

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

JANUARY 2013 | ISSUE 225 | www.linuxjournal.com

Phone
for
Smartphone
Application
Development

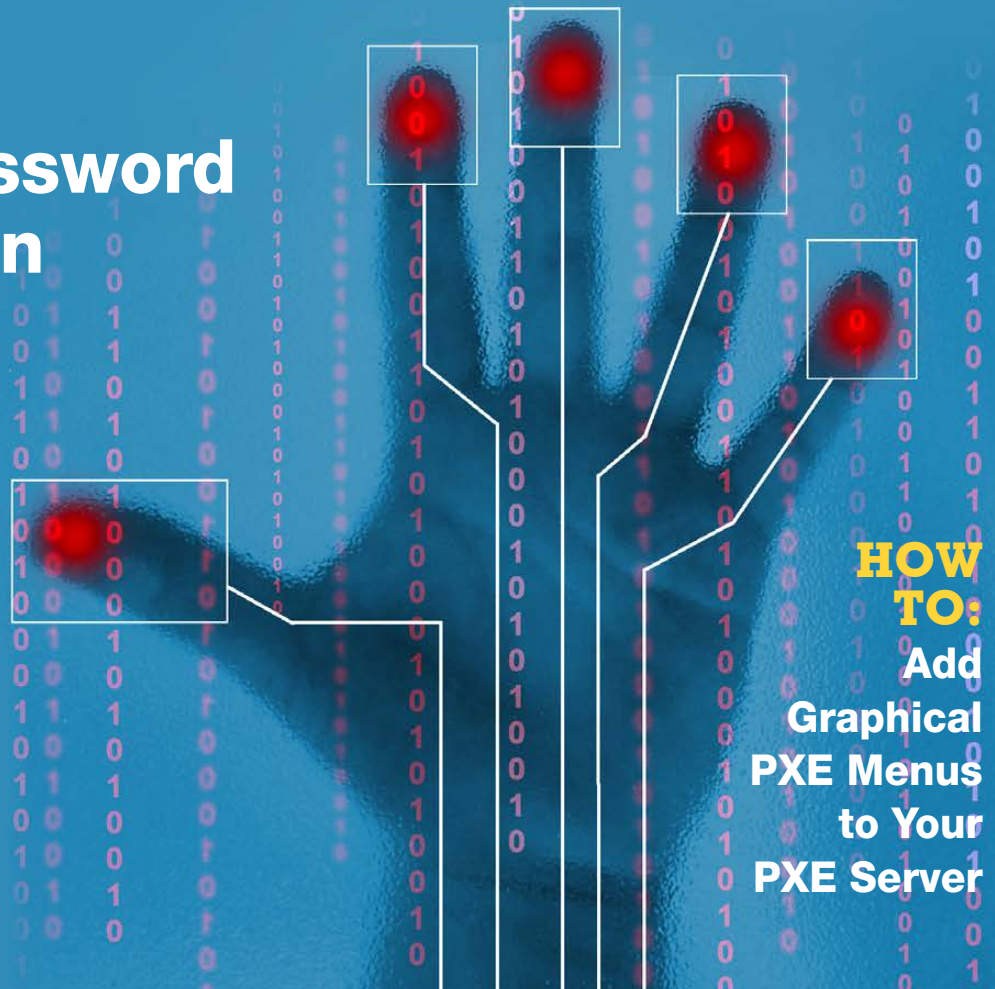
FREE TO
SUBSCRIBERS
EPUB, Kindle, Android, iPhone & iPad editions

SECURITY

Configure
One-Time Password
Authentication
with **OTPW**

Provide Stronger
Security with
**Elliptic Curve
Cryptography**

Project: Build a
Wi-Fi Honeypot



**HOW
TO:**
Add
Graphical
PXE Menus
to Your
PXE Server

**BEST PRACTICES
FOR CREATING
PASSWORDS**

**WORKING
WITH WEB
SOCKETS**

**PLUS: MORE
SHELL SCRIPT
GAMES**



**THIS TIME
WE'RE TURNING IT UP
TO 11!**

The 11th Annual
Southern California Linux Expo

Featuring Performances by:

- * Matthew Garrett
- * 70 other Open Source Acts!

With Special Events by:

- * PostgreSQL and MySQL Community Days
- * DevOps Day LA
- * UbuCon
- * SCALE U
- and more...



SCALE 11x

February 22-24, 2013
Hilton Hotel @ LAX
Los Angeles, CA

<http://www.socallinuxexpo.org>
Use Promo Code LJ11X for a 30%
discount on admission to SCALE

SILICON MECHANICS



visit us at www.siliconmechanics.com or call us toll free at 888-352-1173
RACKMOUNT SERVERS STORAGE SOLUTIONS HIGH-PERFORMANCE COMPUTING

**"Just because
it's badass,
doesn't mean
it's a game."**

Pierre, our new Operations Manager, is always looking for the right tools to get more work done in less time. That's why he respects NVIDIA® Tesla® GPUs: he sees customers return again and again for more server products featuring hybrid CPU / GPU computing, like the Silicon Mechanics Hyperform HPCg R2504.v3.

We start with your choice of two state-of-the-art processors, for fast, reliable, energy-efficient processing. Then we add four NVIDIA® Tesla® GPUs, to dramatically accelerate parallel processing for applications like ray tracing and finite element analysis. Load it up with DDR3 memory, and you have herculean capabilities and an 80 PLUS Platinum Certified power supply, all in the space of a 4U server.



When you partner with Silicon Mechanics, you get more than stellar technology - you get an Expert like Pierre.

Expert included.

CONTENTS

JANUARY 2013
ISSUE 225

SECURITY

FEATURES

68 Elliptic Curve Cryptography

OpenSSH now supports Elliptic Curve Cryptography. Here are some reasons why you should care.

Joe Hendrix

76 Configuring One-Time Password Authentication with OTPW

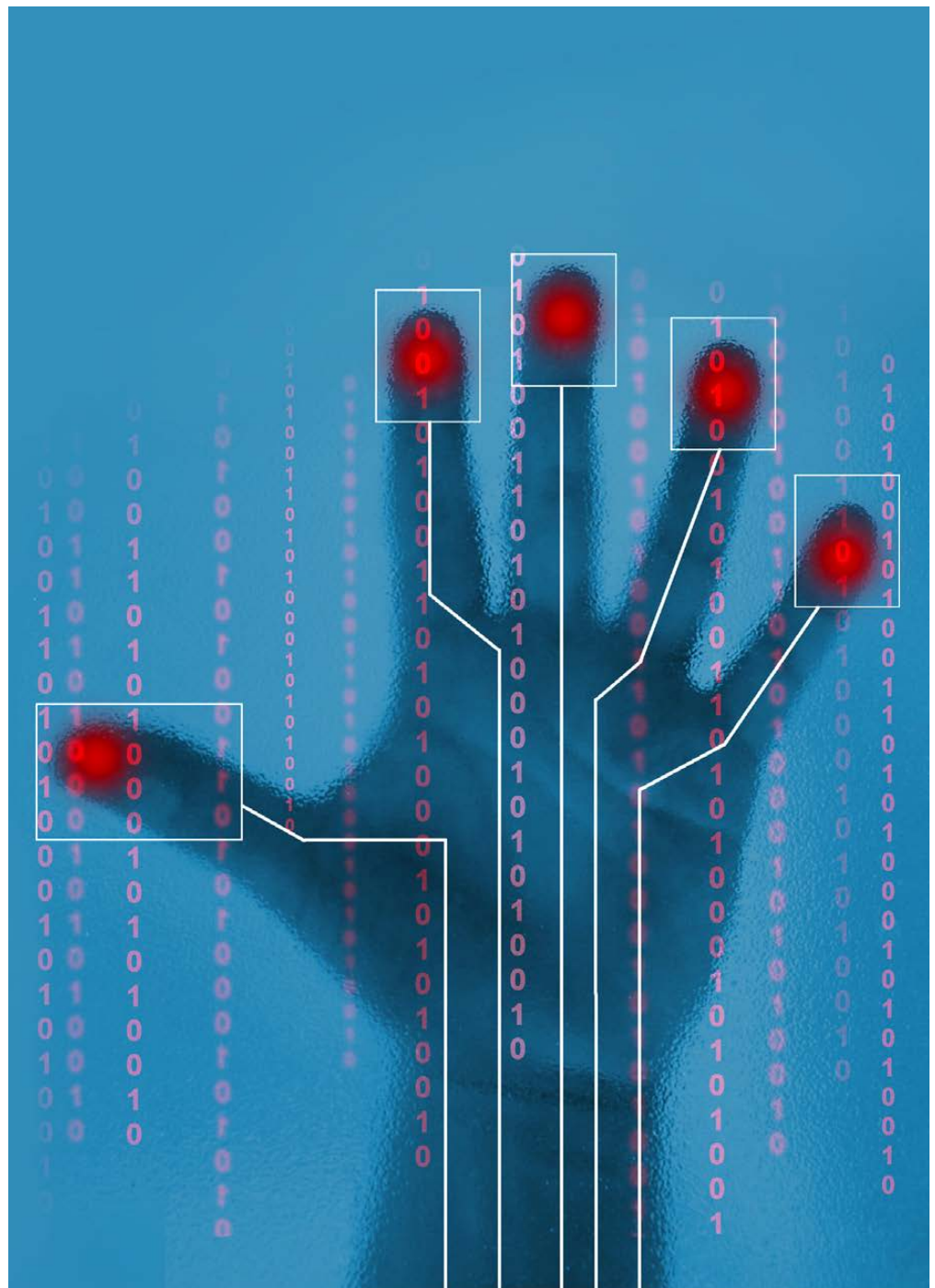
Log in safely from untrusted hosts.

