsd.org/base/head/crypto/openssh/README.hpn?revision=224638&view=markup*.

Miscellaneous

Several other features are also worth mentioning:

- large-scale SMP support for systems with more than 32 CPUs. Previously, the kernel structures were unable to account for such a large number of CPUs so this method implements extensible CPU accounting. Yahoo! provided systems for testing these changes.
- improved USB 3.0 support.
- the default NFS client and nfsd(8) now support NFSv4. Backwards compatibility for older NFS clients is provided with the *oldnfs* mount type.
- a new kernel-mode NFS lock manager has been added, improving performance and behavior of NFS locking. A new clear_locks(8) command has been added to clear locks held on behalf of an NFS client.
- sysinstall has been replaced with bsdinstall. Its features are described at *http://wiki.freebsd.org/BSDInstall* and its usage is detailed in the FreeBSD Handbook: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/bsdinstall.html*.
- the kernel now supports a new textdump(4) format of kernel dumps. A textdump provides higher-level information via mechanically generated/extracted debugging output, rather than a simple memory dump. This facility can be used to generate brief kernel bug reports that are rich in debugging information, but are not dependent on kernel symbol tables or precisely synchronized source code.
- FreeBSD 9.0 can be installed on the Sony Playstation 3 using the instructions at *http://people.freebsd.org/~nwhitehorn/ps3/README*.
- call and return rule actions were added to ipfw(8): *http://svnweb.freebsd.org/base?view=revision&revision=223666*.

Conclusion

With the release of FreeBSD 9.0, the FreeBSD Project continues to innovate in the areas of security, compilers, filesystems, and networking. You can find out more information about the FreeBSD Project and download FreeBSD 9.0 from freebsd.org.

DRU LAVIGNE

Dru Lavigne is author of BSD Hacks, The Best of FreeBSD Basics, and The Definitive Guide to PC-BSD. As Director of Community Development for the PC-BSD Project, she leads the documentation team, assists new users, helps to find and fix bugs, and reaches out to the community to discover their needs. She is the former Managing Editor of the Open Source Business Resource, a free monthly publication covering open source and the commercialization of open source assets. She is founder and current Chair of the BSD Certification Group Inc., a non-profit organization with a mission to create the standard for certifying BSD system administrators, and serves on the Board of the FreeBSD Foundation.

Home Brew Captive Portal With OpenBSD

Have you ever used a public wireless network that has a splash screen such that you have to agree to certain terms before going to the Internet?

It's called a *captive portal*, and we're going to build one of those using OpenBSD's *Packet Filter* (pf). FreeBSD can also use Packet Filter, so these instructions should work for that OS, but I've not tested it. There are several pre-built captive portal solutions for FreeBSD. Captive portals for OpenBSD are more rare, so this is something of a home-brew solution.

When I built this project, I used 172.16.0.0/24 as my captive network. Then there is 192.168.0.0/24 which is a non-captive network with a DSL modem/router at 192.168.0.254. The server has two NIC's: xl0 (192.168.0.1) and em0 (172.16.0.1). See illustration on Figure 1.

As you know, I run OpenBSD with Nginx. By default, Apache is jailed with chroot on OpenBSD. The technique that I'm going to describe won't work if your web server is jailed because we're going to have PHP call for `/sbin/pfctl`. The first thing we need to do is to allow that. I am using a user called `_php` to run PHP in Fast CGI mode for Nginx. If you're using Apache with `mod_php`, then substitute the user that httpd runs as. Use *visudo* to add the following line to your sudoers file:

```
_php ALL=(ALL) NOPASSWD: /sbin/pfctl
```

We'll want our server to provide DHCP for the captive network (remember that it's on `em0`).

```
# echo 'dhcpd_flags="em0"' >>/etc/rc.conf.local

# touch /var/db/dhcpd.leases
```

Now we edit `/etc/dhcpd.conf`. Here's what mine looks like:

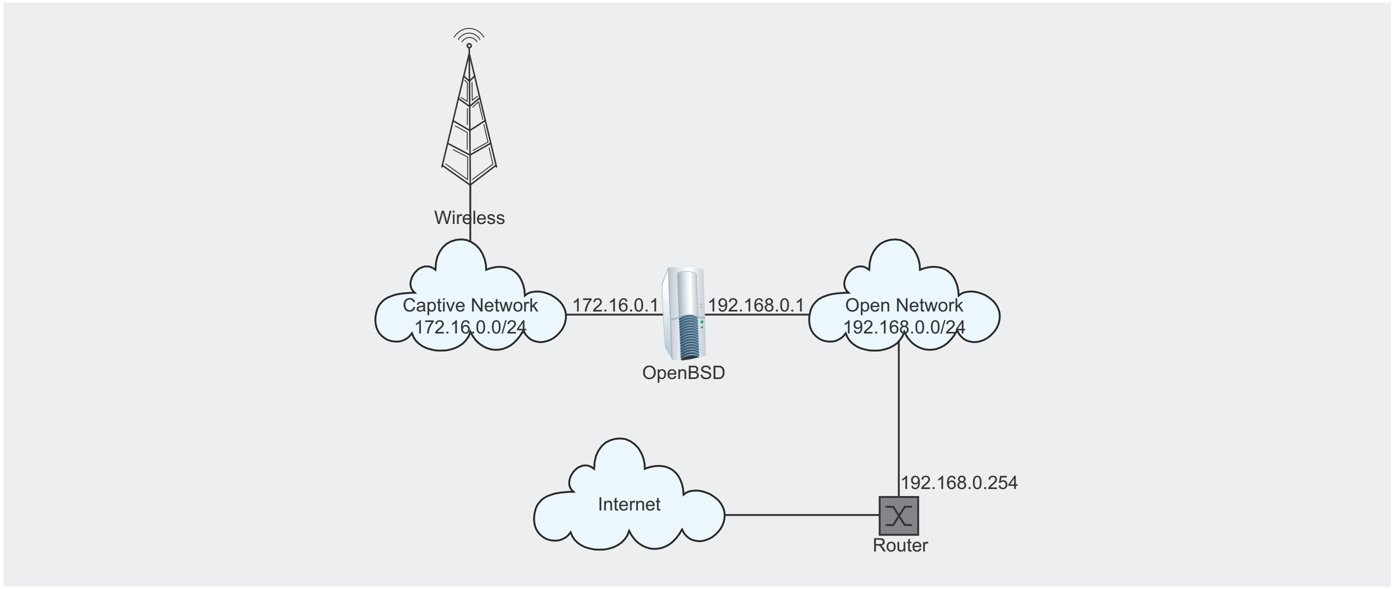


Figure 1. *Network Diagram*

```
option  domain-name-servers 8.8.8.8;

subnet 172.16.0.0 netmask 255.255.255.0 {

option routers 172.16.0.1;

range 172.16.0.100 172.16.0.199;

}
```

Launch the DHCP daemon:

```
# dhcpd em0
```

We want IP forwarding, so edit `/etc/sysctl.conf` to uncomment the following line (I'm not taking IPv6 into account, but if you use IPv6 then you'll know how to enable that as well):

Listing 1. *pf.conf*

```
set skip on lo

pass # to keep state

pass out quick on xl0 proto udp from any to any port 53 nat-to xl0

pass in quick on em0 from <eula> to any

pass out quick on xl0 from <eula> to any nat-to xl0

pass in on em0 proto tcp from any to any port 80 rdr-to 172.16.0.1 port 80
```

Listing 2. *index.php*

```
<html><body>

<?php

$salt="somesalt";

$clientip=$_SERVER['REMOTE_ADDR'];

$clienthash=hash('sha256', $salt . $clientip);

?>

This is my captive portal. Press the button to accept the terms of this network.<br>

<form action="auth.php" method="post">

<input type="hidden" name="myhash" value="<?php echo $clienthash ?>" />

<input type="hidden" name="myip" value="<?php echo $clientip ?>" />

<input type="submit" />

</form></body></html>
```

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

To turn on IP forwarding without rebooting so the system reloads `/etc/sysctl.conf` we issue this command:

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

It's time to edit `/etc/pf.conf`. I'm going to use a table that I'll call *eula* to track the IP addresses of users who have clicked the button to accept the terms of using the network. The line that references UDP:53 allows everybody to resolve DNS names. That will prevent users from getting errors before they click through the portal (Listing 1). Be sure to reload your firewall rules with:

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

That last line is the magic. Anyone whose IP address is not in the *eula* table will be redirected to the web server on the BSD box. As stated at the beginning of this article, it's already assumed that you have a working PHP enabled web server that isn't jailed.

Listing 3. *auth.php*

```
<html><body>

<?php

$salt="somesalt";

$clientip=$_SERVER['REMOTE_ADDR'];

$goodhash=hash('sha256', $salt . $clientip);



$myhash=$_POST['myhash'];

if ($myhash != $goodhash) { die(); }

exec ("sudo pfctl -t eula -T add " . $clientip);

?>

You may now use the Internet.

</body></html>
```

When a user tries to go to *http://somesite.com* then Packet Filter will redirect them to http://172.16.0.1 where our splash screen will live. What if someone tries to go to *http://somesite.com/somepage*? They'll be redirected to *http://172.16.0.1/somepage* which will yield a 404 error. I can't say how to fix this with Apache, but for Nginx, you can fix this with a single line in the `location /` block of `/etc/nginx/nginx.conf`:

```
try_files $uri $uri/ /index.php;
```

Now for some PHP voodoo. My examples are extremely simple. You will want to make your own fancy modifications for aesthetics, security, and code elegance. In this example, a salted hash of the client's IP address is our only security. First, index.php: Listing 2. Next, *auth.php*: Listing 3.

There is one last consideration. We probably want authorization for Internet use to expire. Eight hours seems like a good amount of time before an IP gets the splash page again. In seconds, that's 28800. Add the following to `/var/cron/tabs/root`:

```
* * * * * /sbin/pfctl -t eula -T expire 28800 > /dev/null
```

Every minute, cron will remove IP addresses from the *eula* table that are older than 8 hours. You've created your own captive portal. Now all that's left is for you to start your own coffee shop!

TOBY RICHARDS

Toby Richards has been a network administrator since 1997. Each article comes straight from the notes that he takes when doing a new project with *BSD. Toby recommends bsdvm.com for your hosting needs because they provide console access to your virtual machine.

www.linagora.com

LINAGORA

The leading french Open Source software company !

OBM.org	Open Source messaging & collaboration suite	OSSA OPEN SOURCE SOFTWARE ASSURANCE N° Vert 0 8000 LINUX	Support, assistance & technology watch
LinID LinPKi	Identity management/ Open Source security suite	PROFESSIONAL SERVICES	Training, Consulting, development & engineering and WebStudio
COCKTAIL	ERP Open Source		

```
linagora:$ adduser "YOU" | exec "community devellopement" |
make tablefootball | while [1] ; do more funstuffs; done
```

JOIN LINAGORA !

www.linagora.com/jobs

Puppet on FreeBSD

This article aims to jumpstart a system administrator on how to use Puppet (configuration management tool), to manage server's configurations, particularly on FreeBSD.

What you will learn…

- what is Puppet,
- how to deploy servers using Puppet,
- various scenario in server deployment using Puppet.

What you should know…

- basic network configuration in FreeBSD,
- basic knowledge FreeBSD port system.

Using *Puppet* to manage server's configurations yield these benefits:

- automated server installation
- mass deployment of changes to servers
- maintain server state consistency

Puppet can be use to automate software installation and configuration when deploying servers. This is particularly helpful when deploying many servers with similar service configurations (e.g. sudoers, ssh daemon, web services and others).

Puppet is also useful in situations where a script or program needs to be deployed to a group of servers. After the initial deployment, changes can also be push from the *Puppet* master to these group of servers with minimal effort.

Part of a system administrator's job is to make sure that the server state is in desirable form; for instance ssh daemon should always ask for public keys and password authentication should be disabled. A system administrator have to personalize configuration files and set hem up accordingly in all servers, and services (e.g. ssh, ntp, named ...) should always be up or should restarted to load the changes.

Puppet is written in Ruby and it is Apache licensed (since 2.7.6. previously it was GPL licensed). It is a fork from cfEngine, another powerful configuration management tool. *Puppet* is relatively easy to pickup and start using due to its declarative language.

Instead of reading through the typical lengthy language guide or tutorial to learn about *Puppet*, this article will go through the basic steps of how to setup the *Puppet* client/server model. Then we will cover common scenarios, to get an idea about how *Puppet* works.

The Picture

In this section, the FreeBSD servers involved in the *Puppet* client/server model should look (be installed) like this:

- the installation and configuration of *Puppet* on *master* and *agent*. This is just about SSL certificate signing between *Puppet* master and it's agent.
- ports installation using FreeBSD Ports system, to deploy consistent sudoers configuration and ssh daemon to *Puppet* agent. Then setup and configure Apache web server, automatically.

Listing 1. *Installation of Puppet*

```
# cd /usr/ports/sysutils/puppet
# make install clean
```

In order to achieve the above, we'll break the setup into the following parts:

Part I: setup Puppet master and agent

- installation of *Puppet* and other utilities
- setup Puppet's configuration files:
  - `/usr/local/etc/puppet/puppet.conf`
  - `/usr/local/etc/puppet/fileserver.conf`
  - `/usr/local/etc/puppet/manifests/site.pp`
  - `/usr/local/etc/puppet/manifests/classes/*.pp`
  - `/usr/local/etc/puppet/auth.conf`
  - `/etc/rc.conf`
- certificate signing of *Puppet* master and agent

Part II: various deployment scenarios

- use *Puppet* to setup agent's sudoers configuration
- setup and configure ssh daemon
- setup and configure Apache 2.2 web server

These servers will perform the following roles:

- *puppet-master.example.com* – this server will be the *Puppet* master. It will responsible to push configurations to the *Puppet* agent.
- *puppet-agent.example.com* – the *Puppet* agent. It will receive *Puppet* master's instruction and deploy it accordingly. In this case, deploy sudoer configurations, ssh daemon and Apache web server.

By now, you should have noticed that the server requires FQDN internet name. The *Puppet* master will be named as *puppet-master.example.com* and *puppet-agent.example.com*.

Listing 2. *Skeleton files creation*

```
# mkdir -p /usr/local/etc/puppet/manifests/classes
# mkdir -p /usr/local/etc/puppet/files
# touch /usr/local/etc/puppet/files/sudoers
# touch /usr/local/etc/puppet/files/sshd_config
# touch /usr/local/etc/puppet/files/httpd.conf
# touch /usr/local/etc/puppet/manifests/classes/
                    resource_group.pp
# touch /usr/local/etc/puppet/fileserver.conf
# touch /usr/local/etc/puppet/manifests/site.pp
# touch /usr/local/etc/puppet/manifests/classes/
                    resource_group.pp
```

Disclaimer: The configuration files included in this article are bare minimum to bring up *Puppet* master/agent and it's controlled services, they are might NOT be suitable for production. Use at your own risk!

How Puppet Works

Puppet uses the relationship of *master* and *agent* to control its client from the server. The *Puppet* master will push the instructions to the agent and the agent will then perform the instructions.

These actions are termed *resource*. *Resources* can be group into *classes* to form more manageable *functions*. These *resources* are then performed on the *node*, or agent.

There are various resource types, for example file resources, package resources, user resources and more.

