

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

JAILS FIREWALL WITH PF

INSIDE

FREEBSD PROGRAMMING PRIMER: WRITE HTML, CSS, PHP, AND SQL CODE

USEFUL UTILITIES FOR PF

FREEBSD JAILS FIREWALL WITH PF

IMPROVEMENTS TO JAIL MANAGEMENT VIA THE WARDEN

SPIDERFOOT 2.0: THE OPEN SOURCE FOOTPRINTING TOOL

DTRACE: A DEEPER APPROACH

MSEARCH: MIDNIGHTBSD SEARCH

VOL.7 NO.5
ISSUE 05/2013(46)
1898-9144



855-GREP-4-IX
www.ixsystems.com
Enterprise Servers and Storage
for Open Source



- ✓ Rock-Solid Performance
- ✓ Professional In-House Support

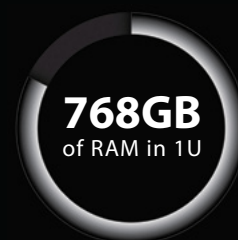
E5-2600

High Performance, High Density Servers for Data Center, Virtualization, & HPC



MODEL: iXR-22X4IB

<http://www.iXsystems.com/e5>



KEY FEATURES

iXR-22X4IB

- Dual Intel® Xeon® Processors E5-2600 Family per node
- Intel® C600 series chipset
- Four server nodes in 2U of rack space
- Up to 256GB main memory per server node
- One Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node
- 12 SAS/SATA drive bays, 3 per node
- Hardware RAID via LSI2108 controller
- Shared 1620W redundant high-efficiency Platinum level (91%+) power supplies

iXR-1204+10G

- Dual Intel® Xeon® Processors E5-2600 Family
- Intel® C600 series chipset
- Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
- Up to 16 Cores and 32 process threads
- Up to 768GB main memory
- Four SAS/SATA drive bays
- Onboard SATA RAID 0, 1, 5, and 10
- 700W high-efficiency redundant power supply with FC and PMBus (80%+ Gold Certified)

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX** | www.iXsystems.com

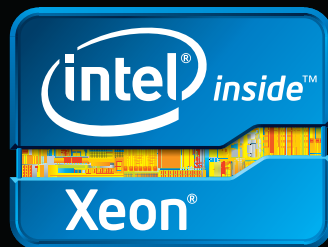
High-Density iXsystems Servers powered by the Intel® Xeon® Processor E5-2600 Family and Intel® C600 series chipset can pack up to 768GB of RAM into 1U of rack space or up to 8 processors - with up to 128 threads - in 2U.

On-board 10 Gigabit Ethernet and Infiniband for Greater Throughput in less Rack Space.

Servers from iXsystems based on the Intel® Xeon® Processor E5-2600 Family feature high-throughput connections on the motherboard, saving critical expansion space. The Intel® C600 Series chipset supports up to 384GB of RAM per processor, allowing performance in a single server to reach new heights. This ensures that you're not paying for more than you need to achieve the performance you want.

The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® Processors E5-2600 Family, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic webservers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® Processors E5-2600 Family, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 Family and the high throughput of Infiniband.



iXR-1204+10G: 10GbE On-Board



iXR-22X4IB

Intel, the Intel logo, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

MAGAZINE BSD

Dear Readers,

May issue of BSD Magazine is dedicated to security matters with the use of Open Source solutions. On the following pages, you will find articles about Packet Filter, Jails and tools for troubleshooting, scanning, and text search.

We start with Rob's column, where he will discuss the matter of property laws and how it happens that good solutions are beaten by technically less advanced ones and perish.

Next, we announce the second release of SpiderFoot— the tool for spidering web pages. Its author, Steve Micallef, will explain its features, installation process, and simply how it works.

In the Get Started section, Michael shows step by step how to configure the firewall to only allow specific traffic to the service jails.

This month's Dev Corner covers PC-BSD and MidnightBSD. Kris will teach you more about jail management with Warden and how to create jails via Hostname / Nickname and change and assign IP addresses on the fly. Meanwhile Lucas will introduce you to msearch – a full text search tool, that offers users the ability to search against filenames or contents of text files.

Then, Dru explores some of the third-party utilities which are available to help you analyze the log and state table of a PF firewall.

Next, we have the fourth part of Rob's series on FreeBSD Programming Primer. This time, sysadmins have an opportunity to learn how to configure a development environment and write HTML, CSS, PHP, and SQL code.

In May 2012, we published the article "Intro to Dtrace" by Carlos Antonio Neira, where he explained the system configuration to enable DTrace probes and some of this tool's features. A year later, he comes back with a much deeper approach...

We hope you will enjoy this issue and find many interesting articles!

Patrycja Przybyłowicz
Editor of BSD Magazine
& BSD Team

Editor in Chief:

Ewa Dudzic
ewa.dudzic@software.com.pl

Supportive Editor

Patrycja Przybyłowicz
patrycja.przybylowicz@software.com.pl

Contributing:

Rob Sommerville, Steve Micallef, Michael Shirk, Dru Lavigne,
Carlos Antonio Neira

Top Betatesters & Proofreaders:

Ahmed Aneeth, Radjiss Mahangoe, Barry Grumbine,
Bjørn Michelsen, Paul McMath, Eric Geissinger,
Eric De La Cruz Lugo, Imad Soltani, Luca Ferrari,
Ewa Duranc, Annie A. Zhang, Ben Milman, Lisa Liang

Special Thanks:

Denise Ebery
Matt Olander

Art Director:

Ireneusz Pogroszewski

DTP:

Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

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

Advertising Sales:

Patrycja Przybyłowicz
patrycja.przybylowicz@software.com.pl

Publisher :

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

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

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

Mathematical formulas created by Design Science
MathType™.

Let's Talk

06 Whose Idea is it Anyway?

By Rob Somerville

With Apple fallen from grace as the world's most valuable company, how can large technology-based companies succeed? The current trend for Intellectual Property laws can only increase the speed at which the race is towards the bottom...

What's New

08 SpiderFoot 2.0: The Open Source Footprinting Tool

By Steve Micallef

The original version of SpiderFoot was created in 2005 with the goal of being a freely available open source tool for footprinting an Internet domain name. Version 2.0 was released May 2013 and is completely re-written in Python with loads of new functionality and is now highly extensible. The target user-base is penetration testers, system administrators and security enthusiasts who wish to gain a better understanding of what a domain name's Internet footprint looks like.

Get Started

12 FreeBSD Jails Firewall with PF

By Michael Shirk

Features are available for fully virtualizing FreeBSD jail networking (as of FreeBSD 8.x). The code has improved in the current 9.x code base but to get a jail up and running with the current install, pf provides the necessary functionality to firewall off multiple jailed services. This article will cover basic jails configuration to highlight how to configure the firewall to only allow specific traffic to the service jails.

Developer's Corner

16 Improvements to Jail Management via the Warden

By Kris Moore

Over the past few months, several exciting new features have been added to the Warden which greatly improve jail management on FreeBSD & PC-BSD systems. Now the Warden will be able to create jails via Hostname / Nickname, and change and assign IP addresses on the fly. This greatly simplifies jail creation via the command-line, allowing you to create the jail and then set addresses as needed later.

18 msearch: MidnightBSD Search

By Lucas Holt

MidnightBSD search, or msearch, is a full text search tool. It offers the user the ability to search against filenames or contents of text files. msearch is not meant to replace other tools like find, locate, or whereis. From this article you will learn the basic usage of the msearch tool and the reason why it was written.

How To

20 Useful Utilities for PF

By Dru Lavigne

PF is a stateful firewall, meaning that it tracks the state of existing connections in a state table, allowing the firewall to quickly determine if packets are part of an established connection. PF also provides a logging facility and the firewall administrator controls which packets get logged by including the log keyword in only the firewall rules which should be logged when matched. This article explores some of the third-party utilities which are available to help you analyze the log and state table of a PF firewall.

Admin

28 FreeBSD Programming Primer – Part 4

By Rob Somerville

In the fourth part of our series on programming, we will continue to develop our CMS. Here we will examine how a modern CMS dynamically generates and controls content and implement a similar model in our PHP code. From this article you will learn how to configure a development environment and write HTML, CSS, PHP, and SQL code.

Tips & Tricks

38 DTrace: A Deeper Approach

By Carlos Antonio Neira

The author of the article "Intro to DTrace", published in May 2012 in BSD Magazine, has described DTrace all the way from configuring your system to enabling DTrace probes to the point of executing some D scripts to show you some DTrace features. This article will take a deeper approach on DTrace.

Whose Idea is it Anyway?

With Apple fallen from grace as the world's most valuable company, how can large technology-based companies succeed? The current trend for Intellectual Property laws can only increase the speed at which the race is towards the bottom.

Technology is a funny beast. You'd think that inventing the best software, the most innovative user interface, developing a commitment from your customer base that is almost religious in its zeal would be enough, but no. The market – and the technology marketplace in particular – is fickle, yet the proponents of draconian Intellectual Property (IP) rights fail to grasp this fact. What is the latest de rigueur soon becomes passé as not only the technology evolves, but customer expectation rises. The paradox is this: while it takes a tremendous amount of financial investment to develop new technology, the returns are often quite random and defy logic and statistical analysis. Take Betamax over VHS for example. Technologically VHS was not as advanced as Betamax, yet the underdog won the battle by having the support of the entertainment industry (partly due to the extra recording time VHS provided) and reaching the tipping point in the marketplace before Sony could roll out a 2 hour version. Result? The company that brought the transistor radio and broadcast quality kit to the world was sorely undermined by a more efficient but less innovative manufacturer.

Now it could be argued that this is a strong basis for IP law, but the problem fundamentally remains – who has the right to an idea? Even more importantly, who has the right to lay sole claim to something that will bring major benefits to mankind? Throughout history there seems to be this “universal consciousness” where ideas arrive via the zeitgeist and monumental battles arise as to who has the best format, original concept, or design. Take Edison versus Tesla for example. Time and time again, the lone inventor is an endangered species when exposed to the power and force of the marketplace. Likewise, a multinational attempting to cling on to success based upon a single idea or philosophy is futile – yesterday's success is no guarantee of tomorrow's profitability. The success of the IBM PC was arguably not down to IBM's innovation, good design, or the fact that they were a market leader – it was the sweat shops in Asia producing clones untouched by Western patent law that blew the market right open. Of course, IBM having its fingers severely burned jumped on the IP bandwagon with Micro-chan-



nel, restricted developers by implementing a licensing policy and guess what? MCA was dead in the water. Even Compaq tried with Extended Industry Standard Architecture (EISA) but they could not overcome the juggernaut that the Industry Standard Architecture (ISA) had become.

Let's play devil's advocate with the whole philosophy of IP. I am paid by my employer to write code, solve problems, and innovate. Any ideas I come up with and any code I write belongs to my employer. That's fair enough in a 9 to 5 environment. However, being the type of person that I am (incurable pedant), if my employer has a problem or my code doesn't do what it says on the tin, I will worry about it. I will want to improve it. I am like a dog with a bone. And that means thinking about it – on the journey home, in the bath, when I wake up in the morning. My wife is witness to me sitting bolt upright in bed at 2:00 AM yelling "You need to compsurf that drive" before settling

down to a more passive stage of unconsciousness. Now, I subconsciously solve the problem in a moment of revelation when I least expect it at 4:00 AM.

Who has the intellectual property on that? According to the lawyers, I am supposed to challenge my employer and say it was my idea but as it was outside of my contractual hours, I cannot share it with them. Or maybe not. The suggestion is ludicrous, unethical, and prohibitive, yet this is where IP is driving the innovators and the creatives. I understand the dilemma that is at the heart of IP – reward and recognition. A good workman is worth his wages, and credit where credit is due. How can we restore the value of the innovators, those that successfully think outside the box, in a society where everybody is a winner? How can large organizations profit yet at the same time protect their investment? Certainly the digital age brings huge challenges

in this regard. It takes little cost or effort to copy software, a customer database, or in the case of Wikileaks, state secrets. We live in an age where technology is demolishing all the boundaries and traditional rules of ethics and conceivably the universe. I cannot clone a car in the time it would take me to clone a credit card, yet potentially the amount of profit I could make from this (albeit illegally) is potentially more than the value of a car that would take one individual month – if not years – to replicate. What is valuable now – information and power – hasn't changed, but the medium and how it is delivered and extracted has.

The last time we had a technological revolution on such a scale, we were living in the 1400's. It could be reasonably argued that the Protestant Reformation was a direct consequence of Johannes Gutenberg and the printing press. The established rule crumbled, and the renaissance brought enlightenment and a much needed freedom of information exchange. Part of the reason for this explosion in knowledge was ironically due to the way information was disseminated prior to the black death – monks in monasteries were responsible for producing books, and the church was anxious to control what was acceptable. The plague reduced the ability to produce books efficiently, and from an economist's point of view, the printing press filled that market need.

Large organizations, like large groups of people – don't like change. The flexibility of the small or medium sized company far outweighs that of the established behemoths. All large technology companies must face the fact that they are not immortal or omnipotent, as history proves time and again. It's that fickle marketplace again. Redhat has made major inroads into powering major financial institutions, yet its share price remains a fraction of Apple Inc. The fact that a business model based on Open Source can breach the bulwark of the capitalist business model should be a wake up call to those that believe that the traditional rules still apply. Technology makes a great slave but a terrible master. We live in interesting times.

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.

SpiderFoot 2.0

The Open Source Footprinting Tool

The original version of SpiderFoot was created in 2005 with the goal of being a freely available open source tool for footprinting an Internet domain name. Back then, it was written in C# and only ran on the Windows platform with fairly limited functionality. Version 2.0 was released May 2013 and is completely re-written in Python with loads of new functionality and is now highly extensible.

What you will learn...

- What is footprinting, and why is it used?
- What does SpiderFoot do, and how can it be of use to you?
- How to install and use SpiderFoot

What you should know...