Todd A. Jacobs

90 Wi-Fi Mini Honeypot

Make your network more secure and have a little fun at the same time with a Wi-Fi honeypot.

Marcin Teodorczyk



COVER IMAGE: © Can Stock Photo Inc. / samc352

INDEPTH

102 Phonegap Application Development

With Phonegap, you can write your application and take it with you.

Mike Diehl

COLUMNS

34 Reuven M. Lerner's At the Forge Real-Time Messaging

44 Dave Taylor's Work the Shell Counting Cards: *Cribbage*

48 Kyle Rankin's Hack and / More PXE Magic

58 Shawn Powers' The Open-Source Classroom The Secret Password Is...

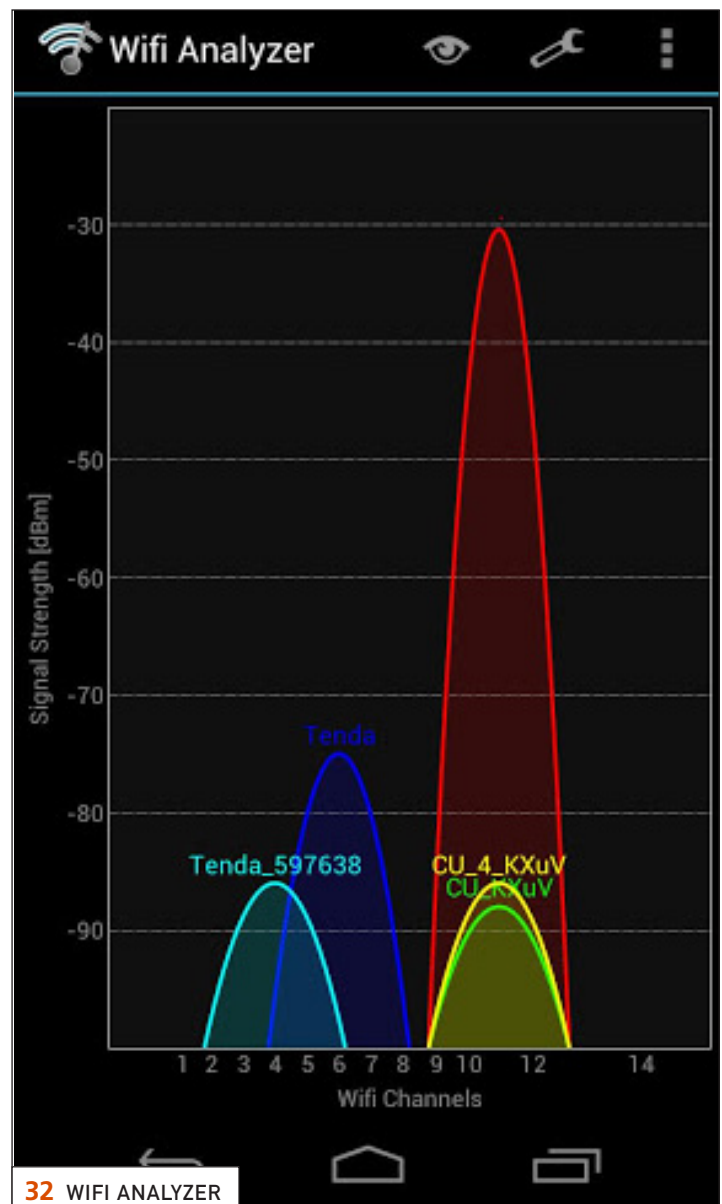
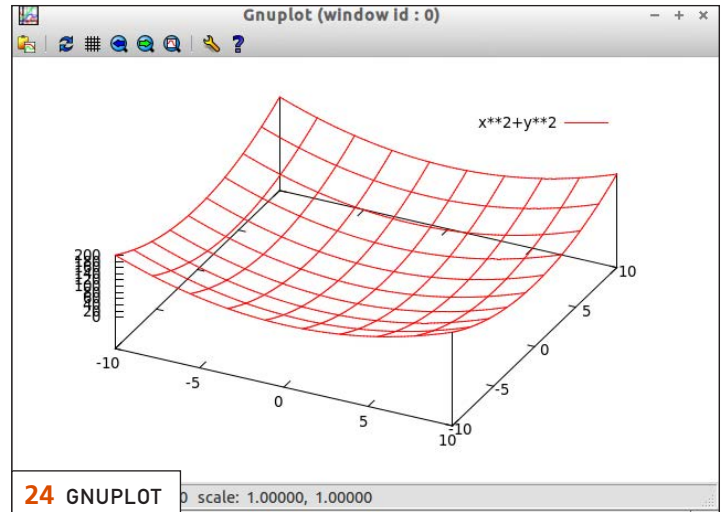
110 Doc Searls' EOF On Infrastructure, Geology and Other Temporary Things

IN EVERY ISSUE

- 8 Current_Issue.tar.gz
- 10 Letters
- 18 UPFRONT
- 32 Editors' Choice
- 64 New Products
- 117 Advertisers Index

ON THE COVER

- Phonegap for Easy Smartphone Application Development, p. 102
- Configure One-Time Password Authentication with OTPW, p. 76
- Provide Stronger Security with Elliptic Curve Cryptography, p. 68
- Project: Build a Wi-Fi Honeypot, p. 90
- How-To: Add Graphical PXE Menus to Your PXE Server, p. 48
- Best Practices for Creating Passwords, p. 58
- Working with Web Sockets, p. 34
- Plus: More Shell Script Games, p. 44



LINUX JOURNAL™

Subscribe to
Linux Journal
Digital Edition
for only
\$2.45 an issue.



ENJOY:

- Timely delivery
- Off-line reading
- Easy navigation
- Phrase search and highlighting
- Ability to save, clip and share articles
- Embedded videos
- Android & iOS apps, desktop and e-Reader versions

SUBSCRIBE TODAY!