Puppet behaviour:

- *Puppet* uses SSL to secure the communications between *master* and *agent*.
- *Puppet* master resource permission are defined in the file `/usr/local/etc/puppet/auth.conf`.
- *Puppet* master configuration file is `/usr/local/etc/puppet/puppet.conf`.
- *Puppet* agent do NOT need `/usr/local/etc/puppet/puppet.conf` to run properly.
- *Puppet* error messages are sorted into `/var/log/messages`.

The below tree structure shows how the files and directories needed for *Puppet* master to function:

```
/usr/local/etc/puppet
    ├── auth.conf
    ├── files
    │   ├── httpd.conf
    │   ├── sshd_config
    │   └── sudoers
    ├── fileserver.conf
    ├── manifests
    │   ├── classes
    │   │   ├── resource_group.pp
    │   │   └── something.pp
    │   └── site.pp
    └── puppet.conf
```

Part I: setup Puppet master and agent

Preparation of *puppet-master.example.com*: Listing 1. Setup and configuration: Listing 2. Configure the *Puppet* Master: Listing 3.

Change the following parameters `/usr/local/etc/puppet/puppet.conf`:

- `factsource = puppet://puppet-master.example.com/facts/`
- `pluginsource = puppet://puppet-master.example.com/plugins`

Create the following files with it's contents: Listing 4 and Listing 5.

Finally, start *Puppet* master to listen for certificate signing request so that it can push instructions to *Puppet* agent.

In Listing 5, various *Puppet* agent hostnames are defined and specify what resources are being push out. In this initial setup, there's nothing. We'll looking into this section again when fulfilling various scenarios. Preparation of *puppet-agent.example.com*: Listing 7-9.

Initiate a certificate signing session from the *Puppet* agent, to the *Puppet* master:

```
# puppet agent -v --server puppet-master.example.com --
                    waitforcert 60 --test
```

While this is happening, sign the certificate in *puppet-master.example.com*: Listing 10.

When the certificate is signed, the *Puppet* agent's certificate signing session will terminate. Wait for it to terminate itself.

Start *Puppet* agent in puppet-agent.example.com by adding the following to `/etc/rc.conf`: Listing 11.

Start *Puppet* agent so that it can poll instructions from *Puppet* master:

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

Now that the *Puppet master* and *agent* are more or less setup, do make sure that there's no error message happening in `/var/log/messages`.

Part II: Various Deployment Scenario

Scenario 1: using puppet to setup agent's sudoers configuration

In *puppet-master.example.com*, create a customized sudoer file, which eventually will be deployed to *Puppet*

Listing 3 *Configuration of Puppet Master*

```
# puppet master --genconfig > /usr/local/etc/puppet/
                    puppet.conf
# cp /usr/local/etc/puppet/auth.conf-dist /usr/local/
                    etc/puppet/auth.conf
```

Listing 4 */usr/local/etc/puppet/fileserver.conf*

```
[files]
    path /usr/local/etc/puppet/files
    allow *.example.com
```

Listing 5. */usr/local/etc/puppet/manifests/site.pp*

```
import "classes/*.pp"
filebucket { main: server => 'puppet-
                    master.example.com' }
File { backup => main }
Exec { path => "/usr/bin:/usr/sbin/:/bin:/sbin" }

node 'puppet-agent.example.com' { }
```

Listing 6. *Start the Puppet master*

```
# echo 'puppetmaster_enable="YES"' >> /etc/rc.conf
# /usr/local/etc/rc.d/puppetmaster start
```

Listing 7. *Installation of Puppet agent and some dependency*

```
# cd /usr/ports/sysutils/puppet
# make install clean
# cd /usr/ports/ports-mgmt/portupgrade
# make install clean
```

Listing 8. *Setup and configuration of Puppet agent*

```
# hostname puppet-agent.example.com
```

Listing 9. */usr/local/etc/puppet/auth.conf*

```
path /run
method save
allow puppet-master.example.com
```

Listing 10 *Signing the certificate in Puppet master*

```
# puppet cert --list --all
# puppet cert --sign puppet-agent.example.com
```

Listing 11. */etc/rc.conf*

```
puppet_enable="YES"
puppet_flags="-v --listen --server puppet-
                    master.example.com"
```

agent: Listing 12. Create a class, to group these *resources*: Listing 13.

Include this *resource* (in the class) in a *node* (*Puppet* agent) so that *Puppet* master can push it to the *Puppet* agent: Listing 14.

Activate this change by *kicking* the *Puppet* agent. It willthen retrieve the catalog and perform the necessary actions on the agent:

```
# puppet kick puppet-agent.example.com
```

Once *Puppet* agent refreshes these configurations, the customized sudoers file should be downloaded into `/usr/local/etc/sudoers` of the *puppet-agent.example.com*. Check the file `/usr/local/etc/sudoers` and see if it is identical to the one we create in *puppet-master.example.com*. This scenario is to demonstrate that *Puppet* is able to push configuration files from the *Puppet* master to it's agent.

In Listing 13, the resource type *file* is included in the class named *sudoers*. Whenever a node need to use this resource, we can use the syntax *include* in the class *sudoers*. This same resource can be use multiple times onto different *node*.

Also, the keyword *source* specifies the source of the file to be place in `/usr/local/etc/sudoers`. This URL should be absolute to the parameter `[files]` in Listing 4.

For more information on resource types, check out *http://docs.puppetlabs.com/references/stable/type.html*.

Scenario 2: setup and configure ssh daemon

Similarly, services and configuration files can be managed from *Puppet* master. This is how to deploy a customized ssh daemon with configuration file, and push it to *Puppet*

Listing 12. */usr/local/etc/puppet/files/sudoers*

```
root    ALL=(ALL) ALL
bob    ALL=(ALL) NOPASSWD: ALL
joe     ALL=(ALL) NOPASSWD: ALL
```

Listing 13. */usr/local/etc/puppet/manifests/classes/resource_group.pp*

```
class sudoers {
    file { "/usr/local/etc/sudoers":
        ensure  => file,
        owner   => root,
        group   => wheel,
        mode    => 440,
        source  => "puppet:///files/sudoers",
    }
}
```

Listing 14. */usr/local/etc/puppet/manifests/site.pp*

```
node 'puppet-agent.example.com' {
    include sudoers
}
```

Listing 15. */usr/local/etc/puppet/files/sshd_config*

```
PermitRootLogin no
StrictModes yes
PubkeyAuthentication yes
PermitEmptyPasswords no
```

Listing 16. */usr/local/etc/puppet/manifests/classes/resource_group.pp*

```
class sshd {

        owner       => root,
        group       => wheel,
        mode        => 0644,
        source      => "puppet:///files/sshd_config",
    }
    service { 'sshd_services':
        ensure      => running,
        name        => "sshd",
        enable      => true,
        hasrestart  => true,
        hasstatus   => true,
        subscribe   => File['/etc/ssh/sshd_config'],
    }
}
```

Listing 17. */usr/local/etc/puppet/manifests/site.pp*

```
node 'puppet-agent.example.com' {
    include sudoers
    include sshd
}
```

Listing 18. */usr/local/etc/puppet/files/httpd.conf*

```
ServerRoot "/usr/local"
Listen 80
LoadModule authz_host_module libexec/apache22/mod_authz_
                   host.so
LoadModule include_module libexec/apache22/mod_
                   include.so
LoadModule log_config_module libexec/apache22/mod_log_
                   config.so
LoadModule mime_module libexec/apache22/mod_mime.so
LoadModule dir_module libexec/apache22/mod_dir.so

User www
Group www

ServerAdmin webmaster@example.com
ServerName example.com:80
DocumentRoot "/usr/local/www/apache22/data"

<Directory />
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>

<Directory "/usr/local/www/apache22/data">
    Options Indexes FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

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

ErrorLog "/var/log/httpd-error.log"
LogLevel warn

DefaultType text/plain

<IfModule mime_module>
    TypesConfig etc/apache22/mime.types
    AddType application/x-compress .Z
    AddType application/x-gzip .gz .tgz
    AddType application/x-httpd-php .php .aspx
    AddType application/x-httpd-php-source .phps
</IfModule>

Include etc/apache22/extra/httpd-default.conf
Include etc/apache22/Includes/*.conf
```

Listing 19. */usr/local/etc/puppet/manifests/classes/resource_group.pp*

```
class apache22 {
    package { 'www/apache22':
    ensure  => installed,
    provider => ports,
    }

    file { "/usr/local/etc/apache22/httpd.conf":
        owner   => root,
        group   => wheel,
        mode    => 0640,
        source  => "puppet:///files/httpd.conf",
        require => Package['www/apache22'],
    }
    service { 'apache22_service':
        ensure     => running,
        name       => "apache22",
        enable     => true,
        hasrestart => true,
        hasstatus => true,
        subscribe => File['/usr/local/etc/apache22/
                   httpd.conf'],
    }
}
```

Listing 20. */usr/local/etc/puppet/manifests/site.pp*

```
node 'puppet-agent.example.com' {
    include apache22
    include sudoers
    include sshd
```

agent. Then the ssh daemon in *Puppet* agent will be restarted, for the changes to take effect.

In *puppet-master.example.com*, create a customized ssh daemon configuration file: Listing 15.

Update *Puppet* class file so that this resource can be use later: Listing 16.

Similarly, include this *resource* onto a node, so that *Puppet* master can push it onto it: Listing 17.

`kick` the *Puppet* agent in order to refresh its catalog:

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

After the *Puppet* agent is refreshed, the ssh daemon configuration files should be downloaded to the puppet-agent.example.com and ssh daemon restarted.

In the resource type *service* under class *sshd*, the parameter *ensure* will make sure that the service name *sshd* is *running*. The parameter *subscribe* also make sure that whenever the file `/etc/ssh/sshd_config` has changes, it'll restart this service.

For more info on other available parameters, visit *http://docs.puppetlabs.com/references/2.7.6/metaparameter.html*.

As for services supported under FreeBSD, visit *http://docs.puppetlabs.com/references/2.7.6/type.html#service*.

Scenario 3: setup and configure Apache 2.2 web server

In this scenario, the *Puppet* master will instruct the agent to install Apache 2.2 and download the desire customized httpd.conf to the *Puppet* agent. Then, the Apache 2.2 web service will be restart for the new settings to take effect.

In *puppet-master.example.com*, create a customized Apache configuration file (Listing 18). Once again, create a resource in the class file: Listing 19.

Similarly, include this *resource* so that *Puppet* master can push it to the *Puppet* agent: Listing 20.

`kick` the *Puppet* agent:

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

After *Puppet* agent is refreshed, Apache 2.2 should be installing, with the configuration files downloaded into the node *puppet-agent.example.com* and the service *apache22* be restarted. This would probably take awhile as it would need to download the necessary files and start the installation through the FreeBSD port system.

In this scenario, we've cover two new parameter for resource types, namely *provider* and *require*. The parameter *provider* tells *Puppet* to use ports (source) instead of packages (pkg) to install software. As for *require*, this is to tell *Puppet* agent that the software Apache22 is to be installed before applying the file `/usr/local/etc/apache22/httpd.conf` and restart the web service *apache22*.

In the above setup and scenarios, *Puppet* demonstrated that it is capable of deploying configuration files and services. *Puppet* can be used in many more ways in lighten the Ssystem administrator's job when deploying new installation, pushing configuration files and actions to servers, and streamlining policy across servers. After reading this article, I hope that you'll get an idea on how *Puppet* can help in configuration management as well as managing FreeBSD servers.

On the Web

- *http://docs.puppetlabs.com/* – Puppet Labs Documentation,
- *http://docs.puppetlabs.com/learning/* – Learning Puppet.

EDWARD TAN

The author's day-to-day job is administrating a bunch of servers running on FreeBSD. In his free time, he blogs about techie stuff at http://psybermonkey.net, learns about Perl and thinks about how to contribute back to the FreeBSD community.

FreeBSD IPS With Snort Inline

A number of articles have been written covering the basic configuration of Snort in IDS mode on the different BSD operating systems. One feature that is not typically discussed is Snort's ability to integrate with ipfw that allows for inline IPS mode on FreeBSD. This article covers the basic configuration of Snort in IPS mode on a FreeBSD server.

What you will learn…

- How to configure Snort in IPS mode on a FreeBSD server.
- How to setup Snort rules to block malicious traffic.

What you should know…

- Familiarity with compiling code and installing FreeBSD ports.
- Basic knowledge of Network Security tools, specifically IDS/IPS
- Familiarity with ipfw.

Snort inline is normally thought of as a feature reserved for Linux with iptables integration. With the use of divert sockets, processing of network traffic is diverted from `ipfw` to a userland process by listening on a specific diverted port. Snort is configured to listen on this divert port to process the packets based on the configured signatures. Depending on whether a signature match occurs, Snort either allows the traffic, or proceeds to silently drop or close the connection with a TCP reset. The prevention action is configured in each of the Snort signatures.

This setup requires two installs of FreeBSD 8.2 to be configured on a network . VirtualBox is used to setup the IPS as one virtual machine with the second FreeBSD VM networked as shown in Figure 1.

Both of the FreeBSD VM's should be setup with the i386 minimal install including the ports tree (See FREEBSD-INSTALL for installation instructions). The default FreeBSD kernel for 8.2 includes the ability to use divert sockets, but the module needs to be enabled in order to use the functionality. Listing 1 shows the entry in the `/boot/loader.conf` file on the FreeBSD-IPS install required for divert socket functionality.

Listing 2 details the necessary steps for building Snort with some minor changes to the default configuration. The alert_syslog output plugin is used to demonstrate the capabilities of the IPS instead of using barnyard2 which should always be used in production for processing unified2 output generated by Snort.

In order to demonstrate the ability of Snort to block attacks, Apache is installed to provide a network service to attack. Listing 3 details the installation of apache22 with default settings.