- A basic understanding of TCP/IP and how the Internet works would help, but is not really essential.
- If you're using SpiderFoot on Linux or *BSD, basic knowledge of Python might help

The target user-base is penetration testers, system administrators and security enthusiasts who wish to gain a better understanding of what a domain name's Internet footprint looks like, and perhaps where there may be undesirable information leakage from that domain.

What is Footprinting?

In a generic sense, footprinting is the process of understanding as much as possible about an entity. In the context of the Internet and specifically SpiderFoot, that entity is a DNS domain name, for instance, Google.com. Some people interpret footprinting as port scanning, others as spidering web pages and so on, but what constitutes a "complete" footprint is completely open and can actually change over time.

If you consider what the Internet looked like in the year 2000, the footprint of an Internet domain name would have included hostnames/sub-domains, IP addresses, open ports, and others, but it would not have included anything about social media presence. In the same vein, the Internet is continually evolving with the addition of rich data sources that provide a wealth of information about Internet entities that were not available previously or only offered in unstructured form. A lot of that has since changed, not only resulting in more widely available data, but also data available as web services, thus making its collection and analysis more automatable.

How is it Done?

The most basic data source for footprinting is the website of the entity itself. Simple things like e-mail addresses, hostnames/sub-domains, web server versions, web server technologies, and much more can be gathered simply by fetching web pages from the target, following links, performing some regular expression checks, and analysing HTTP headers.

But the real power of footprinting is combining data from one activity with another to come up with a bigger picture. A simple example is performing a DNS lookup of the entity's domain name to get the IP address, then looking up the IP address in an Internet address registrar (for example, RIPE, ARIN or APNIC) and from there, determining whether the entity owns the entire network range that the IP resides on. Then, armed with that information, you can port scan, banner grab, and so on in order to add to your footprint and in turn use the information obtained there (hostnames, software versions, and other data mentioned in connection banners is one example), to build it up further.

Why Footprint?

Footprinting is not an academic exercise; it is typically the precursor to a penetration test, enabling the penetration tester to gain a birds-eye view into what an entity really looks like at a technical level, what the entry points may be for the penetration test, dependencies to other entities

(ISPs and Hosting providers, for example), and also potential early indicators of points of weakness. Additionally, many large organizations struggle with managing their network perimeter and having an outside-in view of what an entity looks like can help gain and maintain visibility.

SpiderFoot

Now that you understand what footprinting is, how it's done and why, it's more meaningful when we say that SpiderFoot is a footprinting tool designed to automate the footprinting process to the fullest extent possible by extracting information from whatever data can be obtained freely from the Internet.

Background

When SpiderFoot v0.1b was originally released in 2005, it used the then-available Google API, Netcraft and website spidering as methods for building up a footprint, and these methods were hard-coded into the tool. Despite Google dropping support for its API and Netcraft blocking access to much of its data, SpiderFoot continued to be downloaded and used – clearly a need still existed for automated footprinting.

Modules

In version 2.0, which is completely modular and entirely re-written in Python, each method for building up the footprint is encapsulated in its own module. In addition, modules generate each data element identified (i.e. an IP address, a web page, etc.) as an “event” that is consumed by other modules listening for that event. This model enables SpiderFoot to extract “maximum value” out of each piece of data found. SpiderFoot’s modules, at the time of writing, are as follows:

- `sfp_dns`: Performs a number of DNS checks to obtain IP Addresses and Affiliates.
- `sfp_geoip`: Identifies the physical location of IP addresses identified.
- `sfp_googlesearch`: Some light Google scraping to identify links for spidering.
- `sfp_mail`: Identify e-mail addresses in any obtained web content.
- `sfp_pageinfo`: Information about web pages (do they take passwords, do they contain forms, etc.)
- `sfp_portscan_basic`: Scans for commonly open TCP ports on IP addresses found.
- `sfp_ripe`: Queries RIPE to identify owned netblocks and other info.
- `sfp_similar`: Searches various sources to identify similar looking domain names.

- `sfp_spider`: Spidering of web-pages to extract content for searching. Probably the most valuable module.
- `sfp_stor_db`: Stores scan results into the back-end SpiderFoot database. This is modularized for future scalability purposes. For now it stores results to an internal SQLite database.
- `sfp_subdomain`: Identify hostnames / sub-domain names in URLs and obtained content.
- `sfp_websvr`: Obtain web server banners to identify versions of web servers and related technology being used.
- `sfp_xref`: Identify whether other domains are associates (“Affiliates”) of the target.

Going into the inner workings of each module is beyond the scope of this article, but you can find the source code to each of them and more at the GitHub link provided below.

Installing

On Linux, *BSD or Solaris, installing and running SpiderFoot should be a breeze. Provided you have Python 2.6 or 2.7 (Python 3.x support coming soon), all you’ll need are CherryPy and Mako, two modules SpiderFoot uses for its web-based interface.

I am using FreeBSD 9.1-RELEASE as an example here, but if you’re using another BSD, you’ll probably need to adapt your approach slightly. If you’re using Linux, follow the instructions in the `README` file included in the SpiderFoot package.

Step 1

Install pip if you don’t have it already. This will enable you to easily install Python packages.

```
# cd /usr/ports/devel/py-pip
# make && make install
```

Step 2

Install SQLite for Python.

```
# cd /usr/ports/databases/py-sqlite3
# make && make install
```

Step 3

Install CherryPy and Mako Python modules.

```
# pip install cherryypy
# pip install mako
```

Step 4

Unpack SpiderFoot into a location of your choice.

```
~$ tar xzf spiderfoot-2.x.x-src.tar.gz
~$ cd spiderfoot
```

Starting

To run SpiderFoot, simply execute `sf.py` from the directory you extracted SpiderFoot into:

```
$ python ./sf.py
```

Once executed, a web-server will be started, which by default will listen on 127.0.0.1:5001. You can then use the web-browser of your choice by browsing to `http://127.0.0.1:5001`. You should then see something like this: Figure 1.

Configuring

With the exception of the IP and Port bound to by the SpiderFoot web server, which are set on the command-line, all other SpiderFoot configuration settings are controlled in the UI. After clicking on the *Settings* button in the title bar, you will be presented with a few global settings followed by module-specific settings (Figure 2).

Here you can configure things like the User-Agent string to use during spidering, the period of time to pause between web requests, TCP ports to scan, and more. Save settings keep them persistent between scans even if you stop and start SpiderFoot completely.

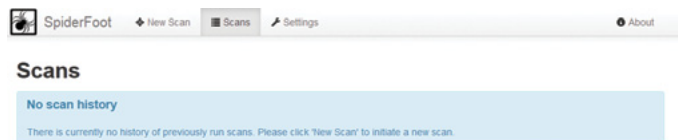


Figure 1. The SpiderFoot interface after starting it up for the first time

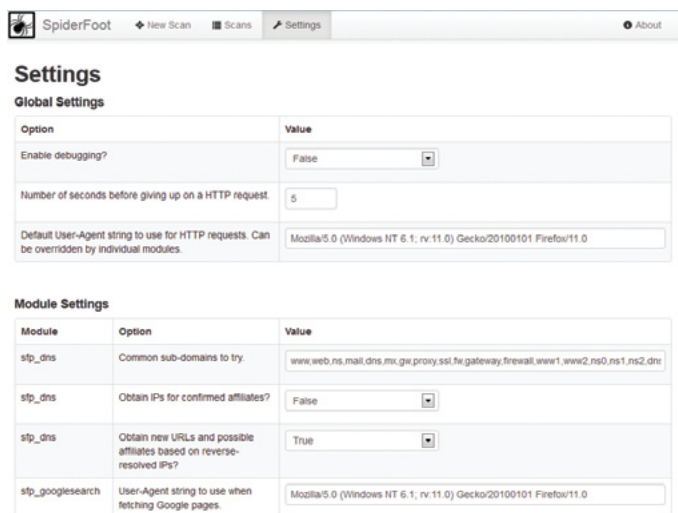


Figure 2. User interface for setting SpiderFoot's configuration

Running Scans

Running a scan is extremely simple – click the *New Scan* button in the title bar, then give the scan a descriptive name, specify the target you want to scan, and then select which modules you would like enabled or disabled: Figure 3.

Browsing Results

Thanks to the introduction of an SQLite database back-end in 2.0, scan results are stored – in real time as the scan progresses – locally in a database file. By clicking on the *Scans* button in the title bar, you can see a list of scans run previously, in addition to the scan you have just initiated. Click the name of the scan you are interested in and you will be presented with the data available for that

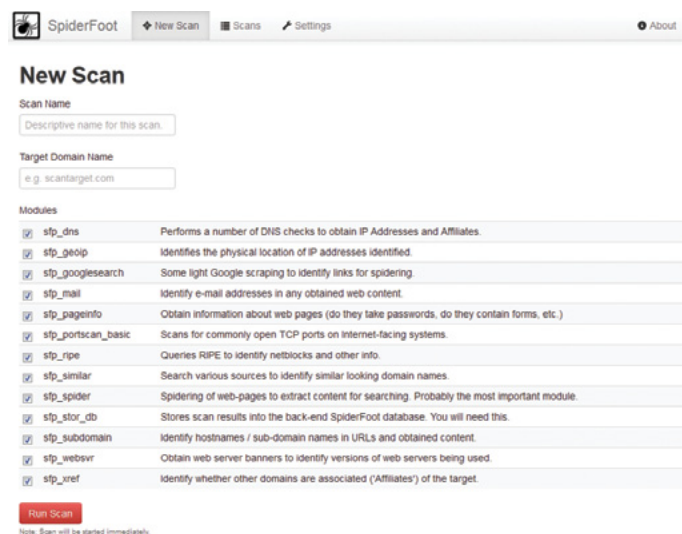


Figure 3. The SpiderFoot interface for initiating new scans

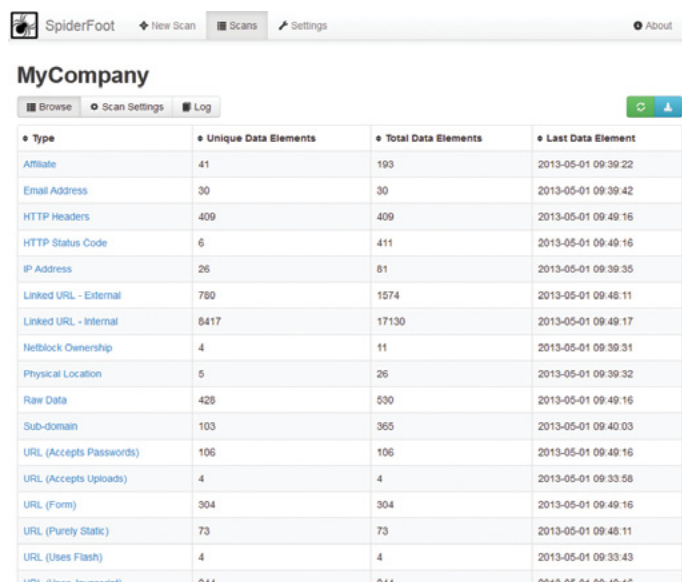
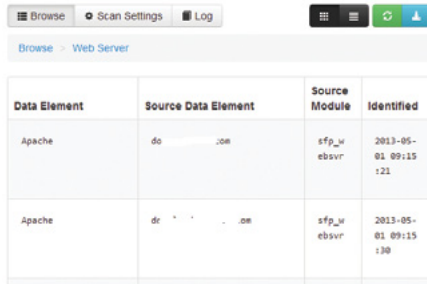


Figure 4. A list of data elements making up the footprint of a target

On the Web

- <http://www.spiderfoot.net> – The SpiderFoot website.
- <http://github.com/smicallef/spiderfoot> – SpiderFoot source code on GitHub.
- <http://twitter.com/binarypool> – SpiderFoot twitter feed.

MyCompany



Data Element	Source Data Element	Source Module	Identified
Apache	doom	sfp_w ebsvr	2013-05- 01 09:15 :21
Apache	drom	sfp_w ebsvr	2013-05- 01 09:15 :30

Figure 5. A detailed listing of the data elements (in this case, Web Servers) from a footprint

scan. This starts getting populated the moment a scan initiates; see Figure 4. From here you are then able to “drill down” into the actual data. Data can also be exported to CSV format for offline manipulation/analysis if desired by clicking the blue icon to the right (Figure 5).

Looking Ahead

Hopefully this article has given some insight into the interesting world of footprinting with SpiderFoot. The tool is still very much in its infancy, but it does the job it is tasked to do with big plans for new modules and additional core functionality. Plans for future modules include SSL certificate checks, identifying the entity's ISPs (possibly using Traceroute or BGP tables), and 3rd party integration with vulnerability scanners and the like, but you can get a full list on the GitHub project site with the link provided below.

Happy Footprinting!

STEVE MICALLEF

Steve Micallef has been specializing in IT Security for the past 13 years, currently working in a large financial institution. With a passion for security and for delivering quality security solutions, Steve has designed, built and delivered global solutions in the areas of SIEM (Security Information & Event Management), Vulnerability Scanning, Data Leakage Prevention and more.

Steve created SpiderFoot with the goal of giving Penetration Testers a way to automate the more cumbersome and time-consuming process of a penetration test – footprinting. He is constantly looking at ways to improve the tool, always with that goal in mind.

**BSD development
and consultancy**

Zabbix Monitoring

**Bacula enterprise
backup**

BSD Thin Client

**Corporate BSD
Desktop**

**Solution
management
with Puppet**

and more ...

www.mtier.org

contact@mtier.org

FreeBSD Jails Firewall with PF

Features are available for fully virtualizing FreeBSD jail networking (as of FreeBSD 8.x). The code has improved in the current 9.x code base but to get a jail up and running with the current install, PF provides the necessary functionality to firewall off multiple jailed services.

What you will learn...

- Configuration of PF to setup nat and rdr rules for ssh access
- Basic setup of jails using ezjail and the jls and jexec utils

What you should know...