LINUX JOURNAL

Executive Editor	Jill Franklin jill@linuxjournal.com
Senior Editor	Doc Searls doc@linuxjournal.com
Associate Editor	Shawn Powers shawn@linuxjournal.com
Art Director	Garrick Antikajian garrick@linuxjournal.com
Products Editor	James Gray newproducts@linuxjournal.com
Editor Emeritus	Don Marti dmarti@linuxjournal.com
Technical Editor	Michael Baxter mab@cruzio.com
Senior Columnist	Reuven Lerner reuven@lerner.co.il
Security Editor	Mick Bauer mick@visi.com
Hack Editor	Kyle Rankin lj@greenfly.net
Virtual Editor	Bill Childers bill.childers@linuxjournal.com

Contributing Editors

Ibrahim Haddad • Robert Love • Zack Brown • Dave Phillips • Marco Fioretti • Ludovic Marcotte
Paul Barry • Paul McKenney • Dave Taylor • Dirk Elmendorf • Justin Ryan

Publisher	Carlie Fairchild publisher@linuxjournal.com
------------------	--

Advertising Sales Manager	Rebecca Cassity rebecca@linuxjournal.com
----------------------------------	---

Associate Publisher	Mark Irgang mark@linuxjournal.com
----------------------------	--------------------------------------

Webmistress	Katherine Druckman webmistress@linuxjournal.com
--------------------	--

Accountant	Candy Beauchamp acct@linuxjournal.com
-------------------	--

**Linux Journal is published by, and is a registered trade name of,
Belltown Media, Inc.**

PO Box 980985, Houston, TX 77098 USA

Editorial Advisory Panel

Brad Abram Baillio • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faltys • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruiuzenga • David A. Lane
Steve Marquez • Dave McAllister • Carson McDonald • Craig Oda
Jeffrey D. Parent • Charnell Pugsley • Thomas Quinlan • Mike Roberts
Kristin Shoemaker • Chris D. Stark • Patrick Swartz • James Walker

Advertising

E-MAIL: ads@linuxjournal.com
URL: www.linuxjournal.com/advertising
PHONE: +1 713-344-1956 ext. 2

Subscriptions

E-MAIL: subs@linuxjournal.com
URL: www.linuxjournal.com/subscribe
MAIL: PO Box 980985, Houston, TX 77098 USA

LINUX is a registered trademark of Linus Torvalds.

TrueNAS™ Storage Appliances Harness the Cloud



Unified. Scalable. Flexible.

Thanks to the Intel® Xeon® Processor 5600 series and high-performance flash, every TrueNAS Storage appliance delivers the utmost in throughput and IOPS.

As IT infrastructure becomes increasingly virtualized, effective storage has become a critical requirement. iXsystems' TrueNAS Storage appliances offer high-throughput, low-latency backing for popular virtualization programs such as Hyper-V, VMWare®, and Xen®. TrueNAS hybrid storage technology combines memory, NAND flash, and traditional hard disks to dramatically reduce the cost of operating a high performance storage infrastructure. Each TrueNAS appliance can also serve multiple types of clients simultaneously over both iSCSI and NFS, making TrueNAS a flexible solution for your enterprise needs.

For growing businesses that are consolidating infrastructure, the **TrueNAS Pro** is a powerful, flexible entry-level storage appliance. iXsystems also offers the **TrueNAS Enterprise**, which provides increased bandwidth, IOPS and storage capacity for resource-intensive applications.

Call **1-855-GREP-4-IX**, or go to **www.iXsystems.com**

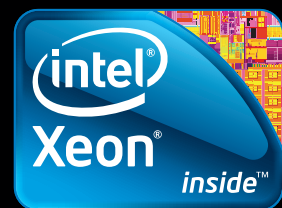
- ✓ Supports iSCSI and NFS exports simultaneously
- ✓ Compatible with popular Virtualization programs such as Hyper-V, VMware, and Xen
- ✓ 128-bit ZFS file system with up to triple parity software RAID

TrueNAS Pro Features

- One Six-Core Intel® Xeon® Processor 5600 Series
- High Performance Write Cache
- Up to 480GB MLC SSD Cache
- Up to 220 TB SATA Capacity
- Quad Gigabit Ethernet
- 48GB ECC Memory

TrueNAS Enterprise Features

- Two Six-Core Intel® Xeon® Processors 5600 Series
- Extreme Performance Write Cache
- Up to 1.2TB High Performance ioMemory
- Up to 500TB SATA or 320TB SAS Capacity
- Dual Ten Gigabit Ethernet
- 96GB ECC Memory





SHAWN POWERS

Sticky Note of Doom

Years ago, I had the brilliant idea that all my users in the finance department should have complex passwords. This made perfect sense to everyone, since dealing with millions of dollars of revenue is something that should be secured. So, the passwords were changed with complexity requirements enforced. I slept better that night knowing our paychecks were no longer secured by passwords like “mustang” or “mrwhiskers”.

I came in the next day only to find very complex passwords written on sticky notes and affixed to everyone’s monitors. Security software is no match for a Sharpie marker and a Post-It. It was a lesson well learned. This month is our Security issue, and although we don’t have an answer to the Sticky Notes of Doom, we do have some great articles on Linux-related security.

Reuven M. Lerner starts off the issue with an interesting column on real-time messaging over the

Web. Back in the days when every user was in a terminal window, a quick `wall` command would send everyone a message. Reuven describes a similar concept, but with Web users. Dave Taylor follows up not with Web programming, but with game programming. Using his talent for making learning fun, Dave shows how to write a script to play *Cribbage*.

Kyle Rankin returns to his PXE magic from a couple years back and explains how to leverage the network bootloader not only to install operating systems, but also to boot them directly. If you’ve ever been intimidated by PXE menus, or if you thought PXE was too limited, you’ll want to read Kyle’s column. It’s a great followup to his last piece on the topic, and it showcases just how flexible PXE can be.

I joined the security bandwagon this issue and decided to talk about passwords. If you (or a “friend”) use the same password for every

Web site, or if you use your pet's name to secure your credit-card statements, you really need to read my column this month. Whether it's to pick up some hints on password creation or just get some pointers for convincing others to use good passwords, I hope you'll find my tips helpful.

If you're fascinated by data encryption, Joe Hendrix's article on Elliptic Curve Cryptography is more than just an interesting read. Joe not only shows how to implement this method, but also how to use it in real life with OpenSSH. With most encryption methods, people just keep making a bigger and bigger encryption key to improve security. Elliptic Curve Cryptography offers more security and smaller key sizes. When it comes to passwords, encryption is great, but even better is to destroy the password completely after using it. Todd A. Jacobs teaches how to configure one-time passwords on your servers. If you're working from an open Wi-Fi hotspot, a one-time password is a way to make sure you're safe even if your password is hijacked.

Speaking of Wi-Fi, Marcin Teodorczyk has a fun article on

setting up a Wi-Fi honeypot. If you want to have fun with your neighbors, or if you're just curious about what people do to an open access point, Marcin shows you what to do. If you live near a place people tend to gather, your results should astound!

We've also got lots of other goodies for you this month. Mike Diehl discusses how to create smartphone apps with Phonegap. Joey Bernard takes a great look at Gnuplot. Our New Products section features a mention of Kyle Rankin's new book, and our Upfront section has useful tips to inform and entertain. So, in honor of the Security issue, maybe take this opportunity to remove sticky notes from monitors and challenge people to change their passwords to something other than their dogs' names. This was a fun issue for us, and we hope the same will be true for you! ■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the [#linuxjournal](https://freenode.net) IRC channel on Freenode.net.

letters



Backblaze?

Regarding Doc Searls' "Heavy Backup Weather" in the October 2012 issue: wait, you use Backblaze? They don't

offer Linux support. I'd rather not see references to services that are not available to me.

—Josh

Doc Searls replies: *You have a good point, and I'm sorry I didn't think about that before I wrote the piece.*

We've wanted to put together some kind of backup that would work for both our Linux and Mac boxes, and for all three of our places (two temporary apartments and one house), as well as on the road. I've used Backblaze for my Mac laptop. I'll drop it.

Meanwhile, recommendations for backing up anything from any OS (starting with Linux), anywhere, over the Net, would be welcome.

Excellent Python Article

I just wanted to send complements and thanks for the excellent article "Python Scripts as a Replacement for Bash Utility Scripts" by Richard Delaney in the November 2012 issue. He nailed it: good background, reasoning and explanations, well-chosen examples and clearly written sample code (the latter two often are disappointments in otherwise good articles about coding).