One of the most important settings to configure is the divert socket firewall rule which is required to send packets to Snort for processing. Listing 4 shows the firewall rules to use with `ipfw`. The rule file should be saved

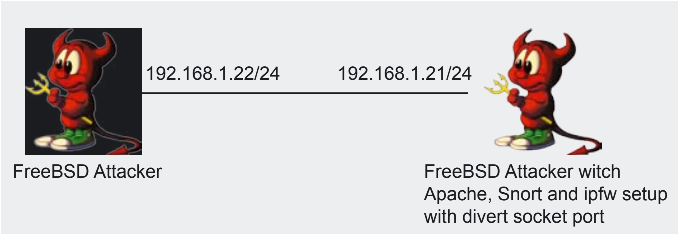

192.168.1.22/24
192.168.1.21/24
FreeBSD Attacker
FreeBSD Attacker witch Apache, Snort and ipfw setup with divert socket port

Figure 1. *Network setup including the FreeBSD Snort setup*

Listing 1. *The /boot/loader.conf setting required for the FreeBSD-IPS to use divert sockets*

```
echo 'ipdivert_load="YES"' >> /boot/loader.conf
```

Listing 2. *Install prerequisite ports and download Snort with DAQ to build with ipfw divert socket support. The following steps use -DBATCH and assume the default settings for each port*

```
cd /usr/ports/ftp/wget
make -DBATCH install clean
rehash
cd /usr/ports/textproc/flex
make -DBATCH install clean
cd /usr/ports/devel/pcre
make -DBATCH install clean
cd /usr/ports/net/libdnet/
make -DBATCH install clean
mkdir /usr/src/snort && cd /usr/src/snort
wget http://www.snort.org/dl/snort-current/daq-
                    0.6.2.tar.gz -O daq-0.6.2.tar.gz
tar xzf daq-0.6.2.tar.gz
cd daq-0.6.2
./configure
make
make install
wget http://www.snort.org/dl/snort-current/snort-
                    2.9.2.tar.gz -O snort-2.9.2.tar.gz
cd snort-2.9.2
./configure --enable-ipv6 --enable-gre --enable-mpls --
                    enable-targetbased --enable-decoder-
                    prepocessor-rules \
--enable-ppm --enable-perfprofiling --enable-zlib --
                    enable-active-response --enable-
                    normalizer --enable-reload \
--enable-react --enable-flexresp3
make
make install
mkdir -p /usr/local/etc/snort
cp /usr/src/snort/snort-2.9.2/etc/*.conf* /usr/local/
                    etc/snort/
cp /usr/src/snort/snort-2.9.2/etc/*.map /usr/local/etc/
                    snort/
mkdir -p /usr/local/lib/snort_dynamicrules/
mkdir -p /usr/local/etc/snort
mkdir -p /usr/local/etc/snort/rules
mkdir -p /usr/local/etc/snort/so_rules
mkdir -p /usr/local/etc/snort/preproc_rules
mkdir -p /var/log/snort
mkdir -p /var/log/barnyard2
touch /usr/local/etc/snort/rules/local.rules
sed -i '' "s/ipvar HOME_NET any/ipvar HOME_NET \
                    [192.168.1.21\/32\]/" /usr/local/
                    etc/snort/snort.conf
sed -i '' 's/ipvar EXTERNAL_NET any/ipvar EXTERNAL_NET
                    \[!\$HOME_NET\]/' /usr/local/etc/
                    snort/snort.conf
sed -i '' 's/var RULE_PATH \.\.\/rules/var RULE_PATH
                    rules/' /usr/local/etc/snort/
                    snort.conf
sed -i '' 's/var SO_RULE_PATH \.\.\/so_rules/var SO_
                    RULE_PATH so_rules/' /usr/local/etc/
                    snort/snort.conf
sed -i '' 's/var PREPROC_RULE_PATH \.\.\/preproc_rules/
                    var PREPROC_RULE_PATH preproc_rules/
                    ' /usr/local/etc/snort/snort.conf
sed -i '' '/^include \$RULE_PATH\/.*.rules$/d' /usr/
                    local/etc/snort/snort.conf
echo "output alert_syslog: LOG_DAEMON LOG_ALERT" >>
                    /usr/local/etc/snort/snort.conf
echo 'include $RULE_PATH/local.rules' >> /usr/local/etc/
                    snort/snort.conf
```

Listing 3. *Install apache22 from the ports tree*

```
cd /usr/ports/www/apache22/
make -DBATCH install clean
```

Listing 4. *Creating the file/etc/rc.firewall-ips (Note: Additional rules are added for ICMP, TCP/80 and UDP/53 to pass the traffic if it is not blocked by Snort)*

```
cat << EOF > /etc/rc.firewall-ips;
ipfw -q flush
ipfw -q add 01100 divert 8000 ip4 from any to any
ipfw -q add 01200 allow tcp from any to me 22 in via em0
ipfw -q add 01300 allow tcp from any to me 80 in via em0
ipfw -q add 01400 allow udp from any to me 53 in via em0
ipfw -q add 01500 allow icmp from any to any
ipfw -q add 01600 allow ip4 from any to any
EOF
```

Listing 5. *Creating the snort rc script and adding startup information to /etc/rc.conf.local for apache22 and Snort*

```
cat << EOF > /usr/local/etc/rc.d/snort;
#!/bin/sh

# \$FreeBSD\$
#
# PROVIDE: snort
# REQUIRE: LOGIN
# KEYWORD: shutdown
#
# Add the following lines to /etc/rc.conf.local or /etc/
                    rc.conf
# to enable this service:
#
# snort_enable (bool):   Set to NO by default.
#               Set it to YES to enable snort.
# snort_config (path):   Set to /usr/local/etc/snort/
                    snort.conf
#               by default.
#

. /etc/rc.subr

name="snort"
rcvar=\${name}_enable

command=/usr/local/bin/\${name}

load_rc_config \$name

: \${snort_enable="NO"}
: \${snort_config="/usr/local/etc/snort/snort.conf"}

command_args="--pid-path /var/run --create-pidfile --
                    daq=ipfw -Q -D -k none -c \$snort_
                    config"

run_rc_command "\$1"

EOF

chmod 555 /usr/local/etc/rc.d/snort;

echo 'accf_http_load="YES"' >> /boot/loader.conf
echo 'apache22_enable="YES"' >> /etc/rc.conf.local
echo 'snort_enable="YES"' >> /etc/rc.conf.local
```

Listing 6. *Adding the /etc/rc.conf settings to start ipfw when booting the system*

```
echo 'firewall_enable="YES"' >> /etc/rc.conf
echo 'firewall_script="/etc/rc.firewall-ips"' >> /etc/rc.conf
```

Listing 7. *Creating a signature to block an attempt to retrieve /etc/passwd*

```
cat << EOF >> /usr/local/etc/snort/rules/local.rules;

# This signature looks for an attempt to retrieve the
# /etc/passwd file with a web request.

block tcp $EXTERNAL_NET any -> $HOME_NET $HTTP_PORTS
                    (msg:"HTTP FreeBSD-ATTACKER /etc/
                    passwd attempt"; flow:established,to_
                    server; content:"/etc/passwd";
                    http_uri; nocase; classtype:
                    attempted-admin; sid:2200001; rev:
                    1;)

EOF
```

Listing 8. *Command line output from the FreeBSD-Attacker system trying to attack the Apache server*

```
FreeBSD-Attacker# printf "GET / HTTP/1.1\r\nHost:
                    192.168.1.21\r\nUser-Agent: Netcat\
                    r\nContent-type: text/html\r\n\r\n"
                    | nc -vvnn 192.168.1.21 80
Connection to 192.168.1.21 80 port [tcp/*] succeeded!
HTTP/1.1 200 OK
Date: Sat, 17 Dec 2011 06:20:26 GMT
Server: Apache/2.2.17 (FreeBSD) mod_ssl/2.2.17 OpenSSL/
                    0.9.8q DAV/2
Last-Modified: Sat, 20 Nov 2004 20:16:24 GMT
ETag: "ba199-2c-3e9564c23b600"
Accept-Ranges: bytes
Content-Length: 44
Content-Type: text/html

<html><body><h1>It works!</h1></body></html>FreeBSD-
                    Attacker#
FreeBSD-Attacker# printf "GET /etc/passwd HTTP/1.1\r\
                    nHost: 192.168.1.21\r\nUser-Agent:
                    Netcat\r\nContent-type: text/html\r\
                    n\r\n" | nc -vvnn 192.168.1.21 80
Connection to 192.168.1.21 80 port [tcp/*] succeeded!
```

Listing 9. *Snort log entry for the blocked attack from /var/log/messages*

```
Dec 17 01:20:49 FreeBSD-IPS snort[838]: [1:2200001:1] HTTP FreeBSD-ATTACKER /etc/passwd attempt [Classification:
                   Attempted Administrator Privilege Gain] [Priority: 1] {TCP} 192.168.1.22:17123 -> 192.168.1.21:
                   80
```

References

- FREEBSD-INSTALL: *http://www.freebsd.org/doc/handbook/install-start.html*
- FreeBSD-snort_inline: *http://freebsd.rogness.net/snort_inline/*
- FreeBSD IPFW: *http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/firewalls-ipfw.html*
- FreeBSD Inline Forum Post: *https://forums.snort.org/forums/support/topics/snort-inline-on-freebsd-ipfw*
- Snort: *http://www.snort.org*
- Emerging Threats Rules: *http://www.emergingthreats.net*

as `/etc/rc.firewall-ips` to differentiate it from the standard `/etc/rc.firewall` rule file distributed with FreeBSD.

Listing 5 creates the necessary Snort rc script to be placed in `/usr/local/etc/rc.d`. With the local settings in `/etc/rc.conf.local`, Snort and apache22 will automatically start at boot.

Listing 6 adds the necessary `/etc/rc.conf` entries to load the custom firewall rules when booting the FreeBSD-IPS VM.

There is an important issue to keep in mind when using divert sockets with `ipfw`. If Snort is not running to process packets, all of the traffic will be dropped as it sent to the divert socket port with nothing to redirect the packet back to the other `ipfw` rules for processing.

Normally, a Snort signature contains the *alert* rule option which will generate an alert but take no action in preventing the packet from reaching the target. In order to block the packet, the *block* or *drop* rule option is required. Listing 7 creates an HTTP signature to block traffic based on the presence of a string in the network traffic.

Rebooting the FreeBSD-IPS VM will start Apache, Snort and will setup the firewall to work with Snort inline. Once the system has started up, the FreeBSD-Attacker system should be able to ping the IPS sensor. In order to demonstrate the IPS, the FreeBSD-Attacker VM will use printf and netcat to make an HTTP request to the Apache server. Listing 8 shows a valid HTTP request, followed by an attempt to retrieve the Apache server.

The first request was passed to the Apache web server and was answered with an HTTP 200 OK response header. The second request included the `/etc/passwd` string in the URL, which Snort blocked and did allow to be processed by the Apache server. Listing 9 shows the log entry from `/var/log/messages` as Snort was setup to use syslog for logging.

This article presents just the basics of setting up an inline solution using Snort and FreeBSD. Complex setups can include setting up a private network sitting behind the FreeBSD-IPS thereby protecting a network segment from attacks. In this example, only a single signature was used to demonstrate the functionality of blocking attacks. There are freely available rulesets such as Emerging Threats along with the registered ruleset available from Sourcefire that can be used to protect a system or network from attacks.

MICHAEL SHIRK

Michael Shirk is a BSD zealot who has worked with OpenBSD and FreeBSD for over 6 years. He works in the security community and supports Open-Source security products that run on BSD operating systems. The author wishes to thank J. J. Cummings for his assistance with testing this article.

malloc(9)

The Kernel's General Purpose Memory Allocator

The release of 4.3BSD in 1988 introduced a new memory allocation mechanism intended to be general enough to effectively meet the needs of diverse kernel subsystems requiring dynamic memory allocation.

What you will learn…

- dynamic kernel memory allocation
- vmstat output for kernel memory usage statistics

What you should know…

- A page of memory – a fixed sized piece of memory (typically 4k in size) used by the virtual memory management system.

Before 4.3BSD, the kernel had several different mechanisms for handling requests for memory, each specialized for the particular type of allocation. Developing new allocation mechanisms to meet the unique demands of new kernel services was producing complexity and reducing efficiency. The new kernel memory allocator, malloc(9), replaced all other memory allocators in 4.3BSD. Its design, based upon known patterns of memory usage, proved to be both time and space efficient. It created a single, easy to use interface, similar to the malloc(3) and free(3) functions found in the familiar C library. The original design by Marshall Kirk McKusick and Michael Karels is described in the paper *Design of a General Purpose Memory Allocator for the 4.3BSD Kernel*, (*Proceedings of the San Francisco USENIX Conference*, June 1988)

FreeBSD, NetBSD and OpenBSD still use the malloc(9) interface for dynamically allocating kernel memory. Its current usage however is limited to requests from subsystems requiring memory of arbitrary sizes. Frequent allocations for fixed size, single purpose pieces of memory are usually handled by dedicated kernel memory pools or slab allocators.

This article gives an overview of malloc(9) and its corresponding function free(9) and explains how this type of dynamically allocated memory is managed within the kernel. Although many of the high level concepts are the same across the BSD distributions, the lower level implementations differ enough that a comprehensive overview of malloc(9) across all BSD derivatives would exceed the scope of this article. It should be assumed therefore that everything is OpenBSD specific unless explicitly stated otherwise.

The collection of kernel memory usage statistics for malloc(9)/free(9) is set by the KMEMSTATS option when building the kernel. OpenBSD turns this option on by default. The statistics themselves are accessible to userland programs via the sysctl(3,8) facility. The vmstat utility queries sysctl to gather relevant information which it then displays to the user. On OpenBSD, the systat(1) utility also displays this information with the added advantage of periodically updating the output similar to the well known top(1) utility. The particular systat(1) views discussed in this article are the *malloc* and *buckets* view.

A Bit About How Kernel Memory is Organized

The total available memory on a given system is divided into two parts: memory which is designated for userland programs and memory which is dedicated for exclusive use by the kernel. Kernel memory is inaccessible to user lever programs.

Within the range of kernel memory are submaps – segregated, contiguous regions set aside for specific purposes. Submaps isolate a region from the rest of kernel memory. They have their own management mechanism that controls access to that range of memory, thereby allowing related allocations to be kept together and not be intermixed with other, unrelated allocations. A submap's size, measured in pages, is static and does not change over time; it can therefore be used to impose a constraint on how much memory a kernel subsystem may consume.

`kmem_map` is a submap used exclusively for dynamic memory allocations. malloc(9) and free(9) are the interface into this region of memory and, along with lower level routines, provide the mechanisms for its management.

The more common name for `kmem_map` is the *kernel malloc arena*. It is created as part of the boot process when the kernel initializes structures needed to manage its memory. The size of the malloc arena is machine dependent and takes into account the system's page size and available memory. In OpenBSD and NetBSD the default upper and lower bounds are set in `/usr/include/machine/param.h` (`NKMEMPAGES_MAX`, `NKMEMPAGE_MIN`). On a running system, the size can be determined by multiplying the system's page size by the number of pages in the malloc arena, e.g.,

```
prompt#> pagesize
4096
prompt#> sysctl vm.kmempages
vm.nkmempages=32768
```

The command pagesize(1) returns the hardware pagesize in bytes. The userland sysctl(1) program queries the system for the number of pages in the malloc arena (*vm.nkmempages*). On this particular system, the size of the malloc arena is 4096 * 32768 (128 megabytes), which incidentally is the default upper bound. If the default size is unacceptable, then it can be set statically by compiling a custom kernel. See the options(4) manpage, specifically the variable NKMEMPAGES.

The subsystems which use memory from the malloc arena are spread throughout all parts of the kernel – networking, file systems, devices, cryptography, and kernel management operations, etc. For each subsystem that calls malloc(9) there is a kmemstats structure which is used to maintain statistics and set limits on that particular subsystem's memory usage. There is a static upper bound on how much memory a single subsystem may use at any given time, and a counter which, under certain conditions, tracks how often this upper bound has been reached. Other counters track the number of times a subsystem calls malloc(9) and the current number items in use.

The Implementation

On OpenBSD, malloc(9) and free(9) are the only two functions which dynamically allocate and free general purpose memory.

malloc(9) takes three arguments: the size of memory requested, the subsystem which is requesting the memory and a set of four possible flags, three of which indicate what should occur if malloc(9) is unable to immediately satisfy the request. The usual case is when the subsystem making the request is already using the maximum memory it is allowed have. If the `M_WAIT` flag is set, malloc(9) will sleep until the subsystem returns enough memory to bring its total memory usage below the maximum threshold, whereupon it will then wake up and return the requested memory. Alternatively, the `M_NOWAIT` flag tells malloc(9) not to sleep but rather to immediately indicate failure by returning `NULL`.

A third flag, `M_CANFAIL` indicates how malloc should fail if the `M_WAIT` flag is set, but certain rare conditions prevent it from be able to return the requested memory. This can occur if there is no more free memory available in the malloc arena. In such a case, if `M_CANFAIL` is set, malloc(9) will return `NULL` to indicate failure; otherwise the system will panic with the error message *malloc: out of space in kmem_map*.