- Basic FreeBSD knowledge to navigate the command line
- Familiarity with PF and navigating the ports system

This article will cover basic jails configuration to highlight how to configure the firewall to only allow specific traffic to the service jails. The first thing that needs to be completed is an install of FreeBSD 9.1 (amd64) with an install of the system source and the ports tree (See [FREEBSD-INSTALL](#) for installation instructions). To help with the jail configuration, I am using `ezjail`. Listing 1 shows how to install the `ezjail` port and how to configure a basic jail called “ssh-test”.

The key thing about this configuration is that I am using an IP on the local interface “127.0.1.1”. `ezjail-admin` will output that the interface has not been configured when creating the jail. Listing 2 demonstrates the configuration to get the jail up and running on the local interface with an alias on `lo0`.

Once the system has rebooted, the new jail will be up and running with the local alias IP “127.0.1.1”. Listing 3 shows the output of the `jls` command and the alias on the local interface.

Listing 1. *Install ezjail and setup ssh-test jail. (Note: for the jail creation, em0 is the interface type for a VirtualBox VM. This may be different in your setup so use the appropriate interface)*

```
# cd /usr/ports/sysutils/ezjail/
# make -DBATCH install clean
..
(Output from install ezjail port)
# echo `ezjail_enable="YES" >> /etc/rc.conf
# ezjail-admin install
..
(This will take some time, as it creates a base jail)
# ezjail-admin create ssh-test `em0|127.0.1.1`
```

Listing 2. *Configuring the interface to load up with the jail IP*

```
# echo `ifconfig_lo0_alias0="inet 127.0.1.1 netmask 0xffffffff" >> /etc/rc.conf
# reboot
```

For now, we will setup ssh to automatically start inside the jail. In addition, we will create a “test-user” to be able to login over ssh. Listing 4 shows the commands to change the default ssh port to 2022, add the test-user and enable sshd on startup.

At this point, when running from the host operating system, you should be able to ssh on port 2022 into the jail. However, if you wanted to connect in from a remote system, the local interface connection would not be available. This is where `pf` can be configured to redirect traffic into

the jail. Listing 5 shows a basic `pf` configuration to provide NAT redirection for the jail.

The firewall rules essentially take all tcp port 2022 traffic and redirect it to the jailed `sshd` service. Any traffic sent back will be NATed on the host interface (`em0` in this example). The firewall needs to be configured at startup, which is demonstrated in Listing 6.

The system will reboot and from another remote system (or the host OS) you should be able to ssh on port 2022 into the jail. Check the above configuration settings if this

Listing 3. Output of `jls` showing new interface alias and the `ssh-test` jail up and running. (Note: the jail ID in this configuration is 1, which is used with the `jexec` command to run a shell inside the jail)

```
# jls
  JID  IP Address      Hostname          Path
  ---  -
   1  127.0.1.1      ssh-test         /usr/jails/ssh-test

# ifconfig lo0
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> metric 0 mtu 16384
    options=600003<RXCSUM,TXCSUM,RXCSUM_IPV6,TXCSUM_IPV6>
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x5
    inet 127.0.0.1 netmask 0xff000000
    inet 127.0.1.1 netmask 0xffffffff
    nd6 options=21<PERFORMNUD,AUTO_LINKLOCAL>

# jexec 1 tcsh
root@ssh-test:/ # ls
.cshrc  COPYRIGHT  bin        dev        lib        media      proc       root       sys        usr
.profile basejail   boot       etc        libexec    mnt       rescue    sbin      tmp        var
root@ssh-test:/ #
```

Listing 4. Changing the default port for `sshd` and enable it on startup for the jail.

```
root@ssh-test:/ # sed -i 's/^.*Port 22.*$/Port 2022/g' /etc/ssh/sshd_config
root@ssh-test:/ # echo 'sshd_enable="YES"' >> /etc/rc.conf
root@ssh-test:/ # pw user add -n test-user -s /bin/csh -m
root@ssh-test:/ # passwd test-user
Changing local password for test-user
New Password:
Retype New Password:
root@ssh-test:/ # /etc/rc.d/sshd start
..
(Output from the SSH key generation)
root@ssh-test:/ # sockstat -4
USER      COMMAND  PID  FD  PROTO  LOCAL ADDRESS    FOREIGN ADDRESS
root     sshd     1301  3   tcp4   127.0.1.1:2022   *.*
root     sendmail 1141  3   tcp4   127.0.1.1:25     *.*
root     syslogd  1087  6   udp4   127.0.1.1:514    *.*
root@ssh-test:/ # exit
#
(You should be out of the jail for the next steps.)
```

Listing 5. */etc/pf.conf configuration for redirecting remote traffic to jailed services. (Note: adjust the \$ext_if according to the interface type you are using.)*

```
ext_if="em0"
SSHTEST="127.0.1.1"

# nat all jail traffic
nat pass on $ext_if from $SSHTEST to any -> ($ext_if)

# port 2022 is redirected to the jail
rdr pass on $ext_if proto tcp from any to any port 2022
-> $SSHTEST port 2022

# port 22 on host
pass in log on $ext_if inet proto tcp from any to port
22 flags S/SA
pass out log on $ext_if proto tcp all keep state flags S/
SA
```

Listing 6. */etc/rc.conf configuration for enabling pf on startup.*

```
# echo `pf_enable="YES" >> /etc/rc.conf
# echo `pf_rules="/etc/pf.conf" >> /etc/rc.conf
# echo `pf_program="/sbin/pfctl" >> /etc/rc.conf
# echo `pf_flags="" >> /etc/rc.conf
# reboot
```

Listing 7. *Running ssh to remotely connect into the jail.*

```
$ ssh -p2022 test-user@192.168.58.120
Password:
Last login: Mon May 06 17:24:04 2013 from 192.168.58.1
FreeBSD 9.1-RELEASE (GENERIC) #0 r243825: Tue Dec 4
09:23:10 UTC 2012

Welcome to FreeBSD!

Before seeking technical support, please use the
```

is not working. Listing 7 shows the output of remotely logging into the jail with ssh. This is only a basic configuration for providing services within a jail. If you include the ports system for the jail, additional software can be added to provide web services and any other basic services. Using

References

- FREEBSD-INSTALL: <http://www.freebsd.org/doc/handbook/bsdinstall.html>
- Jails: <http://www.freebsd.org/doc/handbook/jails.html>
- ezjail: <http://erdgeist.org/arts/software/ezjail/>

following resources:

- o Security advisories **and** updated errata information **for** all releases are at <http://www.FreeBSD.org/releases/> - always consult the ERRATA section **for** your release first **as** it's updated frequently.
- o The Handbook **and** FAQ documents are at <http://www.FreeBSD.org/> **and**, along **with** the mailing lists, can be searched by going to <http://www.FreeBSD.org/search/>. If the doc package has been installed (or fetched via `pkg_add -r lang-freebsd-doc`, where lang **is** the 2-letter language code, e.g. en), they are also available formatted **in** `/usr/local/share/doc/freebsd`.

If you still have a question **or** problem, please take the output of `uname -a`, along with any relevant error messages, and email it **as** a question to the questions@FreeBSD.org mailing list. If you are unfamiliar **with** FreeBSD's directory layout, please refer to the `hier(7)` manual page. If you are **not** familiar **with** manual pages, type `man man`.

Edit `/etc/motd` to change this login announcement.

```
test-user@ssh-test:/home/test-user %
```

pf, all services can be provided to external connections while at the same time authorizing only the necessary ports for jail access giving additional security for services.

MICHAEL SHIRK

Michael Shirk is a BSD zealot who has worked with OpenBSD and FreeBSD for over 7 years. He works in the security community and supports Open Source security products that run on BSD operating systems. Michael is the Chief Executive Manager of Daemon Security Inc., a company which provides security solutions utilizing the BSD operating systems: <http://www.daemon-security.com>

BSDCAN 2013

THE BEST EVENT OF 2013
<http://www.bsdcn.org/>

Ottawa, Canada



BSDCan 2013 – The event to be at this year

BSDCAN 2013

WHERE

15-16 May – tutorials
17-18 May – conference

WHO

All who are working on and with 4.4BSD based operating systems and related projects.

VENUE

University of Ottawa
<http://www.uottawa.ca/>

High value. Low cost. Something for everyone.

AT FEES YOU CAN AFFORD

We plan to keep to a minimum. As such, the conference will be held at University of Ottawa and accommodation is available within the University residences. Hotels are also within close walking distance of the conference venue.

WHAT DOES IT COST?

Type	CAD
Individual	\$195
Corporate	\$350
Additional Corporate	\$175
Student	\$60
Tutorial (per half day)	\$60
University of Ottawa Staff&Student	\$45

Take the BSDA Certification exam.

For details see

<http://bsdcertification.org/>

- A NetBSD based Tracking Radar
- FreeBSD Kernel Security
- Automating the deployment of FreeBSD & PC-BSD
- Backup and Restore with Bacula
- Benchmarking FreeBSD
- Switching from Linux to FreeBSD
- DNSSec: Troubleshooting and Deployment
- Embedding NetBSD: VOIP applications
- FreeBSD, Capsicum, GELI and ZFS
- FreeBSD Doc Sprint
- FreeBSD storage options
- Hands-on bhyve, the BSD Hypervisor
- Introduction to pkgsrc
- MCLinker BSD
- Making FreeBSD Ports
- Managing FreeBSD at scale
- Modern package management
- Mozilla on OpenBSD
- OpenIKED
- Runtime Process Infection
- The BSD ISP
- Buffer Cache in OpenBSD
- Wireless networking - mobile, gigabit and beyond
- Complexity of checksums in TCP/IP

Improvements to Jail Management via the Warden[®]

Over the past few months, several exciting new features have been added to the Warden which greatly improve jail management on FreeBSD & PC-BSD systems.

PC-BSD

Historically the Warden has always organized its collections of jails via a primary IP address. This was functional but not the optimal point of reference when dealing with large quantities of jails on a system. Thanks to some recent cooperation between the PC-BSD & FreeNAS teams, this has been done away with and improved.

Now the Warden will be able to create jails via Hostname / Nickname, and change and assign IP addresses on the fly. This greatly simplifies jail creation via the command-line, allowing you to create the jail and then set addresses as needed later.

```
# warden create myjail
# warden set ipv4 myjail 192.168.0.25/24
# warden set ipv6 myjail fe80::8a89:a5ff:fe52:ad19
```

In addition to being able to set both a primary Ipv4 and Ipv6 address, jails can also include a number of other addresses. Any number of aliases for both Ipv4 and Ipv6 can be set, along with the default router for Ipv4 & Ipv6. The Warden is also now configured to automatically use the VNET option, giving each jail its own virtual network stack. This includes giving jails their own network interface and can allow a wider variety of services

to run behind a jailed interface. Because of this feature, the Warden will require that your kernel is compiled with the `VIMAGE` option enabled. Users of PC-BSD & TrueOS rolling-release will be able to update to this kernel via the normal `freebsd-update` mechanisms. With these new features brings new options which can be set via the command-line:

```
# warden set myjail alias-ipv4 192.168.0.200/24
# warden set myjail bridge-ipv4 192.168.0.2/24
# warden set myjail alias-bridge-ipv4 192.168.0.3/24
```

Along with new virtual networking functionality, the Warden also has a few new tricks up its sleeve. For PC-BSD & TrueOS 9.1 and higher users, we have begun building and maintaining our own full package repository using `pkgng`. When creating standard jails, the Warden will handle automatically boot-strapping the `pkgng` package and repository.

Should this process be unable to complete, such as on a system with no internet connectivity, or be corrupted by a well-meaning end user, it can be re-run at any time:

```
# warden bspkgng myjail
```


Another long-requested feature was the ability for the Warden to manage setting various permissions and flags for a jail and handle user-supplied nullfs mounts. These can both be easily configured per-jail by using the “set flags” and “fstab” options respectively.

```
# warden set myflags myjail allow.raw_sockets=true
# warden fstab myjail
```

All of these new features and options are also fully exportable. This will allow you the ability to provision a jail on your PC-BSD workstation, either via the command-

line or GUI. Once you have finished the initial configuration and testing of your jail, you can then easily export it to a single archive file. This export file can then be taken to another system, such as FreeNAS, and then imported.

```
# warden export myjail -dir=/exports
# warden import /exports/myjail.wdn
```

At the time of this writing many of these changes are also being implemented into the Warden’s Graphical Interface. As easy as the command-line flags may be, the GUI takes it a step further, making jail creation and management possible without having to remember or look up a single command.

So what is next for the Warden? Even with these new features still hot off the press, there are other improvements waiting in the wings. One of these will be the ability to create and manage various jail “templates”. This will allow you to build a jail template for a particular FreeBSD release (say you have a product which needs to run on 8.3). By creating the 8.3 template, you will be able to customize it with software or configuration options specific to your needs. Then when it comes time to build jails, you will be given the option of using the latest release or your own jail template. Stay tuned to BSD Magazine for more details on this in a future issue.