My only complaint is that I did have to print the article to paper to read it, as my old-geek eyes could not make out the sub-7-point light-blue font used in the e-version.

Python is currently one of my two favorite programming languages. The other is Ruby. As I read Delaney's article, it occurred to me that *LJ* could actually get two-fer-one mileage out of his contribution simply by doing a global search and replace of "Python" with "Ruby", plus a small number of trivial changes to vocabulary and library references—with, of course, sample code in Ruby rather than Python. Voilà! A whole new, second article! (Might I get co-authorship credit for this edit if/when you publish it?)

My point is that there is practically complete functional equivalence between the two languages (pax, fellow aficionados/purists of either!), especially in this particular domain of Bash-replacement utility scripts. Ruby too has a very large and capable library, both standard and gems, for dealing with exactly the same kinds of problems that Delaney covers so well. For example, Ruby also includes a powerful `OptionParser` module (“`optparse`”) for handling command-line switches and arguments easily and generally. I know, I’m developing a large and ever-growing repertory of Ruby utility scripts, many of which are replacements and improvements of older Bash versions—the Ruby versions are much more readable, understandable, maintainable and expandable.

Either is an excellent choice in this domain. Ruby just happens to be my personal favorite, but others may prefer Python. Your mileage may vary, and I’m not trying to start a language flame war here. Interested readers may want to read David Bryant Copeland’s excellent book *Build Awesome Command-Line Applications in Ruby: Control Your Computer, Simplify Your Life* (2012, Pragmatic

Bookshelf), which explores this application domain in detail. He also covers and recommends “`optparse`” (`OptionParser`) to build truly powerful command-line utility apps.

With both Ruby and Python bulwarking good-old Bash, Long Live the Command Line! Thanks again.

—Lorin Ricker

Richard Delaney replies: *Thank you very much for your letter. I think you bring up a brilliant point, and having had little real experience with Ruby, I can only take your word for it. Python and Ruby fill a very similar void in what they hope to achieve and make any “flame wars” between the two communities all the more illogical.*

As you prefer Ruby, my preference is for Python, but only because I program with Python on a daily basis. The reality is that a number of these beautifully crafted languages, such as Python and Ruby, are almost drop-in replacements for one another. The interpreted, dynamic nature of both these languages along with other advantages, such as an REPL, makes their choice as a Bash replacement all the more enticing.

While selecting your language of choice for such things, it's also important not to ignore some of the more modern compiled languages. Slightly more abstract languages like Haskell provide a very elegant way to write programs, as well as the relative newcomer Go. Aspects like program readability, knowledge of the language and documentation often are more important traits for a programmer than the language itself. Thanks again for your kind words.

More Feedback on Richard Delaney's Python Article

I am enjoying reading the November issue and enjoyed the article by Richard Delaney on using Python as a replacement/alternative for writing shell scripts. I wanted to pass on some comments on the same.

The idea resonates with me. Of late, I have resorted to writing Python scripts to perform tasks that usually would be done using shell scripts. My personal reason is that I am not good at Bash scripting and find all the \$s and !s to quite cryptic. However, I feel that whereas that may be the main reason for me, it's actually a good thing, because I can implement

what I have in mind really quickly, as I know Python well. For example, say I need to search across a huge code base for certain strings, and I want the paths of filenames where the strings occur. I used to try doing it using cat, grep and find, but usually the result turned out to be something I didn't like, or I had to search on the Web for solutions. So, I just wrote a Python script to do the job for me (<https://gist.github.com/3759263>). I have been using it ever since, and I am quite happy with it. Bash experts may replace my whole program by a single line, but I am not one of them, and I don't always like to search for solutions for which I know the logic, but am constrained by my lack of Bash scripting knowledge to implement the solution.

Thanks for your article. It was a good read. I would like to point you to docopt (<http://docopt.org>), if you haven't heard of it. It's pretty amazing.

—Amit

Richard Delaney replies: *Thank you very much for your letter. Feedback is always great to hear. I really enjoyed reading your solution to common everyday scripting needs. Often, the*

easiest way to overcome problems is to write from the ground up in Python what you want. It has the advantage of being very expressive, and you don't need to waste time due to unfamiliarity with certain tools or wrangling with Bash's strange syntax.

However, I really would recommend trying to grapple with some of the real GNU utilities when you get time. Tools like `cat`, `head` and `tail` can be very powerful if you spend any amount of time processing files. They allow you to sample parts of files very easily among a host of other things. Often it's easier when you know how to use commands such as these to combine them with a meatier Python script that does the heavy lifting and leaves the simple utilities to the GNU utilities. This has many advantages, such as having decade-tested tools at your disposal, which are the quickest they can be at what they do. The UNIX philosophy of "do one thing well" doesn't translate to every problem, but in this case, I find it is at its most powerful.

PS. Thanks for recommending `docopt`. I will be looking into it for new projects.

More-Advanced Articles?

Is it possible to get more in-depth articles in *Linux Journal*? Everyone that reads this magazine is super smart. I know you don't know what your readers know and what they don't know. So, it makes it hard to go too in-depth into particular articles.

But, from my personal perspective, I see a lot of introduction articles, like intro to Python, or intro to `dbus`, or intro to setting up your router, or intro to KVM or intro to Git.

Sometimes these intro articles seem similar to those from previous years. Like the Git article—I think there was an article in August 2011 that was similar to an article in 2005.

I would like to see some more in-depth articles. Here are some example topics.

Instead of an intro to KVM, what about some hints and tips for setting up KVM with different network setups (NAS, private network), or setting up removable devices (USB stick) or setting it up with multiple monitors? All these things, for me,

seem non-trivial.

Or, what about an entire issue on benchmarking? You could do one article on comparing different distros via benchmarking. You could do another article on comparing virtualization to physical. Or, you could do a project where you show how readers could benchmark their boxes, and readers could send in their results.

Finally, I know you just did an issue on the kernel, but there was no article about where the kernel is going. What about an article on possibilities for kernel development? Are there more and newer additions to the kernel for virtualization? How will the kernel compete with Mountain Lion and Windows 8? What new additions to the kernel will be added for touchscreens?

Overall, I love your magazine. It's great, and I wouldn't be a subscriber if it wasn't. However, this is what I feel is missing.

—Doug

Thank you for your letter. It's only

feedback like this that lets us know the thoughts of our readers. We do try to cover both ends of the spectrum, largely due to feedback like yours and feedback quite opposite: "I want to learn more about Linux, but your magazine seems to exclude all but the seasoned veterans."

We will try to find the proper balance, and if the pendulum has gone the other way a bit, hopefully, it will swing back toward the middle. Seriously though, thank you for the feedback, it's critical!—Ed.

Games

I've read about all the games that are being brought to Linux now that Windows 8 is out. I guess the developers hate how 8 is locked down. How about some articles on the games? I love games.

—Doug

Me too. "Loving" very rarely equates to "is good at", but I try not to let that stop me. I have been keeping up on the Steam client for Linux, and once that's released, perhaps it will spark a renewed interest in gaming. I freely volunteer my

services as resident game-tester!—Ed.

Ignorance in Windows Does Not Make You Knowledgeable in Linux

I have been a UNIX user for more than 30 years. I love UNIX because of the philosophy on which UNIX was designed.

I have noticed that in the Linux community, many people try to show they are knowledgeable by showing they are ignorant in Windows. An example of this is in The Open-Source Classroom column in the November 2012 issue. The author stated that “after I learned to press Ctrl-Alt-Del to log in....” One must have lived on Mars for decades not to know how to log in to Windows. It is theoretically possible but practically very unlikely for someone working in the IT field never to have logged in to Windows.

—John Du

My apologies if my sarcasm was taken as a serious point of confusion. I was trying to make a humorous observation that “Ctrl-Alt-Del” historically has been how to reboot a computer, and

it’s ironically the way to begin computing in modern Windows.

If I’m being completely transparent, however, my new day job does actually mark the first time I’ve ever had to press Ctrl-Alt-Del to log in. I managed one Windows server in my last sysadmin position, but I used pGina for authentication, so I never had to use the three-finger salute to get a login prompt! (Your observation is correct though, I knew perfectly well how to log in in my new Windows environment.)—Shawn.