The free(9) function is used by the subsystem to return memory it no longer needs. free(9) takes only two arguments: the address of the memory to be returned and the subsystem returning the memory.

Memory allocated from the malloc arena is organized as a set of lists, commonly referred to as *buckets*. Each bucket is a single list (called the *free-list*) of equally sized pieces of memory, and there is one bucket for each power of two, ranging from 16 bytes (the minimum size) to 524288 bytes (the maximum size). The malloc(9) function takes the size argument, rounds it up to the next power of two and allocates a piece of memory from the appropriate bucket (i.e., free list). For example, a request for 23 bytes of memory will take memory from the bucket containing 32 byte pieces. When a subsystem calls free(9), the memory is put back onto the free list for the appropriate bucket where it can be re-used again for later requests.

When the system initializes the malloc arena, the free lists in all the buckets are empty. The first call to malloc(9) will cause a low level memory allocation. For buckets that hold items less than the size of a page, this lower level allocation will return a single page which will be carved up into appropriate sized pieces and put in the bucket. This kind of allocation is efficient because a single request creates several pieces of memory which become immediately available. Buckets with items equal to or larger than a page cannot benefit from this optimization. For these requests, the memory is given directly to the subsystem which called malloc(9). However, when this memory is freed, it will be put on the free list in the appropriate bucket where it will remain until there is another request. The special case is for buckets with items larger than twice the page size. Because these items are so large, they are never put onto the free list after a call to free(9) but are instead returned to the malloc arena. Therefore the free lists in these buckets are always empty.

With the exception of allocations twice the page size, a page of memory allocated to a bucket remains associated with that bucket permanently. Counters exist which track the amount of free memory in a bucket, but no action is taken to remove unused pages and return them to the malloc arena.

Each bucket has its own kmembucket structure for maintaining usage statistics. Bucket statistics displayed by vmstat are taken exclusively from this structure.

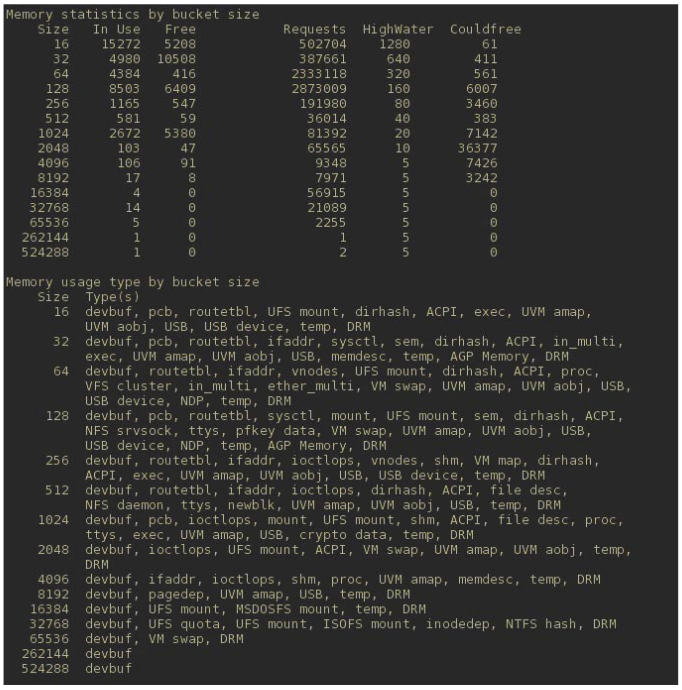
Figure 1 is the partial output of *vmsstat -m* (on OpenBSD, these statistics are also available with *systat buckets*). The upper table shows memory statistics by bucket size:

- Size – the size of the bucket items
- In Use – the number of items from this bucket which are actively in use by a kernel subsystem
- Free – the number of items on the free list
- Requests – total number of requests (since boot) for an item in this bucket
- HighWater – a high watermark for the number of items in the bucket
- Couldfree – over high watermark and could free

The *Size* column shows the ascending power of two bucket sizes. The total amount of memory allocated to a given bucket equals *Size* * (*Free* + *In Use*). *HighWater* is used for calculating unused memory in a bucket. For buckets with items less than a page size the value is the number of items which fit on a single page of memory multiplied by 5 (e.g., on a system with 4k pages, and a bucket holding 64 byte pieces, the high watermark is 4096 / 64 * 5 = 320). Buckets with larger items have a high watermark of 5. Note also that the free lists in buckets for sizes larger than twice the page size (16k) are (and always will be) zero.

When malloc(9) was first implemented in 4.3BSD, consideration was given to how unused memory on a bucket's free list could be reclaimed if there were a memory shortage in the malloc arena. The values under *HighWater* and *Couldfree* were created with this purpose in mind. *Couldfree* is incremented after a call to free(9) which results in 1) there being an unused page in a bucket, i.e., a page from which no pieces are in use by a subsystem, and 2) the number of items on the bucket's free list is above the high watermark.

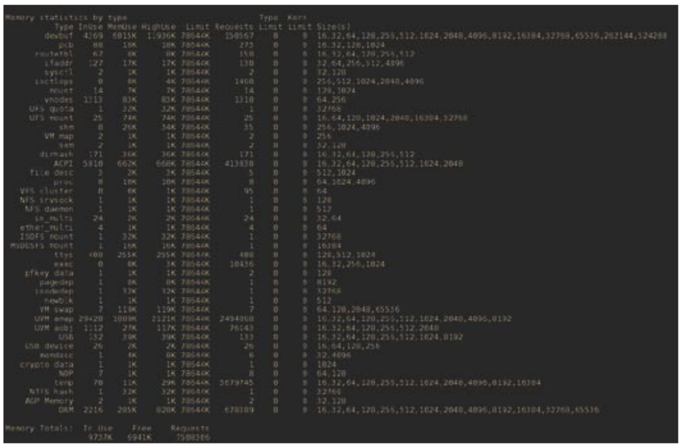
The original paper by McKusick/Karels addressed the problem that free lists could get very large and thereby reduce available memory for other requests. In practice however, this didn't occur, and it was therefore not considered an immediate problem. (While it has not been possible to do an exhaustive search of all releases based on 4.3BSD, the author, having searched through many



```
Memory statistics by bucket size
    Size   In Use   Free           Requests  HighWater  Couldfree
      16    15272   5208             502704     1280         61
      32     4980  10508             387661      640        411
      64     4384    416            2333118      320        561
     128     8503   6409            2873009      160       6007
     256     1165    547             191980       80       3460
     512      581     59              36014       40        383
    1024     2672   5380              81392       20       7142
    2048      103     47              65565       10      36377
    4096      106     91               9348        5       7426
    8192       17      8               7971        5       3242
   16384        4      0              56915        5          0
   32768       14      0              21089        5          0
   65536        5      0               2255        5          0
  262144        1      0                  1        5          0
  524288        1      0                  2        5          0

Memory usage type by bucket size
    Size  Type(s)
      16  devbuf, pcb, routetbl, UFS mount, dirhash, ACPI, exec, UVM amap,
          UVM aobj, USB, USB device, temp, DRM
      32  devbuf, pcb, routetbl, ifaddr, sysctl, sem, dirhash, ACPI, in_multi,
          exec, UVM amap, UVM aobj, USB, memdesc, temp, AGP Memory, DRM
      64  devbuf, routetbl, ifaddr, vnodes, UFS mount, dirhash, ACPI, proc,
          VFS cluster, in_multi, ether_multi, VM swap, UVM amap, UVM aobj, USB,
          USB device, NDP, temp, DRM
     128  devbuf, pcb, routetbl, sysctl, mount, UFS mount, sem, dirhash, ACPI,
          NFS srvsock, ttys, pfkey data, VM swap, UVM amap, UVM aobj, USB,
          USB device, NDP, temp, AGP Memory, DRM
     256  devbuf, routetbl, ifaddr, ioctlops, vnodes, shm, VM map, dirhash,
          ACPI, exec, UVM amap, UVM aobj, USB, USB device, temp, DRM
     512  devbuf, routetbl, ifaddr, ioctlops, dirhash, ACPI, file desc,
          NFS daemon, ttys, newblk, UVM amap, UVM aobj, USB, temp, DRM
    1024  devbuf, pcb, ioctlops, mount, UFS mount, shm, ACPI, file desc, proc,
          ttys, exec, UVM amap, USB, crypto data, temp, DRM
    2048  devbuf, ioctlops, UFS mount, ACPI, VM swap, UVM amap, UVM aobj, temp,
          DRM
    4096  devbuf, ifaddr, ioctlops, shm, proc, UVM amap, memdesc, temp, DRM
    8192  devbuf, pagedep, UVM amap, USB, temp, DRM
   16384  devbuf, UFS mount, MSDOSFS mount, temp, DRM
   32768  devbuf, UFS quota, UFS mount, ISOFS mount, inodedep, NTFS hash, DRM
   65536  devbuf, VM swap, DRM
  262144  devbuf
  524288  devbuf
```

Figure 1. *Statistics by buckets size (above); type statistics by bucket size*



```
Memory statistics by type                           Type  Kern
          Type InUse MemUse HighUse  Limit Requests Limit Limit Size(s)
        devbuf  4169  6815K  11936K 78544K   150567     0     0 16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536,262144,524288
           pcb    88    18K     18K 78544K      275     0     0 16,32,128,1024
      routetbl    67     8K      8K 78544K      158     0     0 16,32,64,128,256,512
        ifaddr   127    17K     17K 78544K      130     0     0 32,64,256,512,4096
        sysctl     2     1K      1K 78544K        2     0     0 32,128
      ioctlops     0     0K      4K 78544K     1460     0     0 256,512,1024,2048,4096
         mount    14     7K      7K 78544K       14     0     0 128,1024
        vnodes  1113    83K     83K 78544K     1118     0     0 64,256
     UFS quota     1    32K     32K 78544K        1     0     0 32768
     UFS mount    25    74K     74K 78544K       25     0     0 16,64,128,1024,2048,16384,32768
           shm     8    26K     34K 78544K       35     0     0 256,1024,4096
        VM map     2     1K      1K 78544K        2     0     0 256
           sem     2     1K      1K 78544K        2     0     0 32,128
       dirhash   171    36K     36K 78544K      171     0     0 16,32,64,128,256,512
          ACPI  5818   662K    668K 78544K   413838     0     0 16,32,64,128,256,512,1024,2048
     file desc     3     2K      3K 78544K        5     0     0 512,1024
          proc     8    10K     10K 78544K        8     0     0 64,1024,4096
   VFS cluster     0     0K      1K 78544K       95     0     0 64
   NFS srvsock     1     1K      1K 78544K        1     0     0 128
    NFS daemon     1     1K      1K 78544K        1     0     0 512
      in_multi    24     2K      2K 78544K       24     0     0 32,64
   ether_multi     4     1K      1K 78544K        4     0     0 64
   ISOFS mount     1    32K     32K 78544K        1     0     0 32768
 MSDOSFS mount     1    16K     16K 78544K        1     0     0 16384
          ttys   408   255K    255K 78544K      408     0     0 128,512,1024
          exec     0     0K      3K 78544K    10436     0     0 16,32,256,1024
    pfkey data     1     1K      1K 78544K        2     0     0 128
       pagedep     1     8K      8K 78544K        1     0     0 8192
      inodedep     1    32K     32K 78544K        1     0     0 32768
        newblk     1     1K      1K 78544K        1     0     0 512
       VM swap     7   119K    119K 78544K        7     0     0 64,128,2048,65536
      UVM amap 29420  1009K   2121K 78544K  2494068     0     0 16,32,64,128,256,512,1024,2048,4096,8192
      UVM aobj  1112    27K    117K 78544K    76143     0     0 16,32,64,128,256,512,2048
           USB   132    39K     39K 78544K      133     0     0 16,32,64,128,256,512,1024,8192
    USB device    26     2K      2K 78544K       26     0     0 16,64,128,256
       memdesc     1     4K      8K 78544K        6     0     0 32,4096
   crypto data     1     1K      1K 78544K        1     0     0 1024
           NDP     7     1K      1K 78544K        8     0     0 64,128
          temp    70    11K     29K 78544K  3679745     0     0 16,32,64,128,256,512,1024,2048,4096,8192,16384
     NTFS hash     1    32K     32K 78544K        1     0     0 32768
    AGP Memory     2     1K      1K 78544K        2     0     0 32,128
           DRM  2216   285K    820K 78544K   678309     0     0 16,32,64,128,256,512,1024,2048,4096,8192,16384,32768,65536

Memory Totals:  In Use    Free    Requests
                 9737K   6941K     7588386
```

Figure 2. *Memory statistics by type*

source files, hasn't found an implementation that tried to recover free, unused pages from a bucket.) The value of *Couldfree* is never decremented, and no action is taken based upon its value.

The second table of output shows memory usage type by bucket size. *Size* is the list of buckets, and *Types(s)* are the various kernel subsystems which have used memory from the given bucket. More descriptive definitions for the *Type(s)* can be found at the top of `sys/malloc.h`.

Figure 2 is the partial output of *vmstat -m* which shows memory statistics by type, i.e., kernel subsystem. (On OpenBSD, this is also displayed with *systat malloc*; values for *MemUse*, *HighUse* and *Limit* are in bytes)

- Type – kernel subsystem
- InUse – the number of bucket items currently in use by this particular subsystem
- MemUse – the total memory currently used by the subsystem
- HighUse – the largest amount of memory used by the subsystem at one time since boot
- Limit – the maximum amount a memory this *Type* is allowed to use
- Requests – the total number of successful calls to malloc(9)
- Type Limit – number of times malloc(9) slept because *MemUse* was greater than *Limit*
- Kern Limit – number of times blocked because there was no more memory in the malloc arena
- Size(s) – a list of buckets from which the *Type* requested memory at least once

Note: Sizes displayed in *K* are byte values rounded up to the next kilobyte.

If a particular subsystem is never used, or never calls malloc(9), then it won't appear in the output. *Limit* is set when initializing the malloc arena. It is the same for all types and the value is dependent on the size of the malloc arena. The formula is nkmempages * *page size* * 6 / 10.

If *MemUse* is already equal to (or above) *Limit* when malloc(9) is called, then the call will either fail and return `NULL`, or, if the caller is willing to wait, malloc(9) will increment *Type Limit* and go to sleep until the particular subsystem returns enough memory to bring *MemUse* below *Limit*.

Sizes lists buckets from which a particular subsystem requested memory. This reflects all allocations since the subsystem made its first request (the vmstat output differs from systat's – systat shows a series of vertical bars (`|`) and/or dots (`.`) representing buckets in ascending order by size. The presence of the vertical bar indicates bucket usage, whereas a dot represents a bucket from which nothing has been allocated; e.g., |.||..||..||.... means memory was allocated from the first, third and fourth buckets (16k, 64k, 128k) etc; none was allocated from second, fifth or sixth buckets, etc.

The 'Kern Limit' value originally appeared in 4.3BSD and was incremented whenever malloc(9) couldn't add more memory to a bucket because there was none available in the kernel malloc arena. Subsequent releases of BSD from CSRG stopped incrementing this variable. Although it is still in the kmemstat structure, it isn't used anywhere in the NetBSD/OpenBSD code except by programs that extract variables from kmemstat (e.g., sysctl, vmstat, systat). It will therefore always be zero.

Conclusion: malloc(9) Then and Now

When malloc(9) was first implemented in 4.3BSD, it was intended to replace all other existing mechanisms for dynamically allocating kernel memory. Original design decisions were based upon known patterns of usage. In the two decades since the release of 4.3BSD other methods have been shown to be more effective for certain types of memory allocation; these include kernel memory pools dedicated to specific structures and slab allocators.