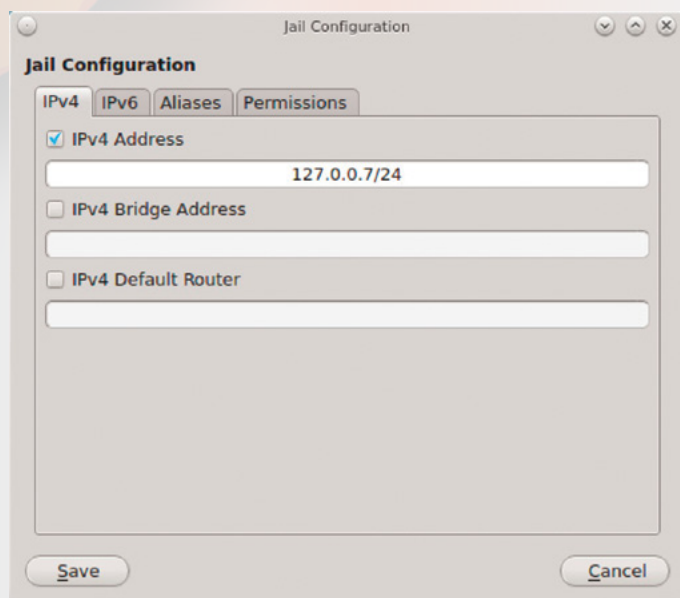


Figure 1. The jails IPv4 configuration

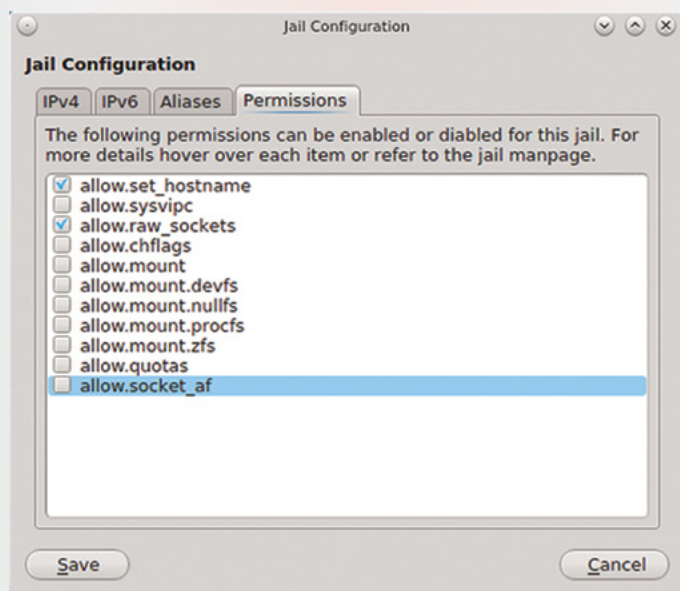


Figure 2. Setting jail permissions

KRIS MOORE

Kris Moore is the founder and lead developer of PC-BSD. He lives with his wife and four children in East Tennessee, USA and enjoys building custom PC's and gaming in his (limited) spare time. He can be reached at: kris@pcbsd.org.



msearch: MidnightBSD Search

A few years ago, I was trying to find a file on my MidnightBSD desktop system. I couldn't remember the name, but knew there was a specific phrase in it. I could use the `grep` command to find the file, but it would take time.

What you will learn...

- the history of the `msearch` tool and why it was written,
- basic usage of the `msearch` tool

What you should know...

- how to install MidnightBSD or download a virtual machine image

I thought about how quickly Apple's Spotlight works in Mac OS X. I also considered how terrible most open source full text search engines operate. I decided to write my own search tool to make searching for files easier.

Using MidnightBSD Search

MidnightBSD search, or `msearch` is a full text search tool. It offers the user the ability to search against filenames or contents of text files.

`msearch` is not meant to replace other tools like `find`, `locate` or `whereis`.

Table 1. `msearch` option flags

Command	Description
<code>-c</code>	Print the match count only.
<code>-l <number></code>	Limit the number of results
<code>-r</code>	Print the ranking information with full text results
<code>-t</code>	Perform a full text search rather than just using filenames
<code>-z</code>	Print pathnames separated by an ASCII NUL character rather than a newline.

How Does MidnightBSD Search Work?

Files on the system are indexed weekly from a periodic script that runs an indexing program. The indexes are used by the command line tool when executing searches.

Indexing in action

`msearch.index` indexes files on the system by determining if the file is a text file using `libmagic`, reading the first 20KB of the file and loading it into the full text indexer. The results are stored in SQLite databases; they are stored in `/var/db/msearch`.

```
Listing 1. Example search queries
# Filename based search, limited to 10 results.
msearch -l 10 msearch

/usr/bin/msearch
/usr/include/msearch.h
/usr/lib/libmsearch.a
/usr/lib/libmsearch_p.a
/usr/lib/libmsearch.so.1
/usr/lib/libmsearch.so
/usr/libexec/msearch.index
# Text based search
msearch -t "Lucas Holt"
/usr/local/mailman/archives/public/midnightbsd-
users/2007-August.txt
/usr/local/mailman/archives/public/midnightbsd-
users/2011-February.txt
/usr/local/mailman/archives/public/midnightbsd-
kernel/2008-September.txt
```

On the Web

- <http://www.midnightbsd.org/> – MidnightBSD Project website,
- <http://www.midnightbsd.org/cgi-bin/cvsweb.cgi/src/lib/libmsearch/> – msearch library.

Glossary

- msearch
- sqlite

The msearch.db file contains a list of filenames, ownership information, sizes, and other general metadata. *msearch_full.db* contains the full text search data.

Turn on msearch indexing

Indexing is enabled by adding `weekly_msearch_enable="YES"` to `/etc/periodic.conf`. If you have many files, it is recommended to have at least a few gigabytes of free space on the `/var` mount point.

Once the index has been generated for the first time, you will be able to use msearch to find files.

Extending MidnightBSD Search

msearch is built on top of a shared library, libmsearch, that allows developers to integrate search functionality into their own applications. Functions for creating and manipulating indexes, as well as performing searches are included.

Consult the msearch.h header file for a complete list of functions.

Future Directions

Following the 0.4-RELEASE of MidnightBSD, I plan to write a graphical application to extend searching and a new indexer. Scalability is a concern with regard to index storage size. Creating an indexing daemon would allow the index to maintain fresh. This would require use of kqueue or porting inotify from Linux.

Summary

msearch is an easy to use full text search tool for MidnightBSD. It allows users to quickly search text files on their system.

LUCAS HOLT

Lucas Holt is the founder of the MidnightBSD project and a Senior Application Programmer/Analyst for the University of Michigan in Ann Arbor, MI, USA.

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

Useful Utilities for PF

This article explores some of the third-party utilities which are available to help you analyze the log and state table of a PF firewall.

What you will learn...

- How to view the PF state table in real time
- How to convert the PF log to HTML format or graph format

What you should know...

- How to restart PF
- How to install third-party software on your BSD system

The PF firewall is developed by the OpenBSD Project. PF has also been ported to FreeBSD, NetBSD, and DragonFly BSD. You can learn more about PF and its features in the PF User's Guide at <http://www.openbsd.org/faq/pf/>.

PF is a stateful firewall, meaning that it tracks the state of existing connections in a state table, allowing the firewall to quickly determine if packets are part of an established connection. PF also provides a logging facility and the firewall administrator controls which packets get logged by including the *log* keyword in only the firewall rules which should be logged when matched.

PF provides the `pfctl` utility for displaying the state table and the built-in `tcpdump` utility can be used to view the PF log. In addition to these tools, some third-party packages can be installed on BSD systems. These can be used to manipulate information from the state table and the PF logging facility in order to get a different view on what is happening with the firewall. This article provides

an overview of the following utilities: `pftop`, `pflogx`, and `pfstat`. These utilities were tested on a PC-BSD system and the utilities were installed using FreeBSD packages. This article assumes that you already know how to restart PF and how to install software on your BSD system using packages, ports, or pkgsrc.

pftop

- Website: <http://www.eee.metu.edu.tr/~canacar/pftop/>
- Availability: pkgsrc, FreeBSD and OpenBSD packages
- Description: provides real time display of PF state table and rule statistics

This utility is similar to `top` as it provides a real time, columnar display. However, instead of displaying the top processes running on the system, it displays real time information about the current connections in the PF state table.

Listing 1. `pfctl` View of State Table

```
pfctl -s states
all tcp 192.168.1.71:19348 → 204.152.184.134:21 ESTABLISHED:ESTABLISHED
all tcp 192.168.1.71:34852 → 204.152.184.134:42342 ESTABLISHED:ESTABLISHED
all udp 192.168.1.71:5353 → 224.0.0.251:5353 SINGLE:NOTRAFFIC
(snip rest of output...)
```



MidnightBSD

the BSD for everyone

Come join MidnightBSD:

- desktop BSD •
- over 2,000 ports •
- unique mports package installer •
- helpful community •
- hands-on learning for new developers •

www.midnightbsd.org

HOW TO

```
pfTop: Up State 1-12/12, View: default, Order: none, Cache: 10000 11:26:46
PR      DIR SRC          DEST          STATE        AGE         EXP         PKTS  BYTES
tcp     Out 192.168.1.71:19348 204.152.184.134:21 10:10 00:01:49 00:01:06 72 4621
tcp     Out 192.168.1.71:34852 204.152.184.134:42342 10:10 00:01:47 00:01:06 113728 66980K
udp     Out 192.168.1.71:5353 224.0.0.251:5353 1:0 00:00:18 00:00:42 1 64
udp     In 192.168.1.71:5353 224.0.0.251:5353 0:1 00:00:18 00:00:42 1 64
udp     Out 192.168.1.71:56063 192.168.1.254:53 2:1 00:00:16 00:00:14 2 138
udp     Out 192.168.1.71:10264 192.168.1.254:53 2:1 00:00:16 00:00:14 2 150
tcp     Out 192.168.1.71:56828 204.152.184.134:21 9:9 00:00:16 00:01:15 36 2313
udp     Out 192.168.1.71:43450 192.168.1.254:53 2:1 00:00:15 00:00:15 2 138
udp     Out 192.168.1.71:22187 192.168.1.254:53 2:1 00:00:15 00:00:15 2 150
tcp     Out 192.168.1.71:61679 204.152.184.134:21 10:10 00:00:15 00:01:17 60 3732
tcp     Out 192.168.1.71:64173 204.152.184.134:21 4:4 00:00:13 23:59:48 54 3686
tcp     Out 192.168.1.71:38365 204.152.184.134:57176 4:4 00:00:12 24:00:00 15565 9120150
```

Figure 1. Default *pf*top Display

```
[root@pcbsd-4345] ~# pfctl -s rules
No ALTQ support in kernel
ALTQ related functions disabled
scrub in all fragment reassemble
block drop in quick on ! lo0 inet from 127.0.0.0/8 to any
block return in from no-route to any
block return in log all
pass out all flags S/SA keep state
block return from <blacklist> to any
pass log proto icmp all keep state
pass proto ipv6-icmp all keep state
pass in quick on em0 proto udp from any to (em0) port = netbios-ns keep state
pass in quick on em0 proto udp from any to (em0) port = netbios-dgm keep state
pass in quick on em0 proto udp from any to (em0) port = sunrpc keep state
pass in quick on em0 proto udp from any to (em0) port = nfsd-keepalive keep state
pass in quick on em0 proto udp from any to (em0) port = nfsd keep state
pass in quick on em0 proto udp from any to (em0) port = lockd keep state
pass in quick on em0 proto udp from any to (em0) port = mdns keep state
pass in quick on em0 proto udp from any to any port 49152:65535 keep state
pass in quick on em0 inet proto udp from any to 224.0.0.251 port = mdns keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = netbios-ns keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = netbios-dgm keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = sunrpc keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = nfsd-keepalive keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = nfsd keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = lockd keep state
pass in quick on iwn0 proto udp from any to (iwn0) port = mdns keep state
pass in quick on iwn0 proto udp from any to any port 49152:65535 keep state
pass in quick on iwn0 inet proto udp from any to 224.0.0.251 port = mdns keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = netbios-ns keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = netbios-dgm keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = sunrpc keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = nfsd-keepalive keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = nfsd keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = lockd keep state
pass in quick on lagg0 proto udp from any to (lagg0) port = mdns keep state
pass in quick on lagg0 proto udp from any to any port 49152:65535 keep state
pass in quick on lagg0 inet proto udp from any to 224.0.0.251 port = mdns keep state
```

Figure 2. Viewing Loaded Rules Using *pfctl*

Typically, the state table is read using `pfctl` as seen in the following example. This output is from a PC-BSD system that is downloading a PBI using AppCafe.

Figure 1 shows the same state table. This time, the display is generated by typing `pftop`.

In order, the columns in this default view list the protocol (TCP or UDP), the direction (into the system or out of the system), the source address and socket, the destination address and port, the state of the connection, the age of the connection, how long until that connection expires from the state table, the number of packets in that connection, and the number of bytes transferred.

`pfptop` also provides a view for displaying which rules are currently loaded. First, Figure 2 shows which firewall rules have been loaded using the built-in `pfctl`.

Next, Figure 3 shows the same rules, this time viewed using `pfptop`. This display adds information such as the number of packets, bytes, and established connections (states) associated with each rule.

`pfptop` also provides an interactive mode where keystrokes can be used to modify the view, sort the column order, change the number of lines to display, and to pause or restart the display. Display filters can also be created using `tcpdump` syntax. Refer to `pfptop(8)` for details.

pflogx

- Website: <http://akldev.free.fr/pflogx/>
- Availability: FreeBSD and OpenBSD packages
- Description: generates an XML file from a PF log which can then be optionally transformed into HTML or csv format

PF writes its logs in a binary format, meaning that they cannot be read using `head`, `tail`, `more`, `less`, or an editor. While the logs can be read in real time using the command `tcpdump -n -e -ttt -i pflog0`, it is sometimes convenient to convert the logging information to another format in order to study it and analyze trends. `pflogx` renders the PF log in XML format and includes the ability to transform the XML into HTML or csv format. Optionally, the generated XML file can be passed to other third-party tools for conversion to other formats.