A Better Shell Script Timeout Method

Regarding Dave Taylor’s column in the November 2012 issue: the following code handles multiple, even overlapping, shell script timeouts reliably. It does not risk killing any unrelated process that just happens to match \$myname. It sends SIGTERM, not SIGALRM.

The following shell routines, taken from the script http://pauljackson.us/watchdog_eg.sh, provide a convenient and robust way for shell script authors to handle multiple,

even overlapping, timeouts in shell scripts:

```
# token = begin_watchdog(secs, func)
#
# Call function 'func' in 'secs' seconds unless
# end_watchdog() called first with 'token'

begin_watchdog()
{
    secs=$1
    func=$2
    dir=$(mktemp -d)
    (
        trap 'rmdir $dir' 1 2 3 15
        sleep $secs
        test -d $dir && $func
    ) 1>/dev/null &

    # the returned token is just the temp directory path
    echo $dir
}

# end_watchdog(token)
#
# Call off the watchdog - rmdir the token directory

end_watchdog()
{
    dir=$1
    rmdir $dir
}
```

```
woof()
{
    echo Woof - Killing process group $$ 1>&2
    kill -TERM -$$ 1>&2
}
```

—Paul Jackson

***Dave Taylor replies:** That's a cool approach. Thanks for sharing it with the Linux Journal readership!*

November 2012 Issue

Thank you all for an excellent November issue. It just sparkled on my tablet, especially all the Python stuff. Richard Delaney's article on how to use Python inside Bash scripts was just what I was looking for. I'd already used Bash inside Python, until discovering I simply was running Bash in Python, so I dropped back to Bash itself. It simply never occurred to me that the other way around was just as valuable.

As for Kyle Rankin's article on the N900 substitute: a very nice try, but personally, I'm still clinging to my N900 until there's a real Linux solution or until I can install a proper native Linux on my transformer—that is, one that

doesn't run on VNC.

Keep up the good work. I for one am very pleased with your digital format; it's getting better and better, and it's a very convenient and cheap solution for us subscribers abroad.

—triantares

Thank you for the kind words. I just purchased a 10" tablet computer after turning in my last one when changing jobs. I was surprised how much I missed the tablet computer, largely due to the beautiful magazine rendering. Much like the new digital billboards popping up all along the highways, I find the digital magazine to grab my attention. I often worry the former might cause car accidents, but I'm quite pleased with the latter.

As to Kyle's N900, well, we've been teasing him ever since Maemo was discontinued. He may have given up his N900, but it will take an army to pry the IBM Model M keyboard from his fingers.—Ed.

WRITE LJ A LETTER We love hearing from our readers. Please send us your comments and feedback via <http://www.linuxjournal.com/contact>.

PHOTO OF THE MONTH

Remember, send your Linux-related photos to ljeditor@linuxjournal.com!

LINUX JOURNAL

At Your Service

SUBSCRIPTIONS: *Linux Journal* is available in a variety of digital formats, including PDF, .epub, .mobi and an on-line digital edition, as well as apps for iOS and Android devices. Renewing your subscription, changing your e-mail address for issue delivery, paying your invoice, viewing your account details or other subscription inquiries can be done instantly on-line: <http://www.linuxjournal.com/subs>. E-mail us at subs@linuxjournal.com or reach us via postal mail at *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Please remember to include your complete name and address when contacting us.

ACCESSING THE DIGITAL ARCHIVE: Your monthly download notifications will have links to the various formats and to the digital archive. To access the digital archive at any time, log in at <http://www.linuxjournal.com/digital>.

LETTERS TO THE EDITOR: We welcome your letters and encourage you to submit them at <http://www.linuxjournal.com/contact> or mail them to *Linux Journal*, PO Box 980985, Houston, TX 77098 USA. Letters may be edited for space and clarity.

WRITING FOR US: We always are looking for contributed articles, tutorials and real-world stories for the magazine. An author's guide, a list of topics and due dates can be found on-line: <http://www.linuxjournal.com/author>.

FREE e-NEWSLETTERS: *Linux Journal* editors publish newsletters on both a weekly and monthly basis. Receive late-breaking news, technical tips and tricks, an inside look at upcoming issues and links to in-depth stories featured on <http://www.linuxjournal.com>. Subscribe for free today: <http://www.linuxjournal.com/emailsletters>.

ADVERTISING: *Linux Journal* is a great resource for readers and advertisers alike. Request a media kit, view our current editorial calendar and advertising due dates, or learn more about other advertising and marketing opportunities by visiting us on-line: <http://www.linuxjournal.com/advertising>. Contact us directly for further information: ads@linuxjournal.com or +1 713-344-1956 ext. 2.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Linus Torvalds recently explained why **ABI** (Application Binary Interface) changes are virtually never acceptable in the kernel. He even said that the existing ABI trumps published standards and specifications. He'd rather keep a non-compliant ABI than fix it if even a single user relied on that ABI.

He listed off what he considered to be the valid reasons to change the ABI. In each case, he said, there was a real external reason justifying the change:

1. Security: in some cases, an ABI has exposed too much information, and that just had to be fixed. Security trumps user convenience.
2. Implementation: in rare corner-cases, an ABI has just been bad. It's made it into the official kernel, but in a nightmare-Frankenstein way that really didn't work right. Apparently, a bad enough breakage could justify a fix.

3. Applications: in some cases, an application should have been portable to Linux, but broke because of the ABI. A fix in that case was warranted.

And aside from the security case, Linus felt that it made more sense to maintain a compatibility layer that included the old behavior, just in case anything really did depend on the old ABI. So it would be cruft, but it would be our cruft, and we'd protect it.

Linus closed his e-mail/rant by saying that anyone who intentionally wanted to change an ABI for some reason other than the ones listed above, or because a change crept in accidentally and needed to be reverted, should not work on the kernel or on library development either—harsh.

Dan Luedtke has produced **LanyFS**, a new minimalist Flash filesystem. The idea is that it would work with anything—even **Arduino**—and allow quick, easy transportation of files between

hardware, without a lot of messing about with file ownership, special permissions or anything like that. The assumption would be that the user had full ownership of everything, which, for regular users, is almost always the case.

There was a lot of initial resistance, because **FAT32** traditionally has been the filesystem used on such drives. But FAT32 has a file size limit, and Dan had hit that limit with an Arduino project and wrote LanyFS partly to overcome that.

Meanwhile, **exFAT** is **Microsoft's** solution to the same use case. The only problem is, as **Carlos Alberto Lopez Perez** put it, exFAT was encumbered by patents and licensing fees, which put it out of the running. But, Microsoft might be reluctant to support LanyFS, if it's pushing its own exFAT solution onto users.

As it turns out, **Arnd Bergmann** also is working on a minimalist Flash filesystem. Although because he was working with a third-party vendor and didn't want to mess up the time frame, he didn't offer any details, except to say that the filesystem would be optimized

for Flash.

The appealingly named **Unified Extensible Firmware Interface** (UEFI) probably will be supported in the Linux kernel at some point. Its purpose is to prevent users from having control of their own systems, so that third-party vendors can run the show. Look it up. It's ugly.