NetBSD and FreeBSD have extended the functionality of the malloc(9) interface, adding features found in the userland version of malloc(3). Also, in FreeBSD, the back end is implemented using slab allocation; useful statistics are visible with the `-z` option to vmstat. Additionaly, the power of two strategy is used only for allocations less than or equal to the page size; larger allocations are rounded up to the next page. The default NetBSD kernel disables the KMEMSTATS option, meaning the kernel doesn't collect statistics on malloc(9)'s usage and therefore doesn't enforce the memory usage limit for subsystems. Although malloc(9) is still used throughout the NetBSD kernel, it is considered deprecated and newer kernel services should use newer, alternate methods for obtaining memory.

PAUL MCMATH

Paul McMath has worked as a Unix admin for 10+ years in Europe and the United States. He has been using one BSD variant or another as his OS of choice since 2002.

Keeping Base System

& Packages Up-To-Date

Today, I would like to 'touch' an ungrateful topic of keeping both FreeBSD's base system and installed packages up-to-date.

What you will learn…

- Knowledge about upgrading/updating FreeBSD's base system (check http://freebsd.org/handbook/updating-upgrading.html chapter from FreeBSD Handbook)
- Knowledge about adding/removing packages (check man pkg_info/man pkg_add/man pkg_delete).
- Knowledge about Ports concept and its general usage (check http://freebsd.org/handbook/ports.html chapter from FreeBSD Handbook)

What you should know…

- Knowledge about keeping both FreeBSD's base system and package up-to-date.

After I started using FreeBSD at 5.4 times (2005) I have tried various methods of keeping my FreeBSD installations up-to-date, many of them terribly failed, but some recent ones seem to do the job as advertised. Even not so recently ago I thought, lets stick to RELEASE and do not compile newer versions of packages as there are available packages at FTP ... but there is a big problem with such attitude. First, once the RELEASE is completed, there are only security fixes for the base system, but there are no bug fixes for the RELEASE. Its even worse with packages for RELEASE since once they are built they are never later updated, even if they have security issues, not even mentioning bugs. So that is definitely not the right way.

The sollution seems to be tracking STABLE tree for the base system along with packages that are built every 2 weeks for the STABLE tree and compiling only when there are security issues in some of the installed packages, but there are for example 10 more days before their rebuilt versions would show up on the STABLE tree FTP. Below I would try to describe all that process of keeping FreeBSD up-to-date as simple as possible. In the first part I would focus on the base system and the second one will cover keeping packages up-to-date.

Some important information about keeping Your system this way. You would not rebuild the base system every day, not even every week, just when needed. Now what does it mean 'when needed' ... For example when there is a security issue, You would just follow the instructions in the SA (security advisory) to fix that issue, there is no need to rebuild whole world. The only reasons to rebuild the base system are that there has been found and fixed a bug in STABLE that affects You or that You need new features that has been merged into the STABLE branch (from CURRENT for example) like newer ZFS version or whatever.

As for the installation, You can install the RELEASE version and update to STABLE or install the daily STABLE snapshot so You would not have to build entire base system from source, the daily ISO images are available at *http://pub.allbsd.org/FreeBSD-snapshots/* server.

PART I
Keeping the FreeBSD Base System Up-To-Date
Some Facts About FreeBSD's Base System

- once RELEASE is completed, there are only security fixes, there are no bug fixes
- bugs in STABLE tree are fixed
- security issues are also fixed in STABLE
- the RELEASE branch allows to use binary updates via freebsd-update tool for security fixes

- the STABLE branch requires compiling of the FreeBSD base system

We need to clone the current cource tree if we want to build up to date STABLE branch FreeBSD's base system, we will also need to update our sources to the current state so its quite handy to find fastest server for Your location, it can be easily done by using `sysutils/fastest_cvsup` package as showed in Listing 1.

For my location it is *cvsup.pl.freebsd.org* which in most cases will be different ther for Your location, so remember to put Your's fastest in the next steps.

Create simple `supfile` that will be used by `csup(1)` to keep FreeBSD's base system sources up-to-date (Listing 2). There are useful examples under `/usr/share/examples/cvsup/` if you want to *dig more*.

Now lets get/update our sources to the current state (Listing 3), the list of edited/checked files will be quite different on Your box since I already have quite up-to-date sources, this will take more time if You do not have the sources on the disk.

Alternatively, You can grab the sources by SVN protocol, but You will need `devel/subversion16` package for that purpose. Its generally a lot faster/easier to *setup* then csup but the 'csup way' has one important advantage, its in the FreeBSD's base system, so its always available, anywhere. With SVN, You will have to add a package first which sometimes may be cumbersome. But as the FreeBSD source tree is kept under SVN it is possible that SVN will be part of the FreeBSD's base system one day.

It's also important to mention, that sources downloaded by SVN are not compatible with the sources grabbed by `csup`, so once You will decide which method to use, stick with it, unless You want to download the whole FreeBSD's source tree again. Below (Listing 4) we have SVN update to 9-STABLE latest state.

It's the same no matter if You download the whole tree or just doing an update from yesterday. If `svn` will complain about anything, just delete the `/usr/src` and type the command again.

Listing 1. *Searching for the fastest csup(1) server with fastest_cvsup(7) package*

```
# pkg_add -r fastest_cvsup
# rehash
# fastest_cvsup -c all
(...)

>>  Speed Daemons:
    - 1st: cvsup.pl.freebsd.org
    - 2nd: cvsup11.ua.freebsd.org
    - 3rd: cvsup5.de.freebsd.org
```

Listing 2 *Creating csup(1) config file that will update sources to 9-STABLE*

```
# cat > /root/stable-supfile << EOF
*default host=cvsup.pl.freebsd.org
*default base=/var/db
*default prefix=/usr
*default release=cvs tag=RELENG_9
*default delete use-rel-suffix
*default compress
src-all
EOF
```

Listing 3. *Updating FreeBSD source tree using csup(1) from the base system*

```
# csup /root/stable-supfile
Connected to 188.125.237.138
Updating collection src-all/cvs
 Edit src/bin/ed/buf.c
 Edit src/sbin/fsck_ffs/main.c
 Edit src/sbin/mdconfig/mdconfig.8
 Edit src/sbin/mdconfig/mdconfig.c
 Edit src/share/man/man4/ath.4
 Edit src/share/man/man4/ath_hal.4
 Edit src/sys/cddl/contrib/opensolaris/uts/common/fs/
                  zfs/zfs_vnops.c
 Edit src/sys/cddl/contrib/opensolaris/uts/common/fs/
                  zfs/zfs_znode.c
 Edit src/sys/dev/ahci/ahci.c
 Edit src/sys/fs/msdosfs/msdosfs_vnops.c
 Edit src/sys/fs/nfsclient/nfs_clbio.c
 Edit src/sys/fs/nfsserver/nfs_nfsdserv.c
 Edit src/sys/fs/nwfs/nwfs_io.c
 Edit src/sys/fs/smbfs/smbfs_io.c
 Edit src/sys/fs/tmpfs/tmpfs_vnops.c
 Edit src/sys/gnu/fs/xfs/FreeBSD/xfs_vnops.c
 Edit src/sys/kern/uipc_usrreq.c
 Edit src/sys/kern/vfs_vnops.c
 Edit src/sys/nfsclient/nfs_bio.c
 Edit src/sys/sparc64/sbus/sbus.c
 Edit src/sys/sys/vnode.h
 Edit src/sys/ufs/ffs/ffs_inode.c
 Edit src/sys/ufs/ffs/ffs_vnops.c
```

Now as we have the sources we can continue to building the FreeBSD's base system from source (Listing 5). As for editing the kernel config, You do not even have to bother about it, just use `GENERIC`, this guide is not about stripping the base system and kernel components, its about keeping everything up-to-date. Of course if You want to, then use Your tweaked kernel config, it will not interfere with the rest of this guide. You may want to put `nice -n 20` in front of `make buildworld` ... line to make that build process less *amusing* for your system. As instructions are completed, Your system will reboot.

Listing 4. *Updating FreeBSD source tree using svn(1) from the devel/subversion16 package*

```
# svn checkout svn://svn.freebsd.org/base/stable/9
                    /usr/src
U    /usr/src/usr.bin/grep/util.c
 U   /usr/src/usr.bin/grep
U    /usr/src/share/man/man5/rc.conf.5
 U   /usr/src/share/man/man5
U    /usr/src/share/man/man9/driver.9
 U   /usr/src/share/man/man9
U    /usr/src/usr.sbin/mergemaster/mergemaster.sh
 U   /usr/src/usr.sbin/mergemaster
 U   /usr/src/lib/libc/stdtime
 U   /usr/src/lib/libc
U    /usr/src/lib/librt/timer.c
 U   /usr/src/lib/librt
U    /usr/src/lib/libpam/modules/pam_ssh/pam_ssh.c
 U   /usr/src/lib/libpam
U    /usr/src/etc/network.subr
U    /usr/src/etc/devd.conf
 U   /usr/src/etc
U    /usr/src/sys/sparc64/pci/schizo.c
U    /usr/src/sys/kern/kern_ctf.c
U    /usr/src/sys/kern/vfs_syscalls.c
U    /usr/src/sys/kern/sys_generic.c
U    /usr/src/sys/netinet/tcp_reass.c
U    /usr/src/sys/netinet6/nd6.c
U    /usr/src/sys/contrib/pf/net/pf.c
 U   /usr/src/sys/contrib/pf
U    /usr/src/sys/amd64/include/segments.h
 U   /usr/src/sys/amd64/include/xen
U    /usr/src/sys/amd64/include/trap.h
U    /usr/src/sys/amd64/amd64/trap.c
U    /usr/src/sys/sys/bus.h
 U   /usr/src/sys
Checked out revision 228452.
```

We are now proceeding to the second phase of the upgrade process (Listing 6). After normal boot (*single-user mode* not required and definitely prohibited while doing upgrade over the network) stop all unneeded services (remember to keep sshd daemon alive if you are doing upgrade via network). If your system booted up properly, then You can make the new testing kernel the default one, at least there should not be any problems with the `GENERIC` kernel config.

The list of started processes (Listing 7) will look something like that including (or not) `sshd(8)` for keeping up the network connection.

Now we can continue to type rest of needed instructions (Listing 8) to finish the update, the `mergemaster(8)` will ask You for the differences in startup scripts that You have modified and configuration files, type `I` to install the new/default config and/or script and select `D` to leave the version that you have in the system, remember that You can also add these changes later, it may be not appreciate to install default firewall config or customized OpenSSH config while doing the network upgrade.

After that second reboot You should have updates to STABLE branch FreeBSD's base system, I wrote 'should' because sometimes things do not go the way we want them to go, especially if you are doing it the first time as once *Aerosmith* sing *I know it's everybody's sin, You got to lose to know how to win*. It would be best to do these instructions

Listing 5. *Building FreeBSD's base system*

```
# cd /usr/src
# rm -r -f /usr/obj
# make buildworld kernel KODIR=/boot/testing
# nextboot -k testing
# shutdown -r now
```

Listing 6. *Making new kernel permanent and stopping unneded services*

```
# cd /boot
# rm -r -f OLD
# mv kernel OLD
# mv testing kernel

# /etc/rc.d/cron stop
# /etc/rc.d/devd stop
# /etc/rc.d/sshd stop
# /etc/rc.d/powerd stop
# /etc/rc.d/syslogd stop
```

as exercise under virtual machine like VirtualBox or QEMU. Also, if you do not feel that STABLE is *production enought*, then You may want to use STABLE packages along with RELEASE base system, You will need to define environment variable `PACKAGESITE` that will point to *http://ftp.freebsd.org/pub/FreeBSD/ports/amd64/packages-9-stable/Latest/* at least for FreeBSD 9.x system.

PART II
Keeping the FreeBSD Packages Up-To-Date

Keeping packages up to date is little more tricky, we will also need the STABLE branch for them as these in RELEASE are not updated. Lets assume that You installed the FreeBSD STABLE snapshot a month ago, along with packages that were built by then, now there

Listing 7. *Cut to minimum list of running processes during the upgrade process*

```
# top -b
last pid: 64835;  load averages:  0.00,  0.00,  0.00  up 0+03:11:51    10:24:37
119 processes: 2 running, 117 sleeping

Mem: 960M Active, 355M Inact, 4014M Wired, 6096K Cache, 8368K Buf, 2527M Free
Swap:

  PID USERNAME      THR PRI NICE   SIZE    RES STATE   C   TIME   WCPU COMMAND
 2178 root            1  54    0 10304K  2748K ppwait  0   0:00  0.00% csh
 2174 root            1  45    0 21696K  1992K wait    0   0:00  0.00% login
 2177 root            1  76    0  6912K  1284K ttyin   0   0:00  0.00% getty
 2176 root            1  76    0  6912K  1284K ttyin   1   0:00  0.00% getty
 2175 root            1  76    0  6912K  1284K ttyin   0   0:00  0.00% getty
  114 root            1  76    0  2764K  1056K pause   0   0:00  0.00% adjkerntz
```

Listing 8. *Installing the newly built base system*

```
# cd /usr/src
# mergemaster -p
# make installworld
# mergemaster -iU
# make delete-old
# shutdown -r now
```

Listing 9. *The ports-check function*

```
function ports-check {
  # FETCH LATEST PORTS TREE
  sudo portsnap fetch update

  # CHECK WHAT NEW VERSIONS EXIST
  sudo portmaster -L --index-only | awk '/ [Nn]ew / {print substr($0,9,9999)}'

  # CHECK SECURITY ISSUES
  sudo portaudit -Fda

  # CHECK /usr/ports/UPDATING MESSAGES
  pkg_updating -d $( ls -ltr -D '%Y%m%d' /var/db/pkg | awk 'END{print $6}' )
}
```

Listing 10. *The ports-update function*

```
function ports-update {

  case ${#} in
    (0) sudo pkg_upgrade -C -a 2>&1 | grep --color=none --line-buffered -E "^(=+>|/usr/ports|/var/db)" ;;
    (*) sudo pkg_upgrade -C $@ 2>&1 | grep --color=none --line-buffered -E "^(=+>|/usr/ports|/var/db)" ;;
  esac

  # FIX DEPENDENCIES AS NEEDED
  sudo portmaster --check-depends
}
```

Listing 11. *The ports-build function*

```
function ports-build {
  # REBUILD SINGLE, SEVERAL OR ALL PORTS
  case ${#} in
    (0) sudo portmaster -y --no-confirm -m 'BATCH=yes'
                   -d -a ;;
    (*) sudo portmaster -y --no-confirm -m 'BATCH=yes'
                   -d $@ ;;
  esac

  # FIX DEPENDENCIES AS NEEDED
  sudo portmaster --check-depends
}
```