In order to use `pflogx`, the PF logging module must be loaded and at least one rule in the PF rulebase must include the log keyword. You can double-check that log entries exist by typing `pflogx -i /var/log/pflog`. As seen in this example, this command displays the log entries to the screen: Listing 2.

```
[root@pcbsd-4345] ~# pftop -v rules
pfTop: Up Rule 1-35/58, View: rules, Cache: 10000
```

RULE	ACTION	DIR	LOG	Q	IF	PR	K	PKTS	BYTES	STATES	MAX	INFO
0	Block	In		Q	!lo0			0	0	0		drop inet from 127.0.0.0/8 to any
1	Block	In						0	0	0		return from no-route to any
2	Block	In	Log					45	8222	0		return all
3	Pass	Out					K	16404	13972610	73		all flags S/SA
4	Block	Any						0	0	0		return from <blacklist> to any
5	Pass	Any	Log			icmp	K	8	672	1		all
6	Pass	Any				ipv6-icmp	K	0	0	0		all
7	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = netbios-ns
8	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = netbios-dgm
9	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = sunrpc
10	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = nfsd-keepalive
11	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = nfsd
12	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = lockd
13	Pass	In		Q	em0	udp	K	0	0	0		from any to (em0) port = mdns
14	Pass	In		Q	em0	udp	K	4	305	2		from any to any port 49152:65535
15	Pass	In		Q	em0	udp	K	2	128	2		inet from any to 224.0.0.251/32 port = mdns
16	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = netbios-ns
17	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = netbios-dgm
18	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = sunrpc
19	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = nfsd-keepalive
20	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = nfsd
21	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = lockd
22	Pass	In		Q	iwn0	udp	K	0	0	0		from any to (iwn0) port = mdns
23	Pass	In		Q	iwn0	udp	K	0	0	0		from any to any port 49152:65535
24	Pass	In		Q	iwn0	udp	K	0	0	0		inet from any to 224.0.0.251/32 port = mdns
25	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = netbios-ns
26	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = netbios-dgm
27	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = sunrpc
28	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = nfsd-keepalive
29	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = nfsd
30	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = lockd
31	Pass	In		Q	lagg0	udp	K	0	0	0		from any to (lagg0) port = mdns
32	Pass	In		Q	lagg0	udp	K	0	0	0		from any to any port 49152:65535
33	Pass	In		Q	lagg0	udp	K	0	0	0		inet from any to 224.0.0.251/32 port = mdns

Figure 3. Viewing Loaded Rules Using `pfptop`

HOW TO

To instead save the log to an XML file, after the input (-i /var/log/pflog), specify the name of the output file (-o filename.xml).

Optional filters can be placed between the input and output. They can be defined by action (-a pass or -a drop), direction (-d in, -d out, or -d in-out), protocol (-p icmp, -p ip, -p tcp, or -p udp), and interface (-n interface_name).

If a filter is not included, all packets in the input log file will be generated to the output XML file. Several filter examples can be found in the *README* file that is installed with pflogx.

The package installs several XSLT files which are used to transform the XML file to HTML, XHTML, or csv format. To transform a generated XML file, copy it to the directory

Listing 2. Sample XML File

```
pflogx -i /var/log/pflog
<?xml version="1.0" encoding="UTF-8"?>
<pflogx version="0.86" >
<logs>
<log date="2013-04-23 12:43:48.261661" if="em0" action="drop" rule="2" direction="in" protocol="udp" src_
  adr="205.233.73.201" src_port="123" dest_adr="192.168.1.71" dest_port="123" />
<log date="2013-04-23 12:44:24.41857" if="em0" action="drop" rule="2" direction="in" protocol="(2)" src_
  adr="192.168.1.254" src_port="" dest_adr="224.0.0.1" dest_port="" />
<log date="2013-04-23 12:46:29.44070" if="em0" action="drop" rule="2" direction="in" protocol="(2)" src_
  adr="192.168.1.254" src_port="" dest_adr="224.0.0.1" dest_port="" />
<log date="2013-04-23 12:47:50.298105" if="em0" action="drop" rule="2" direction="in" protocol="udp" src_
  adr="192.168.1.71" src_port="138" dest_adr="192.168.1.255" dest_port="138" />
<log date="2013-04-23 12:47:50.298145" if="em0" action="drop" rule="2" direction="in" protocol="udp" src_
  adr="192.168.1.71" src_port="138" dest_adr="192.168.1.255" dest_port="138" />
(rest of output snipped....)
```

file:///usr/local/share/examples/pflogx/pflog.xml

Date	Interface	Action	Rule	Direction	Protocol	Src. address	Src. port	Dest. address	Dest. port
2013-04-23 12:43:48.261661	em0	drop	2	in	udp	205.233.73.201	123	192.168.1.71	123
2013-04-23 12:44:24.41857	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:46:29.44070	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:47:50.298105	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138
2013-04-23 12:47:50.298145	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138
2013-04-23 12:48:34.46791	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:48:52.827069	em0	drop	2	in	tcp	192.168.1.96	20261	192.168.1.71	22
2013-04-23 12:49:05.141466	em0	pass	5	in	icmp	192.168.1.96		192.168.1.71	
2013-04-23 12:49:16.92832	em0	drop	2	in	tcp	192.168.1.96	58203	192.168.1.71	23
2013-04-23 12:50:39.48931	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:52:44.50937	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:54:49.53355	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:56:54.55460	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:58:59.57862	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 12:59:51.18030	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138
2013-04-23 12:59:51.18055	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138
2013-04-23 13:01:04.60149	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:03:09.62345	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:05:14.64437	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:07:19.67027	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:09:24.69122	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:11:29.72991	em0	drop	2	in	(2)	192.168.1.254		224.0.0.1	
2013-04-23 13:11:51.738657	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138
2013-04-23 13:11:51.738819	em0	drop	2	in	udp	192.168.1.71	138	192.168.1.255	138

Figure 4. Sample PF log in HTML Format

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



REDSPIHERE™
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

HOW TO

containing these files. On a FreeBSD or PC-BSD system, these files are located in `/usr/local/share/examples/pflogx/`. In the generated XML file, the first line should be:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Insert a second line that contains the name of the XSLT file. For example, to transform to HTML, add this line:

```
<?xml-stylesheet type="text/xsl" href="export_html.xsl"?>
```

Save the edit and you should now be able to view the XML file in a web browser, as seen in the example in Figure 4.

Listing 3. Sample pfstat Configuration

```
# more /usr/local/etc/pfstat.conf
collect 1 = interface "em0" pass bytes in ipv4 diff
collect 2 = interface "em0" pass bytes out ipv4
diff
image "/usr/home/dru/bandwidth.jpg" {
  from 7 days to now
  width 1000 height 400
  left
  graph 1 bps "in" "bits/s" color 0 192 0
  filled
  right
  graph 2 bps "out" "bits/s" color 0 0 255
}
collect 3 = global states entries
image "/usr/home/dru/states.jpg" {
  from 12 months to now
  width 800 height 200
  left
  graph 3 "states" "entries" color 200 0 0
}
```

`pflogx` provides a merge option (`-m`) which can be used to append new log entries to an existing XML file, allowing you to visualize the transformed log over time.

pfstat

- Website: <http://www.benedrine.cx/pfstat.html>
- Availability: `pkgsrc`, FreeBSD and OpenBSD packages
- Description: automatically generates graphs from PF statistics

If you prefer to visualize the PF logs in a graph format, install `pfstat`. Once installed, create its log directory and log file if they do not exist:

```
# mkdir /var/log/pflog
# touch /var/log/pflog/pflog
```

Next, create a configuration file named `/usr/local/etc/pfstat.conf`. This file controls which graphs get created. A comprehensive file with comments on the various graphs it creates can be downloaded from <http://www.benedrine.cx/pfstat.conf>. The following example shows a simpler configuration file which creates two graphs: one displays bandwidth in bits per second and the other charts the state table. Edit the text in red to point to an existing directory path. The filename (e.g. `bandwidth.jpg`) should not already exist in the specified directory as `pfstat` will generate it for you.

Next, type `crontab -e` as the superuser to edit the root user's crontab. Add the following line:

```
* /5 * * * * /usr/local/bin/pfstat -q >> /var/log/pfstat
```

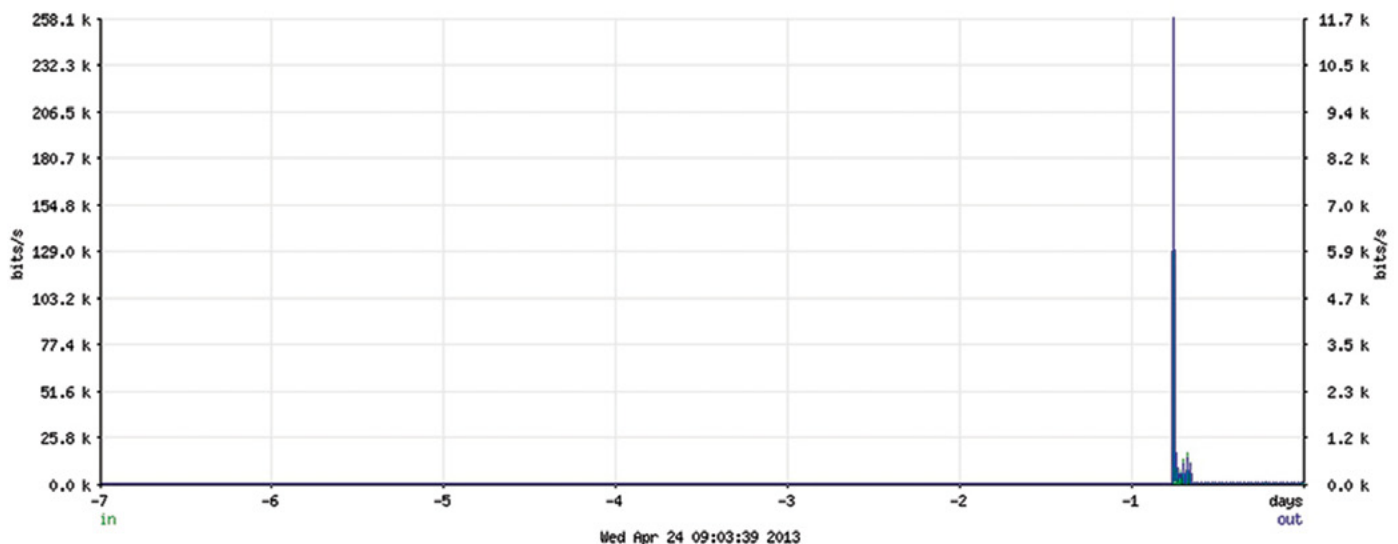


Figure 5. Sample Graph

This instructs `pfstat` to query the logging interface every five minutes and to store the received logging information in its own database, which it uses to generate graphs.

Finally, add this line to the beginning of `/etc/pf.conf` in order to set the logging interface. Replace `em0` with the name of the interface you wish to collect statistics on. Restart the PF firewall after saving this edit.

```
set loginterface em0
```

Wait a bit (at least five minutes) to allow `pfstat` to add logging information to its database. The amount of information added to the database will depend upon how often a logged rule matches the criteria you have configured `pfstat` to graph.

Whenever you want to generate a graph, type `pfstat -p`. This instructs `pfstat` to read the entries in its database and to generate the images to the locations that you specified in `/usr/local/etc/pfstat.conf`. Figure 5 shows a sample `/usr/home/dru/bandwidth.jpg` from the configuration file above, after running `pfstat` for one day on a home desktop system.

`pfstat(8)` provides some more information on how to use `pfstat`, remove old entries from the database, and query a remote host running `pfstatd`.

Summary

`pftop`, `pflogx`, and `pfstat` can be used to help the administrator visualize the traffic flowing through a PF firewall. These utilities are easy to install and configure. If you are using the PF firewall, consider adding them to your administrative toolkit.

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.*

If you wish to contribute to BSD magazine, share your knowledge and skills with other BSD users – do not hesitate – read the guidelines on our website and email us your idea for an article.

Join our team!



Become BSD magazine Author or Betatester

As a betatester you can decide on the contents and the form of our quarterly. It can be you who read the articles before everybody else and suggest the changes to the author.

Contact us:
editors@bsdmag.org
www.bsdmag.org

FreeBSD Programming Primer – Part 4

In the fourth part of our series on programming, we will continue to develop our CMS. Here we will examine how a modern CMS dynamically generates and controls content and implement a similar model in our PHP code.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP, and SQL code

What you should know...

- BSD and general PC administration skills

In the early days of the World Wide Web, HTML pages were literally handcrafted masterpieces of content. Before applications such as Dreamweaver arrived that allowed content providers to design attractive pages with the ease of a document produced in a word processor, it was a matter of writing copious amounts of HTML for each page, checking that the links and the HTML were correct, and repeating for each page. This model was highly inefficient, as not only was a lot of the HTML repeated across pages, the chances of errors coming in and either causing the page to render incorrectly or pointing to the wrong address became greater as the site grew. Managing a website with 100 pages is possible; a website with 10,000 pages a nightmare.

The complex sites we see today on the Internet would be impossible without the Content Management System. Yet even now, large innovative sites are moving away from the CMS model toward frameworks that consider the locally provided content to be only a part of the website with 3rd party content supplying a significant proportion of the content.

While the technology meets the ethos of the web in that data can be shared freely, it poses the web designer and brand manager with a huge challenge – how can we take disparate pieces of content and serve these in a “wrapper” that to our website visitors appears as if it seamlessly represents our brand values? How can we

divorce the business process from the presentation? Is it possible for a website to develop a unique “personality” while at the same time remaining fresh, dynamic and easily changeable?