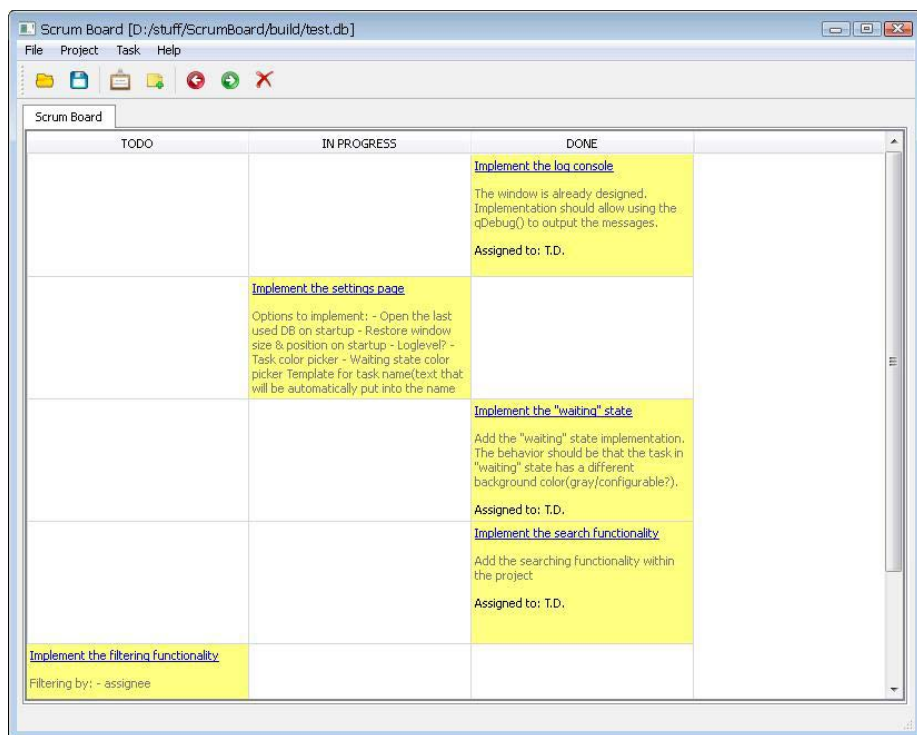
A variety of technical problems are standing in the way of UEFI support. **Matthew Garrett** recently posted some patches to prevent the root user from modifying the kernel. But, **Alan Cox** didn't think the kernel alone could guard against the root user successfully. After all, the foundation of Linux security rests on the idea that once anyone gets to be root, that's the ballgame, and so it's more important to try to prevent that from happening, than to try to interfere with root's actions once it did.

I'd say UEFI almost certainly will be part of the kernel. But at the same time, it wouldn't be enabled by default on any system other than those third-party systems being distributed specifically in order to have UEFI enabled. Hopefully, it won't go too far beyond that.

—**ZACK BROWN**

Non-Linux FOSS

If you're a developer, you've no doubt heard of the SCRUM project development method. In fact, it's very likely you use it. If you're tired of sticky notes on a whiteboard, however, you might want to check out ScrumBoard.



ScrumBoard is an open-source Windows application that provides a digital whiteboard for manipulating project tasks. Tasks can be created, moved and reassigned all inside a completely digital environment. Some people benefit from a physical, visual look at project tasks, but for others, the convenience of a portable digital version is perfect. If you're a SCRUM user, and you're stuck on Windows, check out ScrumBoard: <http://scrum-board.sourceforge.net>.

—SHAWN POWERS

They Said It

I find television very educating. Every time somebody turns on the set, I go into the other room and read a book.

—Groucho Marx

If everyone demanded peace instead of another television set, then there'd be peace.

—John Lennon

Television is not real life. In real life people actually have to leave the coffee shop and go to jobs.

—Bill Gates

Watching television is like taking black spray paint to your third eye.

—Bill Hicks

If it weren't for Philo T. Farnsworth, inventor of television, we'd still be eating frozen radio dinners.

—Johnny Carson

1&1 DEDICATED SERVERS

1&1 25 YEARS

1&1 is celebrating its 25th anniversary. Over the past 25 years 1&1 has grown to become one of the world's leading web hosts. Today, with 12 million customer contracts and 5000 employees 1&1 provides superior web hosting and server solutions to support your business. In celebration here is a gift from us to you.



Our data centers offer top security, Cisco firewall protection and maximum uptime.

- ✓ Unlimited traffic with no extra cost
- ✓ Parallels Plesk® Panel 11 for unlimited domains
- ✓ Exclusive to 1&1: Optional SUSE Linux Enterprise Server
- ✓ Mobile Server Monitoring for Android/iOS devices



Save \$360

Server XL 6

AMD Hexa-Core

6 Cores x 2,8 GHz
(3.3 GHz Turbo Core)

16 GB RAM DDR3 ECC

1,000 GB (2 x 1,000 SATA)

Software RAID 1

Free choice of CentOS, Debian, Ubuntu, or openSUSE.

Unlimited Traffic (100 Mbit/s)

~~\$129.99~~ **\$99.99** per month*

1&1®

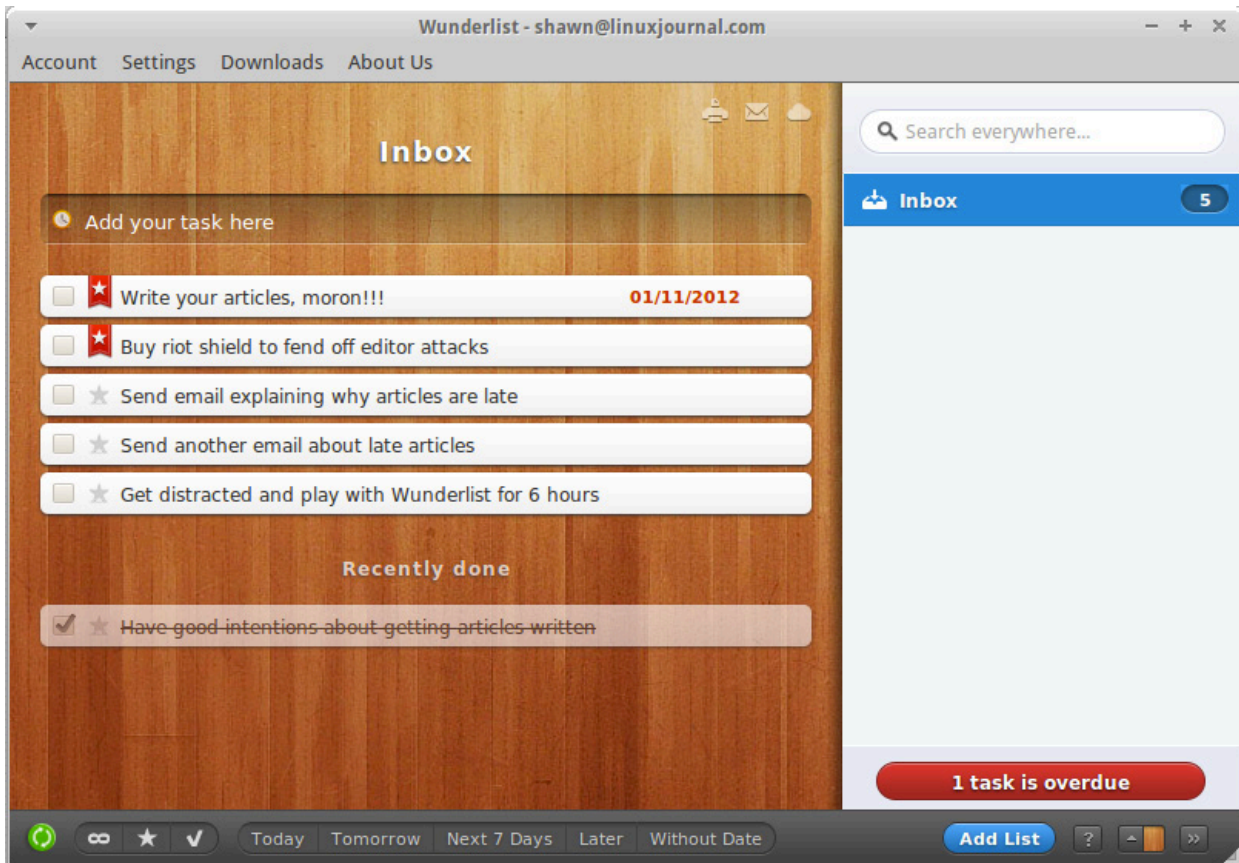


1and1.com

* Offer valid for a limited time only. First year save \$360 off regular price of Dedicated Server XL 6. Other terms and conditions may apply. Visit www.1and1.com for full promotional offer details. Program and pricing specifications and availability subject to change without notice. 1&1 and the 1&1 logo are trademarks of 1&1 Internet, all other trademarks are the property of their respective owners. © 2013 1&1 Internet. All rights reserved.