Example 1. *Typical output about new/updated ports and new versions available*

```
[code]% ports-check
Looking up portsnap.FreeBSD.org mirrors... 5 mirrors found.
Fetching snapshot tag from portsnap5.freebsd.org... done.
Fetching snapshot metadata... done.
Updating from Mon Sep  5 07:11:28 CEST 2011 to Mon Sep
                    5 08:51:01 CEST 2011.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 0 metadata files... done.
Fetching 10 patches.....10 done.
Applying patches... done.
Fetching 1 new ports or files... done.
Removing old files and directories... done.
Extracting new files:
/usr/ports/chinese/c2t/
/usr/ports/chinese/hc/
/usr/ports/devel/Makefile
/usr/ports/devel/p5-System-Command/
/usr/ports/german/mythes/
/usr/ports/math/p5-Statistics-R/
/usr/ports/polish/hunspell/
/usr/ports/textproc/es-mythes/
/usr/ports/textproc/nl-mythes/
/usr/ports/textproc/sk-mythes/
/usr/ports/textproc/sl-mythes/
Building new INDEX files... done.

New version available: ca_root_nss-3.12.11_1
New version available: expat-2.0.1_2
New version available: tinyxml-2.6.2
New version available: bash-4.1.11
New version available: gstreamer-plugins-0.10.35_1,3
New version available: gtk-2.24.6
New version available: gtk-update-icon-cache-2.24.6
New version available: libsamplerate-0.1.8_1
New version available: nas-1.9.3
New version available: nettle-2.4
New version available: p5-Date-Manip-6.25
New version available: p5-Mail-IMAPClient-3.29
New version available: p5-XML-Parser-2.41
New version available: xterm-273
New version available: filezilla-3.5.1
New version available: firefox-6.0.1,1
New version available: gtk-oxygen-engine-1.1.2
New version available: nginx-1.0.6,1
New version available: qemu-0.11.1_10
20 have new versions available
New database installed.
Database created: Thu Sep  1 21:20:00 CEST 2011
0 problem(s) in your installed packages found.
```

Example 2. *A report that also shows some security issues*

```
% ports-check
Looking up portsnap.FreeBSD.org mirrors... 5 mirrors
                    found.
Fetching snapshot tag from portsnap1.freebsd.org... done.
Fetching snapshot metadata... done.
Updating from Mon Sep  5 10:28:51 CEST 2011 to Mon Sep  5
                    12:07:23 CEST 2011.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 0 metadata files... done.
Fetching 20 patches.....10....20 done.
Applying patches... done.
Fetching 1 new ports or files... done.
Removing old files and directories... done.
Extracting new files:
/usr/ports/MOVED
/usr/ports/Mk/bsd.sites.mk
/usr/ports/cad/Makefile
/usr/ports/devel/Makefile
/usr/ports/devel/p5-Bread-Board-Declare/
/usr/ports/devel/p5-Curses-UI/
/usr/ports/devel/p5-Data-Peek/
/usr/ports/devel/p5-Scope-Upper/
/usr/ports/dns/pear-Net_DNS2/
/usr/ports/lang/p5-Try-Tiny/
/usr/ports/mail/p5-Email-Valid/
/usr/ports/math/p5-Math-BigInt/

/usr/ports/net/pear-Net_SMTP/
/usr/ports/ports-mgmt/portaudit-db/
/usr/ports/sysutils/py-supervisor/
/usr/ports/sysutils/zfsnap/
/usr/ports/www/nginx-devel/
/usr/ports/www/nginx/
/usr/ports/www/rubygem-passenger/
/usr/ports/x11/Makefile
Building new INDEX files... done.
New version available: arc-5.21p
New version available: ca_root_nss-3.12.11_1
New version available: expat-2.0.1_2
New version available: tinyxml-2.6.2
New version available: bash-4.1.11
New version available: gstreamer-plugins-0.10.35_1,3
New version available: gtk-2.24.6
New version available: gtk-update-icon-cache-2.24.6
New version available: libsamplerate-0.1.8_1
New version available: nas-1.9.3
New version available: nettle-2.4
New version available: p5-Date-Manip-6.25
New version available: p5-Mail-IMAPClient-3.29
New version available: p5-XML-Parser-2.41
New version available: xterm-273
New version available: filezilla-3.5.1
New version available: firefox-6.0.1,1
New version available: gtk-oxygen-engine-1.1.2
New version available: nginx-1.0.6,1
New version available: qemu-0.11.1_10
20 have new versions available
auditfile.tbz                                     100% of
                    69 kB   54 kBps
New database installed.
Database created: Mon Sep  5 12:35:01 CEST 2011
Affected package: ca_root_nss-3.12.9
Type of problem: ca_root_nss -- Extraction of unsafe
                    certificates into trust bundle..
Reference: http://portaudit.FreeBSD.org/1b27af46-d6f6-
                    11e0-89a6-080027ef73ec.html

Affected package: ca_root_nss-3.12.9
Type of problem: nss/ca_root_nss -- Fraudulent
                    Certificates issued by DigiNotar.nl.
Reference: http://portaudit.FreeBSD.org/aa5bc971-d635-
                    11e0-b3cf-080027ef73ec.html

2 problem(s) in your installed packages found.

You are advised to update or deinstall the affected
                    package(s) immediately.
```

will be quite a lot of new versions for many packages which is not that important, but some of them can (and probably have) security issues and definitely should be updated. You can of course compile them from Ports using `portmaster(8)` but why waste time for compiling, when You can use built every 2 weeks packages from the STABLE branch? The `pkg_upgrade(1)` script from the `sysutils/bsdadminscripts` package will be quite helpful here. It will fetch latest available packages from the STABLE FTP and there is a chance that the security issues will be solved by the newer versions, if not, we are forced to rebuild those packages from source using `portmaster(8)`, but its a lot better and faster to recompile 1-2 packages instead of 30 or more.

As for updating the packages, I generally check them daily, mostly for security issues that would be reported with `portaudit(1)` (from `ports-mgmt/portaudit` package), there are often new versions reported, sometimes even quite lot, but as long as there are `0 problem(s) in your installed packages found`. I do not bother. From time to time I fire up `pkg_upgrade -a -C` to fetch the latest packages from the STABLE branch FTP.

Some of You would certainly ask why use `pkg_upgrade(1)` instead of updating with `portmaster(8)`? Well, for example You have package `z-1.0` installed in Your system, latest package available on the FTP is `z-1.1z-1.2`, so `portmaster(8)` will omit that `z-1.1` package no matter if its newer or not and will force You to compile the `z-1.2` package from the Ports system.

Keeping FreeBSD packages up-to-date in short

- use packages from STABLE that are built every 2 weeks
- use `pkg _ upgrade(1)` to update packages
- use `portmaster(8)` to rebuild packages that have security issues

Example 3. *Solving the security issues by rebuilding the problematic package*

```
% ports-build ca_root_nss

===>>> Currently installed version: ca_root_nss-3.12.9
===>>> Port directory: /usr/ports/security/ca_root_nss

(...)

===>>> Updating dependency entry for ca_root_nss-
                    3.12.11_1 in each dependent port
===>>> Upgrade of ca_root_nss-3.12.9 to ca_root_nss-
                    3.12.11_1 complete

% ports-check
Looking up portsnap.FreeBSD.org mirrors... 5 mirrors
                    found.
Fetching snapshot tag from portsnap2.freebsd.org... done.
Fetching snapshot metadata... done.
Updating from Mon Sep  5 12:07:23 CEST 2011 to Mon Sep  5
                    12:25:09 CEST 2011.
Fetching 3 metadata patches.. done.
Applying metadata patches... done.
Fetching 0 metadata files... done.
Fetching 2 patches.. done.
Applying patches... done.
Fetching 0 new ports or files... done.
Removing old files and directories... done.
Extracting new files:
/usr/ports/devel/p5-File-NFSLock/
/usr/ports/devel/p5-MooseX-Aliases/
Building new INDEX files... done.
New version available: arc-5.21p
New version available: expat-2.0.1_2
New version available: tinyxml-2.6.2
New version available: bash-4.1.11
New version available: gstreamer-plugins-0.10.35_1,3
New version available: gtk-2.24.6
New version available: gtk-update-icon-cache-2.24.6
New version available: libsamplerate-0.1.8_1
New version available: nas-1.9.3
New version available: nettle-2.4
New version available: p5-Date-Manip-6.25
New version available: p5-Mail-IMAPClient-3.29
New version available: p5-XML-Parser-2.41
New version available: xterm-273
New version available: filezilla-3.5.1
New version available: firefox-6.0.1,1
New version available: gtk-oxygen-engine-1.1.2
New version available: nginx-1.0.6,1
New version available: qemu-0.11.1_10
19 have new versions available
auditfile.tbz                                       100% of
                    69 kB   54 kBps
New database installed.
Database created: Mon Sep  5 12:40:01 CEST 2011
0 problem(s) in your installed packages found.
```

Example 4. *Updating the installed packages using STABLE branch*

```
% ports-update
/var/db/uma/FTPINDEX                          100% of
                    21 MB 1139 kBps 00m00s
/usr/ports/packages/All/nettle-2.4.tbz        100% of
                    1082 kB  332 kBps

/usr/ports/packages/All/gstreamer-plugins-0.10100% of
                    4091 kB  942 kBps
/usr/ports/packages/All/gtk-oxygen-engine-1.1.100% of
                    509 kB  339 kBps
/usr/ports/packages/All/filezilla-3.5.1.tbz  100% of
                    3301 kB  232 kBps 00m00s
/usr/ports/packages/All/nas-1.9.3.tbz         100% of
                    487 kB  494 kBps
/usr/ports/packages/All/expat-2.0.1_2.tbz     100% of
                    130 kB  129 kBps
/usr/ports/packages/All/xterm-273.tbz         100% of
                    262 kB  104 kBps
/usr/ports/packages/All/p5-XML-Parser-2.41.tbz100% of
                    184 kB  381 kBps
/usr/ports/packages/All/nginx-1.0.6,1.tbz     100% of
                    225 kB  206 kBps
/usr/ports/packages/All/qemu-0.11.1_10.tbz    100% of
                    12 MB  359 kBps 00m00s
/usr/ports/packages/All/tinyxml-2.6.2.tbz     100% of
                    170 kB  121 kBps
/usr/ports/packages/All/p5-Date-Manip-6.25.tbz100% of
                    1301 kB  681 kBps
===> Update <tinyxml-2.6.1_1> to <tinyxml-2.6.2>
                    (textproc/tinyxml)
=> Update <tinyxml-2.6.1_1> to <tinyxml-2.6.2>
                    (textproc/tinyxml) succeeded
===> Update <p5-Mail-IMAPClient-3.28> to <p5-Mail-IMAPClient-
                    3.29> (mail/p5-Mail-IMAPClient)
=> Update <p5-Mail-IMAPClient-3.28> to <p5-Mail-IMAPClient-
                    3.29> (mail/p5-Mail-IMAPClient) succeeded
===> Update <p5-Date-Manip-6.24> to <p5-Date-Manip-6.25>
                    (devel/p5-Date-Manip)
=> Update <p5-Date-Manip-6.24> to <p5-Date-Manip-6.25>
                    (devel/p5-Date-Manip) succeeded
===> Update <nginx-1.0.5,1> to <nginx-1.0.6,1> (www/nginx)
=> Update <nginx-1.0.5,1> to <nginx-1.0.6,1> (www/nginx)
                    succeeded
===> Update <nettle-2.2> to <nettle-2.4> (security/nettle)
=> Update <nas-1.9.2> to <nas-1.9.3> (audio/nas) succeeded
===> Update <libsamplerate-0.1.7_1> to <libsamplerate-
                    0.1.8_1> (audio/libsamplerate)
=> Update <libsamplerate-0.1.7_1> to <libsamplerate-0.1.8_
                    1> (audio/libsamplerate) succeeded
===> Update <expat-2.0.1_1> to <expat-2.0.1_2> (textproc/expat2)
=> Update <expat-2.0.1_1> to <expat-2.0.1_2> (textproc/
                    expat2) succeeded
===> Update <xterm-271> to <xterm-273> (x11/xterm)
=> Update <xterm-271> to <xterm-273> (x11/xterm) succeeded
===> Update <qemu-0.11.1_9> to <qemu-0.11.1_10> (emulators/qemu)
=> Update <qemu-0.11.1_9> to <qemu-0.11.1_10>
                    (emulators/qemu) succeeded
===> Update <p5-XML-Parser-2.40> to <p5-XML-Parser-2.41>
                    (textproc/p5-XML-Parser)
=> Update <p5-XML-Parser-2.40> to <p5-XML-Parser-2.41>
                    (textproc/p5-XML-Parser) succeeded
===> Update <gtk-update-icon-cache-2.24.5> to <gtk-
                    update-icon-cache-2.24.6> (graphics/
                    gtk-update-icon-cache)
=> Update <gtk-update-icon-cache-2.24.5> to <gtk-update-
                    icon-cache-2.24.6> (graphics/gtk-
                    update-icon-cache) succeeded
===> Update <gtk-2.24.5_1> to <gtk-2.24.6> (x11-toolkits/gtk20)
=> Update <gtk-2.24.5_1> to <gtk-2.24.6> (x11-toolkits/
                    gtk20) succeeded
===> Update <gtk-oxygen-engine-1.1.1> to <gtk-oxygen-engine-
                    1.1.2> (x11-themes/gtk-oxygen-engine)
=> Update <gtk-oxygen-engine-1.1.1> to <gtk-oxygen-engine-1.1.2>
 (x11-themes/gtk-oxygen-engine) succeeded
===> Update <gstreamer-plugins-0.10.35,3> to <gstreamer-
                    plugins-0.10.35_1,3> (multimedia/
                    gstreamer-plugins)
=> Update <gstreamer-plugins-0.10.35,3> to <gstreamer-
                    plugins-0.10.35_1,3> (multimedia/
                    gstreamer-plugins) succeeded
===> Update <firefox-6.0_1,1> to <firefox-6.0.1,1> (www/firefox)
=> Update <firefox-6.0_1,1> to <firefox-6.0.1,1> (www/
                    firefox) succeeded
===> Update <filezilla-3.5.0_1> to <filezilla-3.5.1> (ftp/
                    filezilla)

===> Update <bash-4.1.10> to <bash-4.1.11> (shells/bash)
=> Update <bash-4.1.10> to <bash-4.1.11> (shells/bash) succeeded
===> Update <arc-5.21o_1> to <arc-5.21p> (archivers/arc)
=> Update <arc-5.21o_1> to <arc-5.21p> (archivers/arc) succeeded
===>>> Checking 2bsd-vi-050325_1
===>>> Checking ImageMagick-6.7.1.10
===>>> Checking ORBit2-2.14.19
===>>> Checking OpenEXR-1.6.1_3
===>>> Checking Thunar-1.2.2_2
(...)
```

Some facts about being up-to-date with FreeBSD's packages

- with every RELEASE, packages are built and then they are never updated (even if they have security issues)
- for the STABLE tree packages are rebuilt every 2 weeks

First, we need to install tools that we will use to keep FreeBSD packages up-to-date.

```
# pkg_add -r bsdadminscripts portmaster portaudit
```

Optionally, we can allow users in group *wheel* to perform these task using `sudo(8)sysutils/sudo` package with `pkg_add -r sudo` command first) as they are already allowed to login on the *root* account, we can of course create separate group like *maintainers* that will be allowed to perform upgrades. You will need this line below in `/usr/local/etc/sudoers` file.

```
%wheel ALL=NOPASSWD: /usr/sbin/portsnap, /usr/local/sbin/
pkg_upgrade, /usr/local/sbin/portmaster, /usr/local/sbin/
                    portaudit
```

Here is the most important part, the commands put together into functions that will allow us easy checking for newer versions of the packages, security issues and updating them to newer/fixed versions. The `ports-check` function fetches latest Ports tree, then shows what new packages are available comparing to those installed on the system, next the security issues are checked

Example 5. *Updating only a single package*

```
% ports-update openbox
===> Update <openbox-3.5.0> to <openbox-3.5.0> (x11-
                    wm/openbox)
=> Update <openbox-3.5.0> to <openbox-3.5.0> (x11-wm/
                    openbox) succeeded

(...)

===>>> Checking nss-3.12.11
===>>> Checking open-motif-2.3.3
===>>> Checking openbox-3.5.0
        ===>>> No installed ports depend on openbox-3.5.0
        ===>>> Emptying +REQUIRED_BY file.  Try portmaster -s
```

with `portaudit(8)` and last, the `/usr/ports/UPDATING` file is checked for various messages that can affect us. The `ports-check` (Listing 9) does not rebuild or update any packages, as the name says, only checks.