These hurdles are being overcome with the use of CSS (Cascading Style Sheets) and templating technologies. While the CSS manages the color, fonts, size, etc. of the content, templates allow us to adjust the order and visibility of the content. For example, we want to generate widely different content (both from a stylized and literal

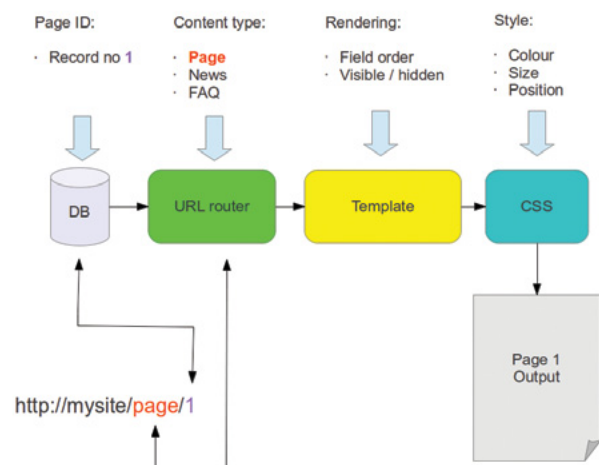


Figure 1. Page generation process

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD



\$99.95

The FreeBSD CD or DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD

Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD.....\$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD.....\$39.95

FreeBSD 9.0 DVD.....\$39.95

FreeBSD Subscriptions

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

FreeBSD Subscription, start with CD 9.1.....\$29.95

FreeBSD Subscription, start with DVD 9.1.....\$29.95

FreeBSD Subscription, start with CD 9.0.....\$29.95

FreeBSD Subscription, start with DVD 9.0.....\$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD.....\$29.95

PC-BSD Subscription.....\$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide).....\$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide).....\$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes).....\$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1.....\$79.95

PC-BSD 9.0 Users Handbook.....\$24.95

BSD Magazine.....\$11.99

The FreeBSD Toolkit DVD.....\$39.95

FreeBSD Mousepad.....\$10.00

FreeBSD & PCBSD Caps.....\$20.00

BSD Daemon Horns.....\$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even MORE items
visit our website today!

www.FreeBSDMall.com

content perspective) depending on website section, page number and content type. See Figure 1 – Page generation process.

MySQL Interface

As it is important that we can quickly test our CMS, for those that would prefer the “Cut, Paste and Click” approach rather than managing long SQL statements via the command line, you can use a lightweight web-based database manager. The lightest of these (a single PHP page) is Adminer. An alternative is SQL buddy, and either of these can be quickly installed if desired by downloading the archive and extracting into a folder under the `/usr/home/dev/data`. The web-based interface can then be accessed from: <http://myserver/dirname>. See Table 1 – Useful links.

Adding New Content Types

At the moment, we only have one content type – a page. This is stored in the pages table and holds the following content as shown in Table 1.

Table 1. Page content from MySQL pages table

id	title	h1	body
1	My first page	Page header	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris interdum auctor tellus sed dignissi...

This results in the following output as seen in Figure 2. Now let us create a second page in our database:

Method 1 – Via CLI

```
$ mysql -uroot -p'cms-password';

mysql> use freebsdcms;
mysql> INSERT INTO `pages` (`title`, `h1`, `body`)
-> VALUES ('My second page', 'H1', '2');
```

Method 2 – Via saved SQL statement

If you prefer, create a SQL file `createpage2.sql` in the SQL directory with the following content:

```
USE freebsdcms;
INSERT INTO `pages` (`title`, `h1`, `body`)
VALUES ('My second page', 'H1', '2');
```

Then execute this at the command line:

```
$ mysql -uroot -p'cms-password' < createpage2.sql
```

Method 3 – Via Adminer / SQL Buddy

Alternatively use the SQL command function in Adminer to execute the following SQL statement:

```
INSERT INTO `pages` (`title`, `h1`, `body`)
VALUES ('My second page', 'H1', '2');
```

Houston, We Have a Problem

We now have two pages in our database, but `index.php` still contains the following code:

```
// Build page - use first record in database
$page['id'] = 1;
buildpage($page);
```

This hard-wires `index.php` to only serve a page with an ID of 1. Depending on the URL passed to the webserver, we want to serve that type of content. For example `http://mysite/pages/1` will serve a page with an ID of 1, whereas `http://mysite/faqs/1` will serve an FAQ with an ID of 1, etc. Visiting `http://mysite` will return the home page (Page 1). This leads us to the next problem – where do we store the content types? We could include this in a separate MySQL table, but this would require an additional SQL query to be executed every time a page is loaded. As content types will not be changed very often, we can create another include file that defines our content

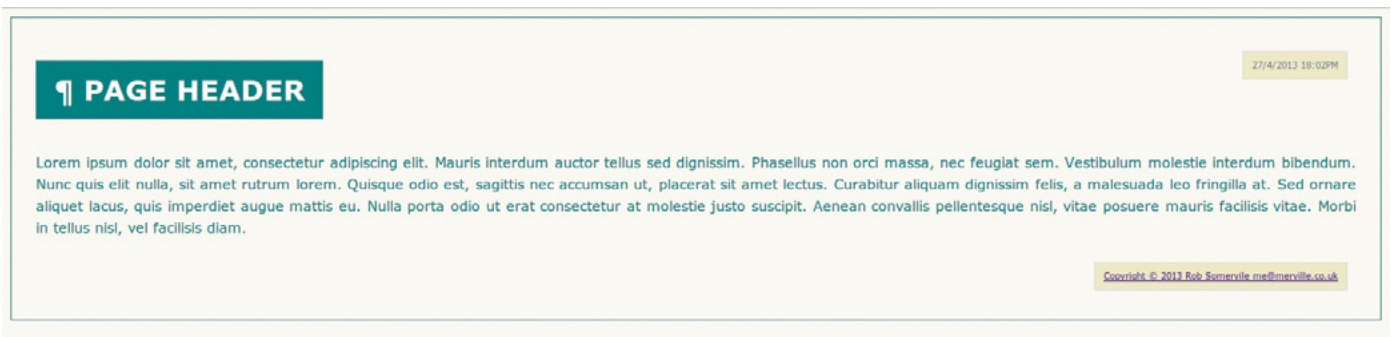


Figure 2. Our first page

Listing 1. content.inc

```
<?php
/*
 *
 * content.inc
 * Defines content types for our CMS
 *
 */
// Define the content type. This must match any tables
//          defined in our
// CMS
$content_types[] = 'page';
$content_types[] = 'faq';
$content_types[] = 'news';
// Map each content type to a table. Each content type
//          must be matched
// to a corresponding table
$content_tables['page'] = 'pages';
$content_tables['faq'] = 'faqs';
$content_tables['news'] = 'news';
```

Listing 2. pages_template.inc

```
<?php
/*
 *
 * pages_template.inc
 * Template for our page content type
 *
 * For content type foo the corresponding template would be:
 * foo_template.inc
 *
 * To display a field:
 * render($theme['name_of_field_as_defined_in_db']);
 *
 * To hide a field omit it from here
 * To change the rendering order, just re-order the fields
 *
 * NOTE: Any content generated by javascript will not be
//          managed here
//          A closing ?> tag is mandatory
 *
 */
render($theme['title']);
render($theme['debug']);
render($theme['h1']);
render($theme['timestamp']);
render($theme['body']);
render($theme['licence']);
?>
```

Listing 3. index.php replacement code

```
// First we need to parse the URL that was passed to us
//          to extract the
// id and the content type.

$URI = $_SERVER['REQUEST_URI'];

if($URI == '/') {
// If this is a request to root (/) redirect to page 1

$request = array('pages',1);
buildpage($request);
}else{
// Parse the request, if it is valid get the content
//          from our DB

$request = parse_request($URI);

if(!is_null($request)){
//          buildpage($request);
}else{
//          echo "Invalid request";
}
}
}
```

Listing 4. core.inc replacement code

```
function buildpage($request) {
// Content definitions
require INCLUDES.'content.inc';

// Routes our incoming request to the right content
//          type and pulls
// the content from our DB.

$content_type = $request[0];
$id = $request[1];
$template_file = TEMPLATES . $content_type . '_
//          template.inc';

// Build the SQL and get the result
```

```

$sql = "SELECT * FROM $content_type WHERE id='$id' LIMIT 1";
$result = mysql_select($sql);

// Check we have some content to display

if($result[0] == 0){

    echo 'No data';
    return;

}

// Check we have a template file

if(!file_exists($template_file)){

    echo 'No template';
    return;

}

// Don't write any output to browser yet as we want
// to post process
// our content using a theme. If enabled use our
// optimization
// callback to remove white space etc.

ob_start("optimize_callback");

// Output our page header

outfile(TEMPLATES . 'header.inc');

// Create our body

echo wraptag('title', $result['title']);
echo HEAD;
echo BODY;

// Generate a unique ID based on content type
// Map the requested content type from our real table name

$ct = array_search($content_type, $content_tables);

echo '<div id="'. $ct. "'>';

// If we are in debug mode, show an alert

if(DEBUG){

    $theme['debug'] = div('&para;', '', 'debug');

}

// Dump the title & id out to our theme template
$theme['id'] = $result['id'];
$theme['title'] = $result['title'];

// As we don't know how many fields we will have in
// our content
// type after our id, iterate through each in turn and wrap
// the field with a div

$offset = $result[1] - 1;
$pos = 0;

foreach($result as $key => $value){

    if($pos > $offset){

        $theme[$key] = div($result[$key], $key.'-'. $id, $key);

    }

    $pos ++;

}

// Add our standard copyright notice

$theme['licence'] = div(ahref(COPYRIGHT, LICENCE, 'Copyright and
licence details'),'','licence');

// Include our template file

require_once($template_file);

// Close our content type tag

echo '</div>';

// Output our HTML page footer
outfile(TEMPLATES . 'footer.inc');

// Flush it all out and display

ob_end_flush();

}

```


Listing 5. core.inc additional code

```

function parse_request($URI) {
    // Returns the type of content and the ID
    // of the content requested.
    // parse_request(/page/1)
    // $array['page'][1]
    // parse_request(/rubbish/123456)
    // NULL

    // Content definitions
    require_once INCLUDES.'content.inc';

    $ct = NULL;
    $id = NULL;
    $valid = 0;

    // Fetch the parameters from the URL

    $array = explode('/', $URI);

    // We don't need the first '/' - delete the first
        empty
    // array item

    $a = array_shift($array);

    // Check we have 2 parameters

    $paramcount = count($array);

    if($paramcount == 2){

        // First test passed - We have 2 parameters

        $valid ++;

        $ct = $array[0];
        $id = $array[1];
    }

    if(in_array($ct, $content_types)){

        // If content type matches our list second test
            passed

        $valid ++;

        // Map the requested content type to our real
            table name

        $array[0] = $content_tables[$ct];
    }

    if(is_numeric($id)){

        // If ID is a number, third test passed

        $valid ++;
    }

    if($valid == 3){

        // Valid parameters passed, return content type
            and page ID

        return $array;
    }else{

        // Test failed - return NULL

        return NULL;
    }

    function optimize_callback($buffer){

        // Replace all spaces and cruft between tags

        if(OPTIMIZE){

            $b = preg_replace('~>\s+<~', '><', $buffer);
            $b = preg_replace('\r\n|\r|\n/', '', $b);
            $b = preg_replace('!\s+!', ' ', $b);

            return $b;
        }
    }
}

```

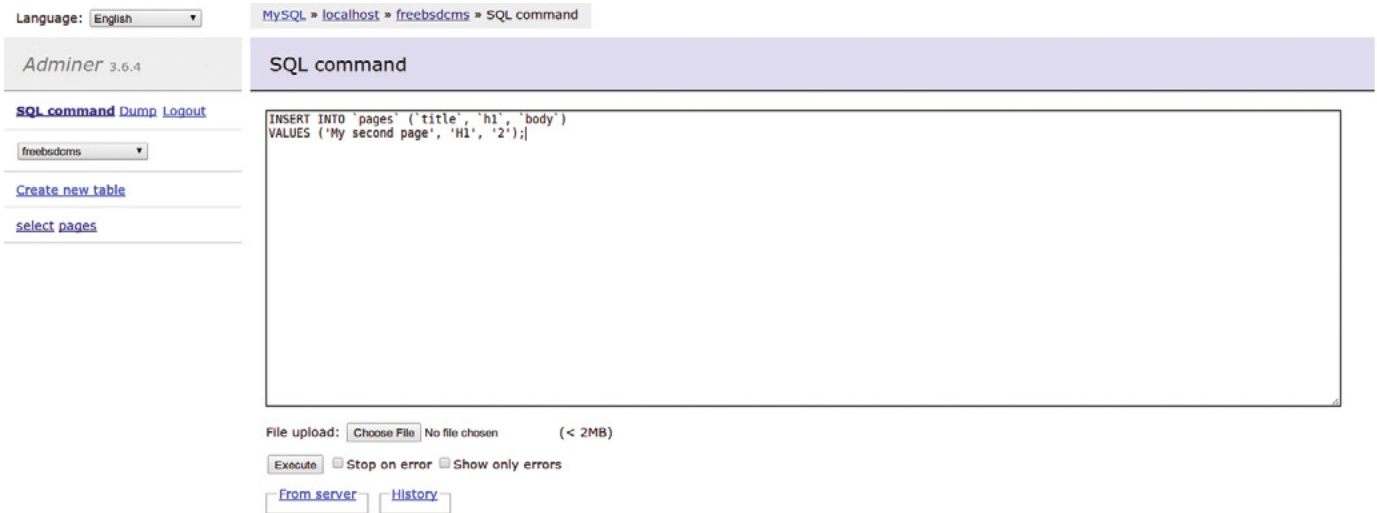


Figure 3. Using Adminer to execute SQL statement

Listing 6. mysql.inc replacement code

```
<?php
/*
 *
 * mysql.inc
 * Contains MySQL functions for our CMS
 *
 */
function mysql_select($sql) {
    $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD, CMSDB);

    if($db->connect_errno > 0){
        die('Unable to connect to database [' .
            $db->connect_error . ']);
    }

    if(!$result = $db->query($sql)){
        if(DEBUG){
            die('There was an error running the query
            [' . $db->error . ']);
        }else{
            die('');
        }
    }

    // Pass our results to an array to be returned

    $r = array();

    $r[] = $result->num_rows;    // No of rows returned

    $r[] = $db->field_count;    // No of columns in table
    $r[] = $db->affected_rows;  // No of rows affected e.g.
                                update / delete

    // Append the results to our result count

    if($result->num_rows != 0){
        $r = array_merge($r, $result->fetch_array(MYSQLI_
            ASSOC));
    }

    // Free the result
    $result->free();

    // Close the connection
    $db->close();

    return $r;
}
```

SPRING FUNDRAISING DRIVE

Your donations have helped make FreeBSD the best OS available! By investing in the services provided by The FreeBSD Foundation you have helped us fund projects to keep FreeBSD a high-performance, secure, and stable OS.