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

Wunderlist



I'm often compared to the Absent-Minded Professor. I take it as a great compliment, because in the movie, he's brilliant. Unfortunately, when people refer to me as him, it's the "absent-minded" part they're stressing—not the "professor" part.

During the past few years, I've written about task-management systems, "get things done" digital tools and ways to keep track of to-do lists in Linux. This month, I'm sharing Wunderlist, which is a cross-platform task-management and sharing utility that is truly amazing. When I say cross-platform, I really mean

it too. Wunderlist works in Windows, OS X, Linux, iOS, Android, Blackberry, the Web and probably another half-dozen interfaces I've yet to encounter.

Although it has a robust feature set including task list sharing, due dates, task notes, the ability to drag tasks between lists and keep track of completed items, for me, its real value is in its simplicity. Wunderlist doesn't try to do too much; it just does task lists really, really well. If you haven't seen Wunderlist in action before, put it on your list today: <http://www.wunderlist.com>.

—SHAWN POWERS

LINUX JOURNAL ARCHIVE DVD



NOW AVAILABLE

Save \$10.00 by using discount code DVDFEB at checkout.

Coupon code expires 2/16/2013

www.linuxjournal.com/dvd

Gnuplot—the Grandfather of Graphing Utilities

In these columns, I have covered several different scientific packages for doing calculations in many different areas of research. I also have looked at various packages that handle graphical representation of these calculations. But, one package that I've never looked at before is gnuplot (<http://www.gnuplot.info>). Gnuplot has been around since the mid-1980s, making it one of the oldest graphical plotting programs around. Because it has been around so long, it's been ported to most of the operating systems that you might conceivably use. This month, I take a look at the basics of gnuplot and show different ways to use it.

Gnuplot is a command-line-driven program. As such, it has been co-opted to provide graphic capabilities in several other applications, such as octave. Thus, you may have used gnuplot without even realizing you were doing so. You can use gnuplot in several ways. It not only can accept input data to plot, but it also can plot functions. Gnuplot can send its output either to

the screen (in both a static file format display or an interactive display), or it can send output to any of a large number of file formats. Additionally, lots of functions are available to customize your plots, changing the labels and axes, among other things.

Let's start by installing gnuplot. Binaries are available for many different operating systems. Most Linux distributions also should come with a package for gnuplot, so installation should be a breeze. If you want the latest and greatest features available, you always can download the source code and build gnuplot from scratch.

Once gnuplot is installed, you can start it by executing the command `gnuplot`. When executed this way, you are launched into an interactive session. Let's start by trying to plot a basic function. You should be able to plot any mathematical function that would be accepted in C, FORTRAN or BASIC. These mathematical expressions can be built up from built-in functions like `abs(x)`, `cos(x)`

or Bessel. You can use integer, real and complex data types as arguments to these functions.

When using gnuplot to generate a plot, you either can have all of the commands in a single file and hand them in to gnuplot as a script, or you can start gnuplot up in interactive mode and issue these commands one at a time in the command environment. To run a gnuplot script, you simply need to add it at the end of the command when you run gnuplot—for example:

```
gnuplot script_to_run
```

When you run gnuplot in interactive mode, you can quit your session with the command `quit`. The two most basic commands are `plot` and `splot`. `plot` generates two-dimensional plots, and `splot` generates three-dimensional plots. To plot a simple function, you can use:

```
plot sin(x)/x
```

This generates a plot window, displaying the graphical results (Figure 1). If you want to add a title to the plot, you can add this option

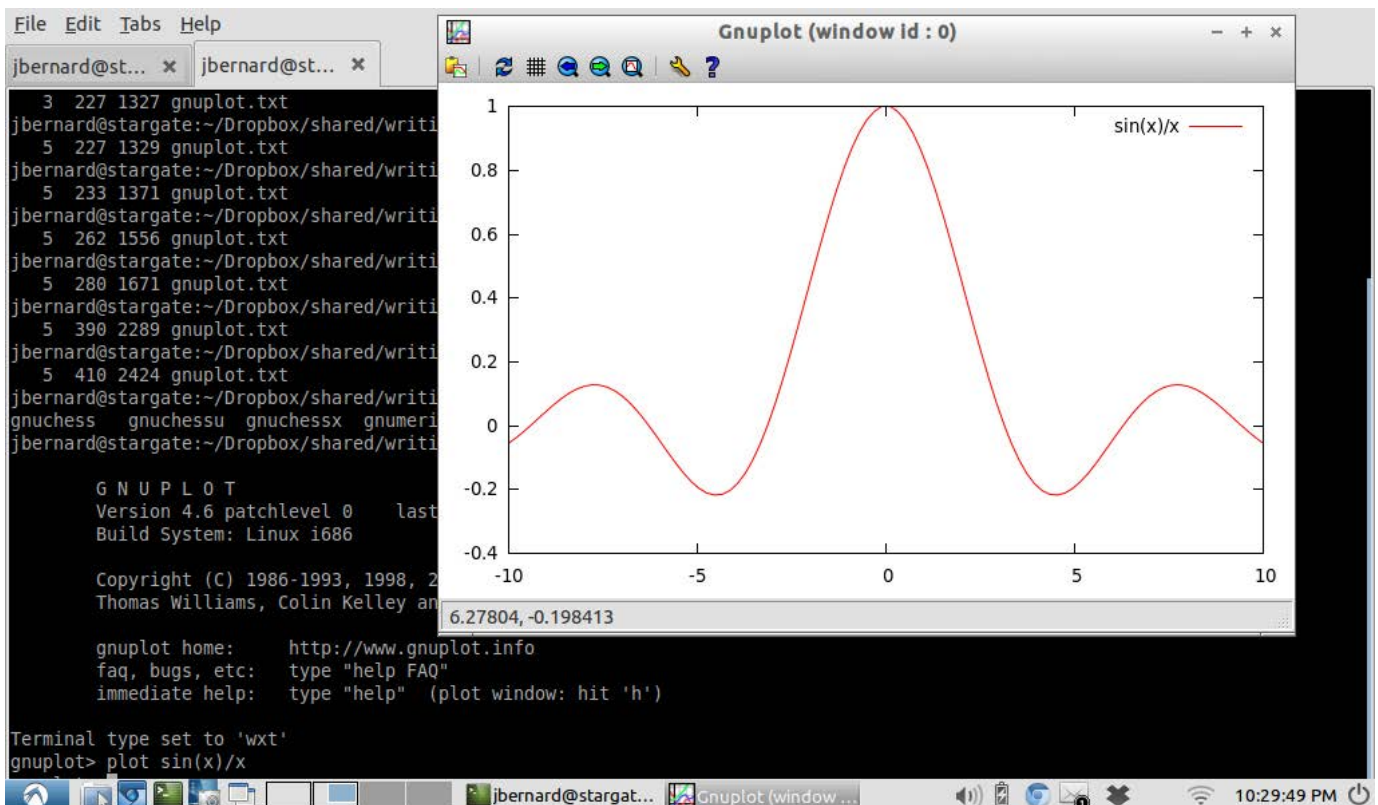


Figure 1. Plotting commands open a new window for display.