The second function, ports-update is for updating the packages (Listing 10) using the STABLE branch, it uses `pkg_upgrade(1)` from `sysutils/bsdadminscripts` package, but it will not compile from Ports. You can use it in two ways, without arguments it will update all packages that will be found on the STABLE FTP, but as showed in Example 5 it can be also used only to rebuild several packages that You need to update. It is also possible to specify several packages to be updated, not only one, like that `ports-update openbox nautilus xterm` for example.

The last one (Listing 11) named `ports-build` rebuilds the specified package or all of them if You do not specify one, used mostly to rebuild packages with security issues. This one also can be used without arguments to rebuild all Ports packages or with argument(s) to rebuild only packages that You need to rebuild.

... and thats it generally, I would show some example of these functions usage below.

You will have to put these functions into Your shells startup files, it will be `/etc/profile` for `sh(1)` shell and also for `bash(1)`, `/etc/zshrc` for `zsh(1)` shell. It will not work for C-shells like `csh(1)` since they do not support functions and are retarded in many other ways: *http://www.grymoire.com/Unix/CshTop10.txt*.

Drawbacks

Using this way of keeping the installed packages up-to-date You have to remember two things.

Customized packages. If You built some package with non-default options by compiling it, after upgrade it will *revert* do the default options and You will have to build it again.

Kernel modules. Any package that comes with kernel modules can and probably will break at some point because the STABLE source tree is a *moving target*, that is one of the good reasons to update the base system and then update to latest packages. The packages that have kernel modules are for example `emulators/virtualbox-ose` (VirtualBox), `sysutils/fusefs-kmod` (FUSE implementation) and most notably `x11/nvidia-driver` (binary nVidia graphics driver).

That's all folks! ;)

SLAWOMIR WOJTCZAK

Slawomir Wojtczak (vermaden) is just another busy sysadmin with interest towards UNIX and BSD systems (mostly FreeBSD), often forced to also work on Linux. KISS principle follower. Active troll at many BSD, UNIX and Linux forums.

OWNED!

Quick & Easy Security and Compliance Through RedSphere's Secure Hosting Solutions

- Instantly Become PCI Compliant
- We Address Other Compliance and Industry Security Requirements
  - Penetration Testing Services
  - Source Code Security Review and Design
  - Custom Security Services and Solutions

powered by FreeBSD®

REDSPHERE ©™
Our Promise is Your Peace of Mind

RedSphere Global Security
Call now to speak to a representative
719.924.5266 sales@redsphereglobal.com

www.redsphereglobal.com

Anatomy

of a FreeBSD Compromise (Part 2)

Continuing in our security series, we will look at the ways that "the bad guys" can gain access and what we can do to mitigate this risk.

What you will learn…

- How to plan a security strategy

What you should know…

- BSD administration skills

As mentioned in the previous article, we highlighted some of the reasons why servers and systems are inherently insecure and why it is impossible to 100% secure any system. In this article, we will examine some of the common techniques used to gain control and what we can do to mitigate the risks.

Attack Vectors

What is an attack vector? Simply put, an attack vector is the method by which the hacker gains access to resources or the data stored on the server, PC or traveling across the network. The standard IT definition generally implies attacks which are based on leveraging technology e.g. *Cross Site Scripting attacks* (XSS), malware, phishing etc. but for the the security conscious, this is only part of the story. While it fairly cut and dried that you have been the victim of an attack if you discover malware on your PC, the standard definition doesn't take into account all the circumstances. For instance, if an attacker gained physical access to your PC when the screen was unlocked and copied a sensitive document onto a USB stick leaving the original intact, you would still have suffered an attack. For this reason, the author would prefer to use a much broader definition that includes physical access, social engineering and identity theft rather than the classic *technical* definition as this helps us think like the opposition rather than confining our world view to bits and bytes.

As you look into the world of IT security, you will discover that many of incidents can be categorized into *Why didn't I see this earlier*? group. This is one of the reasons why we should always have respect for the intruder – albeit a grudging one. Often the method of access is just a question of thinking outside the box, sleight of hand or just plain simply leveraging the limitations of the machine that will only do exactly as it is told without ethical or valued consideration. As a Programmer, Systems Administrator etc. it is your responsibility to add the moral dimension – Computers do not have a conscience!

Some Hacking Scenarios

To illustrate then main principles, consider the following hypothetical attacks. In the hands of an experienced hacker or security expert, and provided they have sufficient resources, your data would be compromised in all these scenarios. Can you work out how? While some attacks are more far-fetched than others, all are potentially feasible and I hope this will help you the reader to *think outside the box*. Answers are at the end of the article.

Attack 1

The PC on your desk is not connected to a network and you work in a standard office with windows on the 10th floor. You decide that to aid productivity and security that you will install voice recognition software that not only allows you to write documents quickly, but also allows you to login. All your data is encrypted, and as you do not use the keyboard much, a key logger (software or hardware based) would not be useful to an attacker. The voice recognition software is of military grade standard, and no vulnerabilities are known to exist. All your colleagues are trusted individuals who are highly security conscious, have no idea what you are working on and the office and the windows are locked at night. A highly confidential document is released to the press that only you have been working on and which you have the only copy, and as you are innocent of this betrayal of trust, you must provide your employer with an explanation of how the attacker gained access. Examining the logs, you discover an intruder unlocked your PC at 7:00 AM on a Monday morning, and copied the file using your user-name, password and encryption code without any errors or probing. The office is regularly swept for bugs and mobile phones / recording devices etc. are strictly forbidden. How did the attacker capture your login details?

Attack 2

A senior manager telephones to ask for their password to be reset. The IT help-desk promptly resets the password to the default, the manager then has to change the password on first login. Later that day, the manager turns on his PC after a business meeting and discovers that all of his data has been deleted from his local drive. What has happened?

Attack 3

All employees PC's are attached to a standard corporate network and the chief executive officer sends an email to all staff informing them of a salary increase of 15% taking effect immediately. Unfortunately, the CEO did not write the email. Looking at the email headers, the email has originated from the CEO's PC, but he was not in the office at the time and his user account has not been compromised. How was this achieved?

The Importance of Physical and Environmental Security

While it is taken as granted by most administrators that servers need to be in a secure area preferably in a data-center with fire protection, clean power, temperature control and restricted access, often the rest of the environment is neglected. The authors own pet hate is the way modern buildings are laid out, with the ubiquitous movable *floor box* that provided power and CAT 5/6/7 services. Meant to be an improvement over surface mount trunking and *ceiling to desk* arrangements, they are in the authors opinion a easy point of access to the infrastructure. With metal / PVC trunking the cabling could be secured by adding security screws to the trunking cover easily, and as any switches etc. would be in locked cabinets, it was just a question of visually ensuring that no extra devices were plugged into the network. Most of these floor boxes are not secured properly to allow the buildings maintenance department easy access, and even if they were secured, the actual floor panels themselves are not. In many data-centers due lack of room or convenience the author has seen switches or devices located under the false floor, and this principle could be extended easily by the unscrupulous into the office space. While getting access to main power is not recommended using a vampire tap (Fried hacker anyone?) splicing a tap into a CAT5 cable takes about 15 minutes with a box cutter, an IDC punch-down tool, some connectors and a pair of wire cutters. If the intruder was sufficiently clever, by using a passive tap his data capturing device would not be visible on the network. With the battery life of mobile devices, many hours worth of data could be captured without raising suspicion. Probably the biggest challenge is getting the carpet tiles to lie flat of the floor afterwards without showing tell-tale movement.

The next problem is the CDROM / USB devices and key-loggers. Once the attacker has physical access to the device, many possibilities exist for capturing un-encrypted data. While users should be encouraged to store data on a network drive, quite often sensitive information is stored locally on PC's hard drives and this is easily accessed across multiple file-system types using a utility such as Trinity if the partitions or drives are not encrypted. The local administrator password for the most popular O/S can be easily reset using such a utility which can be booted from CD or a USB stick. The benefit of using the USB method is that data can easily be copied across from the partition to the stick, and provided no damage is done to the data on the local drive the theft is virtually undetectable unless the culprit is physically caught in the act or on CCTV etc. This could potentially be used to gather data pertaining to identity theft even if the user data is stored on the network, as browsers can save login passwords to Internet sites etc. Depending on the ingenuity of the browser developers, it may be possible to tie this data to the CPU ID (If the PC is Intel based) and thereby circumventing this attack, but to the authors knowledge no such mechanism exists. The other common form of attack is the key-logger, which records all keystrokes generated by the end user. With the growth if the Internet, software key loggers seem to be in abundance, but physical devices are easily obtainable as well. The level of threat here is obvious, all input to the PC including user-names, passwords, document and email text etc. is exposed. Good PC client software that monitors USB usage (and abuse) is available, but consideration should be given to whether or not users should be allowed access to USB at all. Unfortunately, it is easy enough just to install a USB hub and as most PC's nowadays use USB keyboards and mice, even the old trick of disabling the USB ports by disconnecting the cable header from the motherboard (or as somebody suggested filling the sockets with epoxy resin) will be in vain unless the PC is installed in a cage or cabinet that physically limits access to the ports. Next, we have video grabbers that will intercept and capture screen shots onto a separate device. There is even a replacement video cable available that contains a screen grabber that is virtually indistinguishable from a standard cable. Finally, we have a variation of the Ethernet tap scenario, just plug in a mini-hub or switch between the PC and the network. While this may not reveal a lot of corporate data as switches are now intelligent enough to be zoned and virtualized, data on that particular segment would be exposed.

Cultural Security

This is the habitation of the social engineer, the hacker with people skills. Forgetting for the moment the electronic side of social engineering (e.g. phishing, malware, fraudulent password requests etc.), let us focus on the the human side of the attack. The boundaries are not always clearly defined and sometimes little effort is required on the part of the hacker to manipulate the setting to his advantage (e.g. if you know your victim is an avid Chicago Bulls fan, there is a good chance their password is *Basketball*). Most of the attacks here are facilitated through lack of suspicion, naivety or human weakness (e.g. unsolicited spam offering a financial reward to *help* a third party smuggle money out of a foreign country) on the part of the victim, and often the victim doesn't realize the extent of the fraud until it is too late. Sometimes the victim is just a *patsy* for the attackers true target, e.g. obtaining a high level technical support password to gain access to a different users data. Sometime though, these attacks have the co-operation of an insider (e.g feeding credit card details to somebody outside the organization) and as the insider will be deliberately defrauding the company, they will take many steps to ensure that they are not discovered and can lay dormant for quite some time unless rigorous processes are in place to mitigate the risk. More often than not, genuine mistakes will be be admitted to by staff, but the *smell test* can be a good gauge in certain circumstances. If a database is compromised and there is absolutely no evidence whatsoever of a software based attack, carelessness, or external compromise the security officer is left with a problem. While there are a number of techniques that can be used to track and analyze data and access, it is always very difficult to come to terms with the fact that an employee is potentially committing a crime, and great care needs to be taken to ensure that the innocent are not blamed or their characters maligned. This is generally why companies and institutions hush-up internal fraud, not only are they worried that their reputation will be damaged, but the majority of their audience will not appreciate the complexity of actually pursuing fraud within a large organization. Risk can be mitigated by allowing access to systems on a *need to know* basis, and regularly performing audits, but even then dissatisfied employees, poor working relationships and a corporate *Laissez-Faire* attitude to security will not make matters easier. Sometimes though in the case of internal fraud, the perpetrator fails the *smell test* on such issues as severe financial difficulties, relationship problems or a personal habit out of control. In such circumstances Law Enforcement will be your best ally, but is is essential to base investigations on the basis of fact not just circumstantial evidence.

When the attack comes from the outside, it can be commercially very difficult to quickly isolate the source of the problem. As social engineering is effectively a con-trick and many businesses are based on trust and working relationships, the best mitigation is education. Most end users will happily allow a colleague to plug a USB pen drive into their PC but if this was disallowed (or physically restricted) the next question is would they allow an external consultant or *expert* with whom they have no obvious trust relationship? Most people lean towards submission to those in authority, and provided the hacker can project sufficient authenticity, it is an almost automatic social response to comply. Kevin Mitnick, the reformed hacker, was quite open about this – it is often easier to gain access via social engineering than actual hacking. Education, auditing and testing therefore are the best policies as well as validation and cross-referencing security sensitive requests. One of the problems of large anonymous corporations is that there is no easy way to prove that Joe in accounts actually is Joe in accounts. Unless systems are put in place to close the loop and prove that Joe actually is Joe, then the social engineer will always have the upper hand.

Software Security

This is the classic interpretation of an attack vector, and while there are many thousands of exploits available, they really can be split into 4 main conceptual categories. Sometimes, an attacker will take advantage of more than one weakness, and unless a system is very poorly secured, often the attacker must use more than one technique to gain access. For instance, the attack on my FreeBSD box would fall into categories 2 & 3.

Poor System Configuration

This is probably the most obvious hole in security to patch, and covers a multitude of sins from password-less accounts to unused services running on a server. Apart from good practice, eliminating the obvious reduces the available footprint for attack. Certain legacy applications and services (e.g. telnetd) are so inherently insecure they should never be run at all.

Software Performing As It Should Do

This might seem paradoxical, but almost all distributed denial of service attacks (DDOS) take advantage of this premise. Your web-server is there to serve pages, so the attackers mentality is *How can I break this by loading the server outside the performance envelope*?. This can be mitigated by good systems architecture, load balancing, security patches and throttling, but in distributed environments like the Internet if your web-server is attacked by a bot-net (A network of compromised machines) it is extremely difficult to separate legitimate from the Illegitimate traffic, especially if the bot-net is well distributed. The 4.5 million-strong Alureon bot-net is a good example, both bandwidth and servers would be unable to cope under a sustained attack. The DDOS is a fearsome weapon, and while you can mitigate you cannot totally prevent failure under these circumstances. The best you can do is monitor your traffic patterns and take evasive action if an obvious attack takes place.

Bad Design

It is always good for a lively discussion with developers whether a piece of code has a *bug* or a *feature*. This extends right through the Operating System, and virtually no software is immune. With the best will in the world, a developer will write a quality piece of software, dependent on external libraries or the O/S written by others and thereby introducing problems outside of their control. For instance, the C function call strcpy is not bounds checked, and is a common vector for buffer overflow attacks. While good programmers employ *defensive programming* (e.g. writing code that will cope with the unexpected such as parameters out of bounds etc.) sometimes the law of unintended consequences comes into play. HTML based forums are an obvious example, it is a good idea to give the user the ability to add HTML to their post, but what happens if this is malicious Javascript or an XSS attack? How do we separate *Good functionality* from *Potential* *exploit*? While I believe these problems can be resolved to a degree through adequate testing (e.g. fuzzers, pen testing etc.), ultimately as technology advances more vulnerabilities will be exposed. This is where developers have a responsibility to play, and why the author will always lean strongly towards Open Source and *BSD etc. Peer review, an openness about security issues and desire to get it right have been strong driving forces in free software community, and it is so apparent the commitment to both security and quality. On the other hand, Commercial vendors work on a purely *need to know* basis, and often essential patches are not made in a timely fashion as it not commercially viable or would break too many things in the progress. When the author started working in an electronics factory in the late 70's, *Automated Test Equipment* (ATE) was not available and failures were high. Now, ATE is the norm and the failure rates are inconsequential. If we are to look at the security issue from a different perspective, the hackers bot-net is a brute-force software equivalent of ATE – using technology to test technology to destruction. This is where free software has the upper hand over commercial offerings, we have the flexibility and the ingenuity to respond creatively to the threat and to lead the field. While others refuse to accept that security is everybodies problem, the problem will persist and like the emperor with no clothes, they themselves will be exposed.