What will the Foundation accomplish with your donation in 2013?

- Software development projects for FreeBSD: \$600,000.
- Paid staff time supporting Release Engineering and Security teams.
- Grow staff: Five technical staff members by year-end.
- Provide support for BSD conferences around the globe, in Europe, Japan, Canada, and the USA.
- Hardware to maintain and improve FreeBSD project infrastructure: \$130,000.
- FreeBSD community growth through marketing and outreach to users and businesses.
- Legal services and counsel protecting the FreeBSD trademarks.



Support FreeBSD by donating

FreeBSD is internationally recognized as an innovative leader in providing a high-performance, secure, and stable operating system. Our mission is to continue and increase our support and funding to keep FreeBSD at the forefront of operating system technology. But, we can't do this without your help!

Last year with your generosity, we raised over \$770,000. This year we will invest \$1,000,000 to support and promote FreeBSD.

We have kicked off the new year with three newly funded projects, and are actively soliciting additional project proposals.

Please support the Foundation during our Spring Fundraising Drive, and help us raise \$100,000 from 1000 donors between April 15th and May 30th.

*We need your help.
We can't do this without you...*

Make your donation today. Go to:

www.freebsd.foundation.org/donate

Then talk to your employer about matching your gift— or making their own donation.

The
FreeBSD
FOUNDATION

Find out more, visit:

www.freebsd.foundation.org

types. We can then automatically use a custom template depending on the content type to post process our specific content.

First of all, we need to make some modifications to Apache so that it serves our `index.php` page as default. Edit the line in `/usr/local/etc/apache22/httpd.conf` to match the following:

```
DirectoryIndex index.php
```

Find the section marked `<Directory "/usr/local/www/apache22/data">` and add the following:

```
#
# Redirect on error via our CMS
#

ErrorDocument 401 /index.php
ErrorDocument 403 /index.php
```

Listing 7. `html.inc` replacement code

```
<?php
/*
 *
 * html.inc
 * Contains core html functions for our CMS
 *
 */

function wraptag($tag, $text) {

    // Wraps $text with compliant tags
    // wraptag('p',sometext)
    // <p>sometext</p>

    return '<' . $tag . '>' . $text . '</' . $tag . '>';
}

function div($divcontent, $class, $id = '') {

    // Generates a div tag $text with compliant tags
    // div('content','class')
    // <div class="class">content</div>
    // div('content','class','id')
    // <div id="id" class="class">content</div>
    // div('content','','id')
    // <div id="id">content</div>
    // div('content','','')
    // <div>content</div>

    if ($id != '') {

        $id = ' id="' . $id . '"';
    }

    if ($class != '') {

        $class = ' class="' . $class . '"';
    }

    return '<div' . $id . $class . '>' .
        $divcontent . '</div>';
}

function ahref($text, $url, $title = '') {

    // Generates an href tag $text with compliant tags
    // ahref('Click here',frebsd.org)
    // <a href="http://frebsd.org" title="Click
    // here">Click here</a>
    // ahref('Click here',frebsd.org,'Link title')
    // <a href="http://frebsd.org" title="Link
    // title">Click here</a>

    if ($title == '') {

        $title = $url;
    }

    $ahref = '<a href="' . $url . '" title="' . $title
        . '">' . $text . '</a>';

    return $ahref;
}

function render($field) {

    // Renders via template

    echo $field;
}
}
```

```
ErrorDocument 404 /index.php
ErrorDocument 500 /index.php
```

This will force all traffic to be passed to our index.php for processing. As root, delete our unwanted files then restart Apache:

```
$ rm /home/dev/data/index.xhtml
$ rm /home/dev/data/index.html
$ apachectl restart
```

When you visit <http://mysite> or <http://mysite/>, page 1 should be displayed. Now for the modifications that will facilitate content type routing and theme control. Create a file in the includes directory called *content.inc* with the content from Listing 1.

Create the following template file *pages_template.inc* in the templates directory shown in Listing 2.

Remove the following section entirely from index.php:

```
// Build page - use first record in database
```

```
$page['id'] = 1;
buildpage($page);
```

Replace with the one shown in Listing 3. Remove entirely the function call `buildpage($page)` from core.inc. Replace with the code shown in Listing 4. Add the function calls from Listing 5 to the end of core.inc.

Replace html.inc with Listing 7. Append the following to cms.inc:

```
// Optimize output by removing white space between tags etc.
define("OPTIMIZE", true);
```

Errata

In the previous article of this series the following syntax was incorrect:

```
#dev mysql -u root password 'cms-password' <
    createdb.sql
#dev mysql -u root password 'cms-password' < createpagetbl.sql
#dev mysql -u root password 'cms-password' < createpage.sql
```

It should have read:

```
#dev mysql -u root -p'cms-password' < createdb.sql
#dev mysql -u root -p'cms-password' < createpagetbl.sql
#dev mysql -u root -p'cms-password' < createpage.sql
```

Our apologies.

Useful Links

- SQL buddy – <http://sqlbuddy.com>
- Adminer – <http://www.adminer.org>

Testing and Adding New Content

That is a lot of code we have added, but we now have a major jump in functionality. We can create any number of content types now by creating a new table (e.g. *faq*, *news*, etc.) The only essential fields we must define are ID and TITLE. After these two fields you may define as many or as few as you require. You will need to create a matching template file with the fields you want to display or else the content will be unable to render. Once you have added new records to your content type (Adminer makes this quick and easy), the content can be accessed via your browser at: <http://mysite/mycontenttype/mypageid>. If you attempt to access invalid content, you will be presented with a rudimentary error message.

In the next article in the series, we will look at theming in detail and how we can lay out the site using a combination of templates and CSS.

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.

DTrace

A Deeper Approach

In my article “Intro to DTrace”, published in May 2012 in BSD Magazine, I described DTrace all the way from configuring your system to enabling DTrace probes to executing some D scripts in order to show you some DTrace features. This article will take a deeper approach to DTrace.

The processing and buffering of all probe data takes place in the DTrace kernel module. Each probe definition is composed of the four elements separated by colons. The general form is:

```
provider:module:function:name
```

Provider

A provider is a DTrace kernel module, which logically groups together various probes that are related. Exam-

Table 1. *D Macro Variables*

Name	Description	Reference
<code>\$(0-9)+</code>	macro arguments	look at macros
<code>\$egid</code>	effective group-ID	<code>getegid(2)</code>
<code>\$euid</code>	effective user-ID	<code>geteuid(2)</code>
<code>\$gid</code>	real group-ID	<code>getgid(2)</code>
<code>\$pid</code>	process ID	<code>getpid(2)</code>
<code>\$pgid</code>	process group ID	<code>getpgid(2)</code>
<code>\$ppid</code>	parent process ID	<code>getppid(2)</code>
<code>\$projid</code>	project ID	<code>getprojid(2)</code>
<code>\$sid</code>	session ID	<code>getsid(2)</code>
<code>\$target</code>	target process ID	see target process id
<code>\$taskid</code>	task ID	<code>gettaskid(2)</code>
<code>\$uid</code>	real user-ID	<code>getuid(2)</code>

ples of providers in DTrace include: `fbt`, which instruments kernel functions; `pid`, which instruments userland processes; and `syscall` which instruments system calls.

Module

A module is the program location of the group of probes. This could be the name of a kernel module where the probes exist, or it could be a userland library. Example modules are the `libc.so` library or the `ufs` kernel module.

Function

Specifies the specific function which this probe should fire on. This could be something like a particular function in a library such as `printf()` or `strcpy()`.

Name

This is usually the meaning of the probe. Sample names are “entry” or “return” for a function or “start” for an I/O probe. For instruction level tracing, this field specifies the offset within the function. Understanding this allows you to understand the purpose of a particular probe. You can list all the probes on a DTrace instrumented system by provider by running the `dtrace -l` command. It will list the probes in the format described above. If one of them is missing, it will be taken as a wildcard. It could be written as:

```
provider::function:name or provider:*:function:name
```

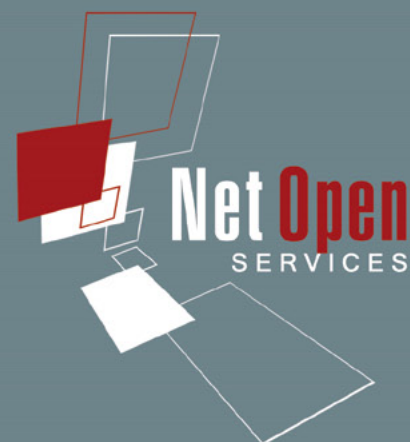


NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)
- REMOTE MONITORING AND MANAGEMENT 24/7
- NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)
- OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)
- DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)
- MANAGED HOSTING IN CARRIER CLASS DATA CENTERS
- DISASTER RECOVERY



WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. NETOPENSERVICES TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS 15 YEARS OF EXPERIENCE IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. NETOPENSERVICES DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

Table 2. *DTrace Built-in Variables*

Type and Name	Description
int64_t arg0, ..., arg9	The first ten input arguments to a probe represented as raw 64-bit integers. If fewer than ten arguments are passed to the current probe, the remaining variables return zero.
args[]	The typed arguments to the current probe, if any. The args[] array is accessed using an integer index, but each element is defined to be the type corresponding to the given probe argument. For example, if args[] is referenced by a read(2) system call probe, args[0] is of type int, args[1] is of type void *, and args[2] is of type size_t.
uintptr_t caller	The program counter location of the current thread just before entering the current probe.
chipid_t chip	The CPU chip identifier for the current physical chip.
processorid_t cpu	The CPU identifier for the current CPU.
cpuinfo_t *curcpu	The CPU information for the current CPU.
lwpsinfo_t *curlwpsinfo	The lightweight process (LWP) state of the LWP associated with the current thread. This structure is described in further detail in the proc(4) man page.
psinfo_t *curpsinfo	The process state of the process associated with the current thread. This structure is described in further detail in the proc(4) man page.
kthread_t *curthread	The address of the operating system kernel's internal data structure for the current thread, the kthread_t. The kthread_t is defined in <sys/thread.h>. Refer to Solaris Internals for more information on this variable and other operating system data structures.
string cwd	The name of the current working directory of the process associated with the current thread.
uint_t epid	The enabled probe ID (EPID) for the current probe. This integer uniquely identifies a particular probe that is enabled with a specific predicate and set of actions.
int errno	The error value returned by the last system call executed by this thread.
string execname	The name that was passed to exec(2) to execute the current process.
gid_t gid	The real group ID of the current process.
uint_t id	The probe ID for the current probe. This ID is the system-wide unique identifier for the probe as published by DTrace and listed in the output of dtrace -l.
uint_t ipl	The interrupt priority level (IPL) on the current CPU at probe firing time. Refer to Solaris Internals for more information on interrupt levels and interrupt handling in the illumos operating system kernel.
lgrp_id_t lgrp	The latency group ID for the latency group of which the current CPU is a member.
pid_t pid	The process ID of the current process.
pid_t ppid	The parent process ID of the current process.
string probefunc	The function name portion of the current probe's description.
string probemod	The module name portion of the current probe's description.
string probename	The name portion of the current probe's description.
string probeprov	The provider name portion of the current probe's description.
psetid_t pset	The processor set ID for the processor set containing the current CPU.
string root	The name of the root directory of the process associated with the current thread.
uint_t stackdepth	The current thread's stack frame depth at probe firing time.
id_t tid	The thread ID of the current thread. For threads associated with user processes, this value is equal to the result of a call to pthread_self(3C).
uint64_t timestamp	The current value of a nanosecond timestamp counter. This counter increments from an arbitrary point in the past and should only be used for relative computations.
uid_t uid	The real user ID of the current process.
uint64_t uregs[]	The current thread's saved user-mode register values at probe firing time. Use of the uregs[] array is discussed in
uint64_t vmregs[]	The current thread's active virtual machine register values at probe firing time.
uint64_t vtimestamp	The current value of a nanosecond timestamp counter that is virtualized to the amount of time that the current thread has been running on a CPU, minus the time spent in DTrace predicates and actions. This counter increments from an arbitrary point in the past and should only be used for relative time computations.
uint64_t walltimestamp	The current number of nanoseconds since 00:00 Universal Coordinated Time, January 1, 1970.

Macro Variables

The D compiler defines a set of built-in macro variables that you can use when writing D programs or interpreter files. Macro variables are identifiers that are prefixed with a dollar sign (\$) and are expanded once by the D compiler when processing your input file. The D compiler provides the following macro variables, shown in Table 1.

Built-in Variables

Table 2 provides a complete list of D built-in variables. All of these variables are scalar global variables; no thread-local or clause-local variables or built-in associative arrays are currently defined by D.

Macro Arguments

The D compiler also provides a set of macro variables corresponding to any additional argument operands specified as part of the dtrace command invocation. These macro arguments are accessed using the built-in names \$0 for name of the D program file or dtrace command, \$1 for the first additional operand, \$2 for the second operand, and so on. If you use the dtrace -s option, \$0 expands to the value of the name of the input file used with this option. For D programs specified on the command-line, \$0 expands to the value of argv[0] used to exec DTrace itself.