Answers

Attack 1

Using a laser based microphone, the attacker scanned your office window to capture the office conversations. It was then a matter of using social engineering to persuade the office cleaner to allow him access to office. To re-enforce his case, he bluffed the cleaner by *proving* he could gain access to you voice operated PC. Unfortunately, the voice recognition software did not pick up on his regional accent, but it did an excellent job of decoding the correct words and phrases. Using a special refractive film or glass on the windows would have stopped this attack dead (as used by certain security agencies and embassies).

Attack 2

The IT help-desk did not have sufficient identity controls in place. The attacker used the managers phone while he was out of the office, and requested a password reset. As support did not identify the manager other than his extension number on the telephone system, he was not challenged. Once access was gained, the attacker securely deleted the files on the local drive. This could be mitigated by giving each user an individual help-desk password only known to themselves and IT.

Reference

- *http://www.argoasecurity.com/product_detail.aspx?productID=880* – Professional laser microphone
- *http://www.ehow.com/how_5969176_make-laser-listening-device.html* – DIY proof of concept of the above
- *http://www.janitha.com/archives/146* – Ethernet taps
- *http://www.keydemon.com* – Key logger
- *http://www.keydemon.com/tiny_frame_grabber/* – Video logger
- *http://trinityhome.org* – Trinity Rescue CD
- *http://www.intel.com/support/processors/pentiumiii/sb/CS-007579.htm* – Intel unique CPU ID
- *http://www.pc-safe.co.uk* – PC security cases
- *http://arstechnica.com/security/news/2011/07/4-million-strong-alureon-botnet-practically-indestructable.ars* – Alureon Botnet
- *http://www.techrepublic.com/article/protect-your-apache-server-from-dos-attacks/5058830* – Securing Apache. This is an old article, but has many valid points still

Attack 3

The SMTP server was not secured with password authentication. By spoofing the MAC and IP address of the CEO's PC, the attacker was able to telnet onto port 25 of the email server and spoof the email conversation. This is also easily achieved on many public ISP networks, spoofing email between addresses on the same domain is trivial. While the solution is easy to fix on internal email servers, it is a major problem on the Internet.

Please Note

The information in these articles is designed for system administrators to help improve systems security. While testing and discovering vulnerabilities in your own personal systems for the purpose of improving security is perfectly legal in the UK, legislation is different in other parts of the world. Employers may take disciplinary and / or legal action against employees if these tools are used in the workplace without permission. Using these tools against third-party systems without permission is not only considered unethical, but is also illegal in many countries. The author and BSD Magazine do not condone unethical or illegal hacking.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Your FreeBSD & PC-BSD Resource

www.FreeBSDMall.com

FreeBSD 8.2 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 8.2 CD	$39.95
FreeBSD 8.2 DVD	$39.95
FreeBSD 7.4 CDROM	$39.95
FreeBSD 7.4 DVD	$39.95

FreeBSD Subscriptions

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

FreeBSD Subscription , start with CD 8.2	$29.95
FreeBSD Subscription, start with DVD 8.2	$29.95
FreeBSD Subscription, CD 7.4	$29.95
FreeBSD Subscription, DVD 7.4	$29.95

PC-BSD 8.2 DVD (Hubble Edition)

PC-BSD 8.2 DVD	$29.95
PC-BSD Subscription	$19.95

BSD Magazine

BSD Magazine	$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 8.2	$79.95

The FreeBSD Bundle

Inside the Bundle, you'll find:

- FreeBSD Handbook, 3rd Edition, Users Guide
- FreeBSD Handbook, 3rd Edition, Admin Guide
- FreeBSD 8.2 4-disc set
- FreeBSD Toolkit DVD

☆ Special: The FreeBSD CD Bundle	$99.95
☆ Special: The FreeBSD DVD Bundle	$99.95

The FreeBSD Toolkit DVD $39.95

FreeBSD Mousepad $10.00

FreeBSD Caps $20.00

PC-BSD Caps $20.00

For *MORE* FreeBSD & PC-BSD items, visit our website at FreeBSDMall.com!

CALL 925.240.6652 Ask about our software bundles!

Elephants in Prato

On November 25th a group of PostgreSQL community members met together in Prato for the 5th Italian PostgreSQL Day (PGDay).

It took more than six months for the *Italian PostgreSQL Users' Group* (ITPUG) to organize the event, that even if quite young (the first conference was organized in 2007), it has quickly become a must for the Italian community.

After two years of vacancy from Prato, the organizers have decided to come back to the place were all started five years ago: the Monash University Center in Prato. Not that the previous two editions, respectively in Pisa and Rome, were unsuccessful, but organizers felt more comfortable in Prato due also to the fact that a lot of the most active members of ITPUG live near this city.

A Quick Summary of the Event

Everything started the night before the event with a great dinner based on local foods (in particular the well known Fiorentina steak) and wines. In a non formal atmosphere you could have seen members of the Italian and international communities, with different skills and backgrounds, all together at the same table. This is a thing organizers are very proud about, since the event aimed not only to promote PostgreSQL as a product, but also as a community.

The day after, the conference took place and the atmosphere became a little more formal. Each attendee, after a registration, was able to choose among two parallel sessions: one more oriented to database administrators (DBAs) and one more oriented to developers.

ITPUG staff, through its president Gabriele Bartolini, welcomed the guests and introduced the conference. Before leaving the microphone to technical guests there were a few words by local government authority, who emphasized how much important the Open Source movement has become in Italy and how much it is encouraged day by day.

The keynote speech was performed by Magnus Hagander, president of the PostgreSQL Europe, who presented all the new features of the new born release 9.1, as well as some of the ongoing development for the next 9.2 version. The speech was held in the superb Grollo room, which fascinated all the guests with its elegance.

After the keynote, it was time for a coffee break, that allowed guests to relax, talk and visit the elegant Monash University and its very elegant rooms.

And then the real conference begun. Speakers alternate themselves in a full time schedule along the two parallel sessions: the database administrators track was held in the Grollo room, while the developers track in the Venice room.

Describing each talk one by one is out the scope of this article; people interested can download slides and material from the conference official Web site (see References).

Just for the record, during the all day there were talks about PostgreSQL's future, the long debate NO-SQL versus relational database (sometimes referred as YES-SQL), backup and stream replication tools, tutorials on how to efficiently use Common Table Expressions (CTEs) and on how to access foreign data tables from a PostgreSQL instance. There was also some talk about security, in particular on how to store password inside PostrgeSQL in a secure way. Other talks presented a few ways to connect to a PostgreSQL database from foreign programming languages or how to write new contrib modules for the database itself in one of such languages. Last but not least, a few talks about the adoption of

PGDay by Numbers

The PGDay had almost 100 attendees, which is a quite big number considering that is around the half of the number of participants of the European PostgreSQL Conference. There were 25 talks presented (including the lightning talks) for about 15 hours of speech.

PostgreSQL in several contexts, from the enterprise level to the local government one.

The morning ended with a great buffet lunch, catered in one of the Monash University room. Besides having time to relax and enjoy very good local meal, this break was an opportunity for guests to meet, talk and spend some quality time together. As an organizer, it was really nice to walk around seeing all these people talking together, discussing problems and solutions, presenting products, and so on. In a few words, exchanging experiences. I strongly believe this is the kind of event that makes Open Source movement really great, because it works as a people attractor. And the most people a project can attract, the most the project can evolve.

The lightning talk session, performed after the lunch, has been really successful with a lot of presentations and quick description of experiences. The rule of the lightning talks is simple: you have no more than 5 minutes to present any argument you want, and then you'll be kicked off. While almost all lightning talks were technical, it was nice to see a few non-technical either. In particular a few speakers presented summaries about past events and/or their experiences on how they became PostgreSQL contributors, at any level from the document writer to the code committer. I believe this kind of talks have to been promoted much more, because people often is scared to start contributing to a project, feeling he will not be able to do a good work or feeling like under constant evaluation. Instead, as pointed out during the lightning talks, everyone can start contributing at almost every level, and the more you will contribute, the more you will become expert on the system and the more you (hopefully) will become addicted by the project!

After the lightning talk session the conference split again in the two parallel tracks, that concluded in the Grollo room later for the closing session. I had the honor, to officially end up the conference. I gave a quick talk to all the audience, first of all thanking all the ITPUG staff and volunteers that helped us making a great event, then thanking also all the attendee for their presence. As usual, I talked about the need for PostgreSQL and ITPUG to constantly gain new active members that can help supporting both the project itself and the Italian community. The hope, of course, was to see again each other the next year at the new PGDay!

And since it is a tradition the PGDays, while the conference officially ended with my talk, the community event did not. And a few minutes later, almost all participants and organizers met together again at a pub, drinking great Guinness beer offered by 2ndQuadrant, a gold sponsor of the event. I was nicely surprised about how many attendees did not miss this off-conference meeting, proving again how much is important for users, developers, passionates to stay together and talk about their favorite database.

At the end of the conference it was time for the organizers to evaluate the event, that was of course of a great success. Interestingly, the number of enterprises which participated to the event has raised constantly during the past editions, and this year it was strongly evident how such enterprises are no more looking at PostgreSQL as an alternate database to some proprietary solution, but are basing their mainstream solutions on PostgreSQL.

As a final note, if you have the possibility, I suggest you to not miss any further PGDay in Italy. You will be glad about the organization and the comfort of a such community based technical event!

History of PGDay(s) at Glance

The first official Italian PGDay was organized by a bunch of volounteers and passionates, including the author, in 2007. At that time ITPUG did not exist at all. The conference catch the attention of several other PostgreSQL related communities, including Japan PostgreSQL Users' Group (JPUG), and had a lot of special guests including members of the core team. This quickly lead the organizers to extend the conference from one day only to a two days event.

Galvanized by the success of the conference, the organizers decided to create the Italian PostgreSQL Users' Group, an entity that promotes PostgreSQL in Italy giving support to users and factories, organizing events, promoting the translation and the development of PostrgeSQL related tools and so on.

The second PGDay, run by ITPUG, was a two day conference that act both as Italian and European official meeting. It was also the event that gave the kick off for the PGeu, the European committee that works similarly to ITPUG but at an European level.

All the other subsequent events have been marked explicitly as Italian, but this does not mean that were not foreign members at all the past editions. And in fact, PGDay-Italy is today a well known event that attracts every year members of the international community and member of the core team.

References

- PGDay 2011 official Web Site: *http://www.pgday.it*
- ITPUG official Web Site: *http://www.itpug.org*
- PostgreSQL official Web Site: *http://www.postgresql.org*

LUCA FERRARI

Luca Ferrari lives in Italy with his wife and son. He is an Adjunct Professor at Nipissing University, Canada, a co-founder and the vice-president of the Italian PostgreSQL Users' Group (ITPUG). He simply loves the Open Source culture and refuses to log-in to non-Unix systems. He can be reached on line at http://fluca1978.blogspot.com.

Keep FreeBSD Free!

Support FreeBSD by donating

The FreeBSD FOUNDATION

The FreeBSD Foundation is a 501(c)(3) non-profit organization that is committed to supporting and building the FreeBSD Project and community worldwide. Founded in March 2000 to fill the need for an outside organization that could support the community's vision and growth, The FreeBSD Foundation exists to serve the FreeBSD community world wide.

The Foundation's fund-raising efforts are essential to keeping FreeBSD free. Private donations fund 100% of the FreeBSD Foundation's efforts.

Join the growing list of donors and users of FreeBSD

SUPPORTING FREEBSD SINCE 2000! 10+ YEARS

To find out more, please visit our Web site:

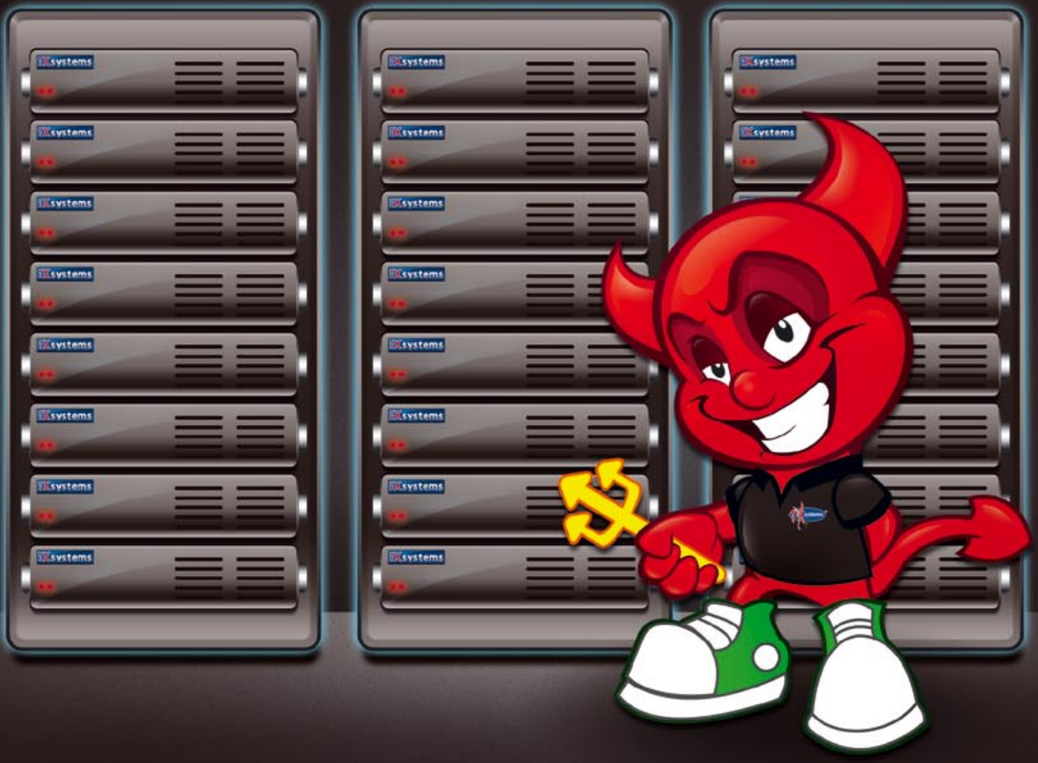
www.freebsdfoundation.org

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

BSD MAGAZINE

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

What has your server vendor done for BSD lately? Probably, not much.



Work with a vendor that supports the operating system you love!

iX is the corporate sponsor of the PC-BSD® Project, a major corporate donor to the FreeBSD Foundation, and leads the FreeNAS™ development team -- all while employing some of the most brilliant minds in the FreeBSD® community. For BSD hardware and software expertise, look no further.

1-855-GREP-4-IX
http://www.iXsystems.com/community

iX systems®