For example:

```
#!/usr/sbin/dtrace -s

syscall::write:entry
/pid == $1/
{
}
```

Target Process ID

Use the \$target macro variable to create scripts that can be applied to a particular user process of interest that is selected on the DTrace command line using the -p option or created using the -c option. The D programs specified on the command line or using the -s option are compiled after processes are created or grabbed and the \$target variable expands to the integer process-ID of the first such process. For example, the following D script could be used to determine the distribution of system calls executed by a particular subject process:

```
syscall::entry
/pid == $target/
{
    @[probefunc] = count();
}
```

Subroutines

Subroutines differ from actions because they generally only affect internal DTrace state. Therefore, there are no destructive subroutines, and subroutines never trace data into buffers. Many subroutines have analogs in the Section 9F or Section 3C interfaces...

alloca

```
void *alloca(size_t size)
```

alloca allocates *size* bytes out of scratch space, and returns a pointer to the allocated memory. The returned pointer is guaranteed to have 8-byte alignment. Scratch space is only valid for the duration of a clause. Memory allocated with alloca will be deallocated when the clause completes. If insufficient scratch space is available, no memory is allocated and an error is generated.

basename

```
string basename(char *str)
```

basename is a D analogue for basename(1). This subroutine creates a string that consists of a copy of the specified string, but without any prefix that ends in /. The returned string is allocated out of scratch memory, and is therefore valid only for the duration of the clause. If insufficient scratch space is available, basename does not execute and an error is generated.

bcopy

```
void bcopy(void *src, void *dest, size_t size)
```

bcopy copies *size* bytes from the memory pointed to by *src* to the memory pointed to by *dest*. All of the source memory must lie outside of scratch memory and all of the destination memory must lie within it. If these conditions are not met, no copying takes place and an error is generated.

cleanpath

```
string cleanpath(char *str)
```

cleanpath creates a string that consists of a copy of the path indicated by *str*, but with certain redundant elements eliminated. In particular ../ elements in the path are removed, and ./ elements are collapsed. The collapsing of ../ elements in the path occurs without regard to symbolic links. Therefore, it is possible that cleanpath-

could take a valid path and return a shorter, invalid one. For example, if `str` was `/foo/./bar` and `/foo` was a symbolic link to `/net/foo/export`, `cleanpath` would return the string `/bar` even though `bar` might only be in `/net/foo/`. This limitation is due to the fact that `cleanpath` is called in the context of a firing probe, where full symbolic link resolution or arbitrary names is not possible. The returned string is allocated out of scratch memory, and is therefore valid only for the duration of the clause. If insufficient scratch space is available, `cleanpath` does not execute and an error is generated.

copyin

```
void *copyin(uintptr_t addr, size_t size)
```

`copyin` copies the specified size in bytes from the specified user address into a DTrace scratch buffer and returns the address of this buffer. The user address is interpreted as an address in the space of the process associated with the current thread. The resulting buffer pointer is guaranteed to have 8-byte alignment. The address in question must correspond to a faulted-in page in the current process. If the address does not correspond to a faulted-in page, or if insufficient scratch space is available, `NULL` is returned, and an error is generated. See Chapter 33, User Process Tracing for techniques to reduce the likelihood of `copyin` errors.

copyinstr

```
string copyinstr(uintptr_t addr)
```

`copyinstr` copies a null-terminated C string from the specified user address into a DTrace scratch buffer and returns the address of this buffer. The user address is interpreted as an address in the space of the process associated with the current thread. The string length is limited to the value set by the `strsize` option. As with `copyin`, the specified address *must* correspond to a faulted-in page in the current process. If the address does not correspond to a faulted-in page, or if insufficient scratch space is available, `NULL` is returned and an error is generated.

copyinto

```
void copyinto(uintptr_t addr, size_t size, void *dest)
```

`copyinto` copies the specified size in bytes from the specified user address into the DTrace scratch buffer

specified by `dest`. The user address is interpreted as an address in the space of the process associated with the current thread. The address in question *must* correspond to a faulted-in page in the current process. If the address does not correspond to a faulted-in page, or if any of the destination memory lies outside scratch space, no copying takes place and an error is generated.

dirname

```
string dirname(char *str)
```

`dirname` is a D analogue for `dirname(1)`. This subroutine creates a string that consists of all but the last level of the pathname specified by `str`. The returned string is allocated out of scratch memory, and is therefore valid only for the duration of the clause. If insufficient scratch space is available, `dirname` does not execute and an error is generated.

lltostr

```
string lltostr(long long num)
string lltostr(long long num, int base)
```

`lltostr` is a D analogue for `strtoll()`. This subroutine creates a string that represents the value of `num`. If `base` is specified, then `num` is interpreted in that base.

msgdsize

```
size_t msgdsize(mblk_t *mp)
```

`msgdsize` returns the number of bytes in the data message pointed to by `mp`. See `msgdsize(9F)` for details. `msgdsize` only includes data blocks of type `M_DATA` in the count.

msgsize

```
size_t msgsize(mblk_t *mp)
```

`msgsize` returns the number of bytes in the message pointed to by `mp`. Unlike `msgdsize`, which returns only the number of `data` bytes, `msgsize` returns the `total` number of bytes in the message.

mutex_owned

```
int mutex_owned(kmutex_t *mutex)
```

`mutex_owned` is an implementation of `mutex_owned(9F)`. `mutex_owned` returns non-zero if the calling thread cur-



rently holds the specified kernel mutex or zero if the specified adaptive mutex is currently unowned.

mutex_owner

```
kthread_t *mutex_owner(kmutex_t *mutex)
```

mutex_owner returns the thread pointer of the current owner of the specified adaptive kernel mutex. mutex_owner returns NULL if the specified adaptive mutex is currently unowned or if the specified mutex is a spin mutex. See mutex_owned(9F).

mutex_type_adaptive

```
int mutex_type_adaptive(kmutex_t *mutex)
```

mutex_type_adaptive returns non-zero if the specified kernel mutex is of type MUTEX_ADAPTIVE, or zero if it is not. Mutexes are adaptive if they meet one or more of the following conditions:

- The mutex is declared statically
- The mutex is created with an interrupt block cookie of NULL
- The mutex is created with an interrupt block cookie that does not correspond to a high-level interrupt

See mutex_init(9F) for more details on mutexes. The majority of mutexes in the illumos kernel are adaptive.

progenyof

```
int progenyof(pid_t pid)
```

progenyof returns non-zero if the calling process (the process associated with the thread that is currently trig-

Table 3. SPARC uregs[] Constants

Constant	Register
R_G0..R_G7	%g0..%g7 global registers
R_O0..R_O7	%o0..%o7 out registers
R_L0..R_L7	%l0..%l7 local registers
R_I0..R_I7	%i0..%i7 in registers
R_CCR	%ccr condition code register
R_PC	%pc program counter
R_NPC	%npc next program counter
R_Y	%y multiply/divide register
R_ASI	%asi address space identifier register
R_FPRS	%fprs floating-point registers state

[GEEKED AT BIRTH]

IM Geek f



You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

- LEARN:
- Advancing Computer Science
 - Artificial Life Programming
 - Digital Media
 - Digital Video
 - Enterprise Software Development
 - Game Art and Animation
 - Game Design
 - Game Programming
 - Human-Computer Interaction
 - Network Engineering
 - Network Security
 - Open Source Technologies
 - Robotics and Embedded Systems
 - Serious Game and Simulation
 - Strategic Technology Development
 - Technology Forensics
 - Technology Product Design
 - Technology Studies
 - Virtual Modeling and Design
 - Web and Social Media Technologies

gering the matched probe) is among the progeny of the specified process ID.

rand

```
int rand(void)
```

`rand` returns a pseudo-random integer. The number returned is a weak pseudo-random number and should not be used for any cryptographic application.

rw_iswriter

```
int rw_iswriter(krwlock_t *rwlock)
```

`rw_iswriter` returns non-zero if the specified reader-writer lock is either held or desired by a writer. If the lock is held only by readers and no writer is blocked or if the lock is not held at all, `rw_iswriter` returns zero. See `rw_init(9F)`.

rw_write_held

```
int rw_write_held(krwlock_t *rwlock)
```

`rw_write_held` returns non-zero if the specified reader-writer lock is currently held by a writer. If the lock is held only by readers or not held at all, `rw_write_held` returns zero. See `rw_init(9F)`.

speculation

```
int speculation(void)
```

`speculation` reserves a speculative trace buffer for use with `speculate` and returns an identifier for this buffer.

strjoin

```
string strjoin(char *str1, char *str2)
```

`strjoin` creates a string that consists of `str1` concatenated with `str2`. The returned string is allocated out of scratch memory and is therefore valid only for the duration of the clause. If insufficient scratch space is available, `strjoin` does not execute and an error is generated.

strlen

```
size_t strlen(string str)
```

Table 4. *x86 uregs[] Constants*

Constant	Register
R_CS	%cs
R_GS	%gs
R_ES	%es
R_DS	%ds
R_EDI	%edi
R_ESI	%esi
R_EBP	%ebp
R_EAX	%eax
R_ESP	%esp
R_EAX	%eax
R_EBX	%ebx
R_ECX	%ecx
R_EDX	%edx
R_TRAPNO	%trapno
R_ERR	%err
R_EIP	%eip
R_CS	%cs
R_ERR	%err
R_EFL	%efl
R_UESP	%uesp
R_SS	%ss

Table 5. *amd64 uregs[] Constants*

Constant	Register
R_RSP	%rsp
R_RFL	%rfl
R_RIP	%rip
R_RAX	%rax
R_RCX	%rcx
R_RDX	%rdx
R_RBX	%rbx
R_RBP	%rbp
R_RSI	%rsi
R_RDI	%rdi
R_R8	%r8
R_R9	%r9
R_R10	%r10
R_R11	%r11
R_R12	%r12
R_R13	%r13
R_R14	%r14
R_R15	%r15

`strlen` returns the length of the specified string in bytes, excluding the terminating null byte.

tolower

```
string (char *str)
```

`tolower` returns a new string which is the lowercase version of `str`.

toupper

```
string (char *str)
```

`toupper` returns a new string which is the uppercase version of `str`.

Creating Debugging Tools

First Case Scenario

Let's suppose we have an application that segfaults when trying to execute an instruction at address `0x40404040`, this is clearly an overflow. With DTrace, we can stop the program before it crashes trying to execute the instruction at this address. This allows us to carry out data collection and analysis, such as printing CPU register values, function parameters, dumping memory:

```
#/usr/sbin/dtrace -s
pid$target:a.out::return
/ uregs[R_EIP] == 0x40404040 / {
    printf("I'm going to crash !!!");
    printf("Module: %s Function %s", probemod, probefunc);
@[ustack(10)]=count(); // 10 deep userland stack
}
```

Here is where `R_EIP` constant came from:

`uregs[]` Array

The `uregs[]` array enables you to access individual user registers. The following tables list indices into the `uregs[]` array corresponding to each supported Solaris system architecture. On AMD64 platforms, the `uregs` ar-

Table 6. Common `uregs[]` Constants

Constant	Register
<code>R_PC</code>	program counter register
<code>R_SP</code>	stack pointer register
<code>R_R0</code>	first return code
<code>R_R1</code>	second return code

References

- <http://dtrace.org/>
- <https://wikis.oracle.com/display/DTrace>
- <http://bsdmag.org/magazine/1800-bsd-security-protect-your-bsd>

ray has the same content as it does on x86 platforms, plus the additional elements listed in Table 5. The aliases listed in Table 6 can be used on all platforms.

Second Case Scenario

You want to take a look at every string that is being written, as you have encountered that a file that has been corrupted by the word "COW".

```
syscall::write:entry
{
    if(copyinstr(arg1) == "COW")
    {
        printf(" some one wrote COW ");
        ustack(); //--> check user stack
    }
}
```

Third Case Scenario

Let's check malloc return pointer and size requested. Nice for quick debugging

```
pid$target::malloc:entry{
    self->trace = 1;
    self->size = arg0;
}
pid$target::malloc:return
/self->trace == 1/
{
    ustack(1);
    printf("malloc return: <ptr=0x%p> <size=%d>", arg1, self->size);
    self->trace = 0;
    self->size = 0;
}
```

Hope this was as useful for you as it was for me! Now it's just a matter of really what you want to look at with DTrace.

CARLOS ANTONIO NEIRA

Carlos Antonio Neira is a C, Unix and Mainframe developer. He develops in asm and does some kernel development for a living. In his free time he contributes to open source projects. Apart from that, he spends his time on testing and experimenting with his machines. What gives him a lot of enjoyment is solving old problems with new ideas. You may reach him at: cneirabustos@gmail.com.

MAGAZINE

BSD

In the next issue:

- **Basic Applications in BSD OS**
- **FreeBSD in Xen Cloud Platform**
- **OSSEC on NanoBSD**

**Next issue is coming
in June!**



VERISIGN®

vBSDCon • SAVE THE DATE!

DULLES, VA • OCTOBER 25-27, 2013

Please join us **October 25-27, 2013 at the Hyatt in Dulles, Virginia for the first biennial vBSDCon event.** This exciting weekend will bring together members of the BSD community for a series of roundtable discussions, educational sessions, best practice conversations, and exclusive networking opportunities. See below for details on this industry weekend not to be missed:

AGENDA

- **Friday, October 25:** Evening Reception
- **Saturday, October 26:** General Session, Birds of a Feather Sessions
- **Sunday, October 27:** General Session, Breakout Sessions

WHO SHOULD ATTEND

- Developers
- Engineers
- Administrators
- Innovators

TOPICS

- PkgNG w/ Baptiste Daroussin
- A comprehensive look at bsdinstall with Devin Teske
- Netflix Demo/Presentation with Scott Long
- netmap with Luigi Rizzo
- Migration from GCC to LLVM/Clang with David Chisnall

REGISTRATION INFORMATION WILL BE SENT TO YOU IN MAY!

Questions? Please contact: eventsteam@verisign.com



VerisignInc.com



Headquarters:
San Jose, CA



855.GREP.4.IX | Contact Us

99% Compatibility

online now...

IXSYSTEMS AND YOU ARE
THE PERFECT MATCH



SHARED INTERESTS

- Enterprise Storage Solutions
- Personalized Customer Service
- Bold New Information Technology

I'm a

In

Looking for

- A Technology Partner
- More Technical Experience
- New Business Opportunities

Visit Today!



iXsystems

Technology Partner Seeking
Resellers/Integrators for
TrueNAS™ Storage Appliance



WWW.IXSYSTEMS.COM/PERFECTMATCH