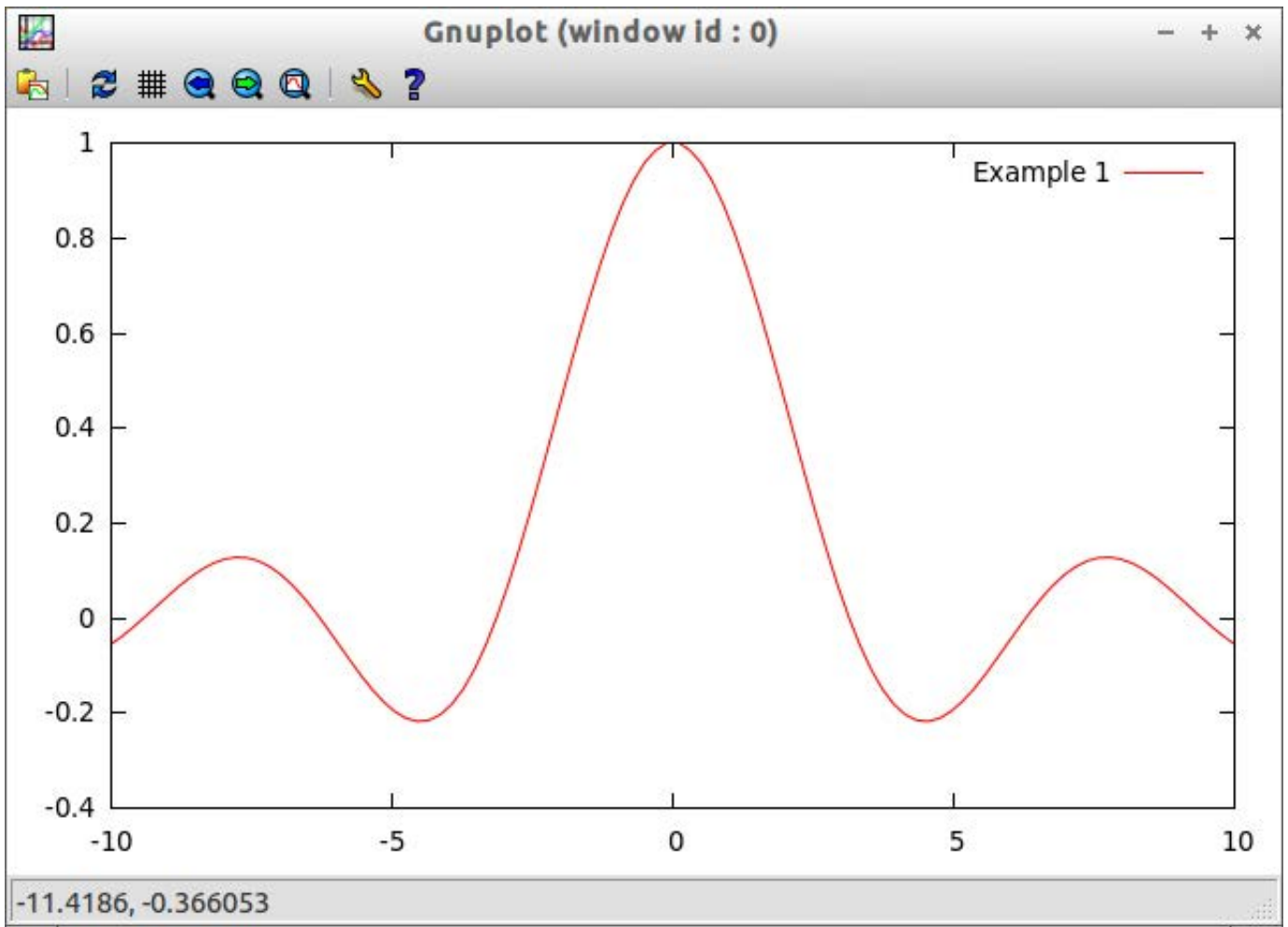


Figure 2. A Basic Plot of $\sin(x)/x$

to the plot command:

```
plot sin(x)/x title "Example 1"
```

You even can plot multiple expressions on the same plot window with:

```
plot sin(x)/x title "Example 1", sin(x) title "Example 2"
```

To plot a three-dimensional graph, simply hand in an expression with two

independent variables to `splot`, such as:

```
splot x**2+y**2
```

If you run into a problem, the first place to look is the built-in help function. To get help with the `plot` command, execute the command:

```
help plot
```

This pulls up the help documentation

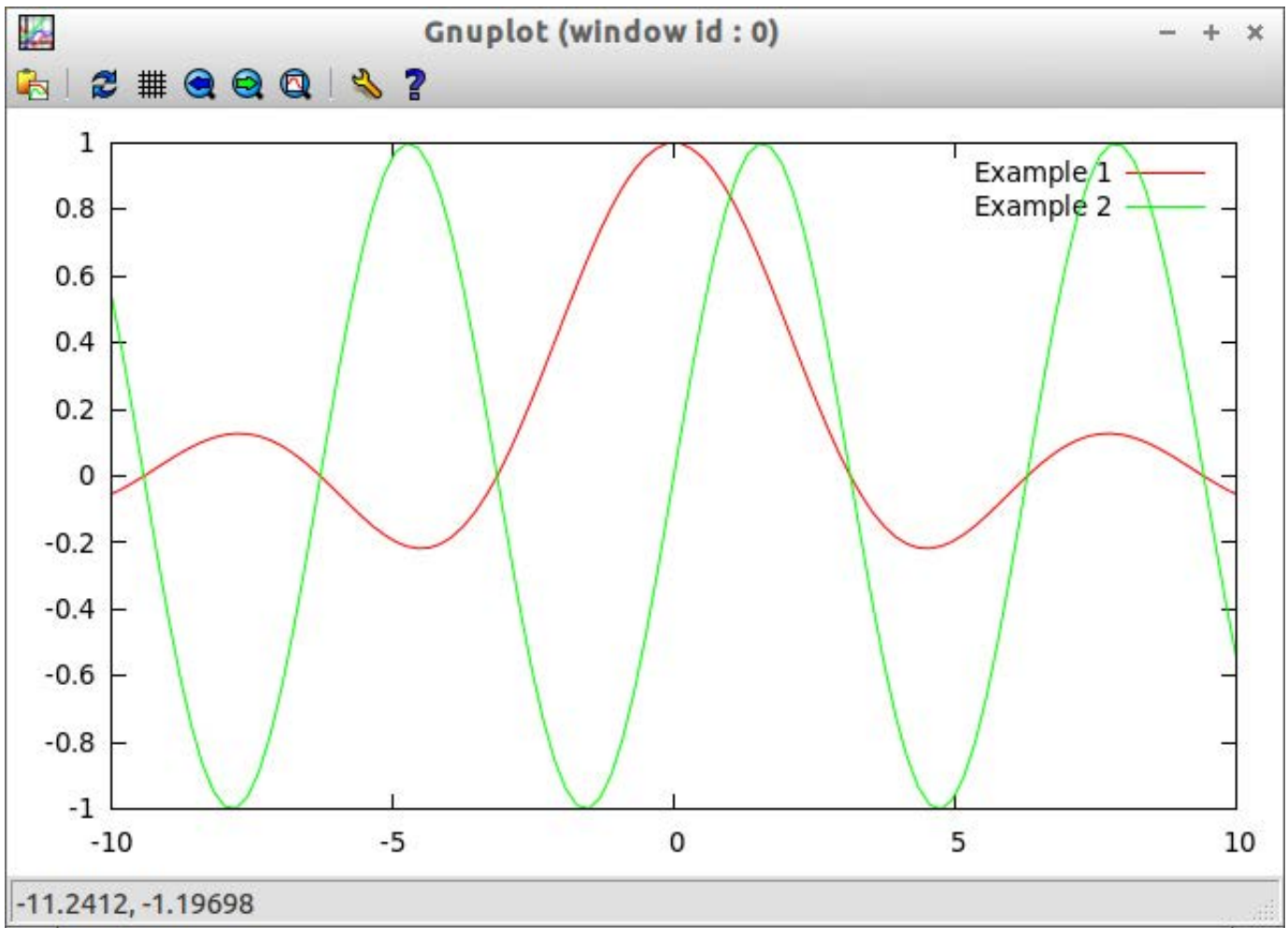


Figure 3. You can plot multiple functions on the same graph.

that gnuplot has regarding the plot command.

This is fine if you are just trying to see what some expression looks like when it is plotted out, but in real science, you often collect data in experiments that need to be plotted so you can do some graphical analysis and get ideas as to what may be happening. Gnuplot can handle this type of plotting too. To do so, you simply

need to hand in the filename of the file containing the data to be plotted. This file should have the data elements arranged in columns, where the columns are separated by white space of some kind. Any lines that start with # are treated as comments by gnuplot and are ignored. If your data file contains several data columns, you can select which columns are pulled in to be plotted as options to the plot or

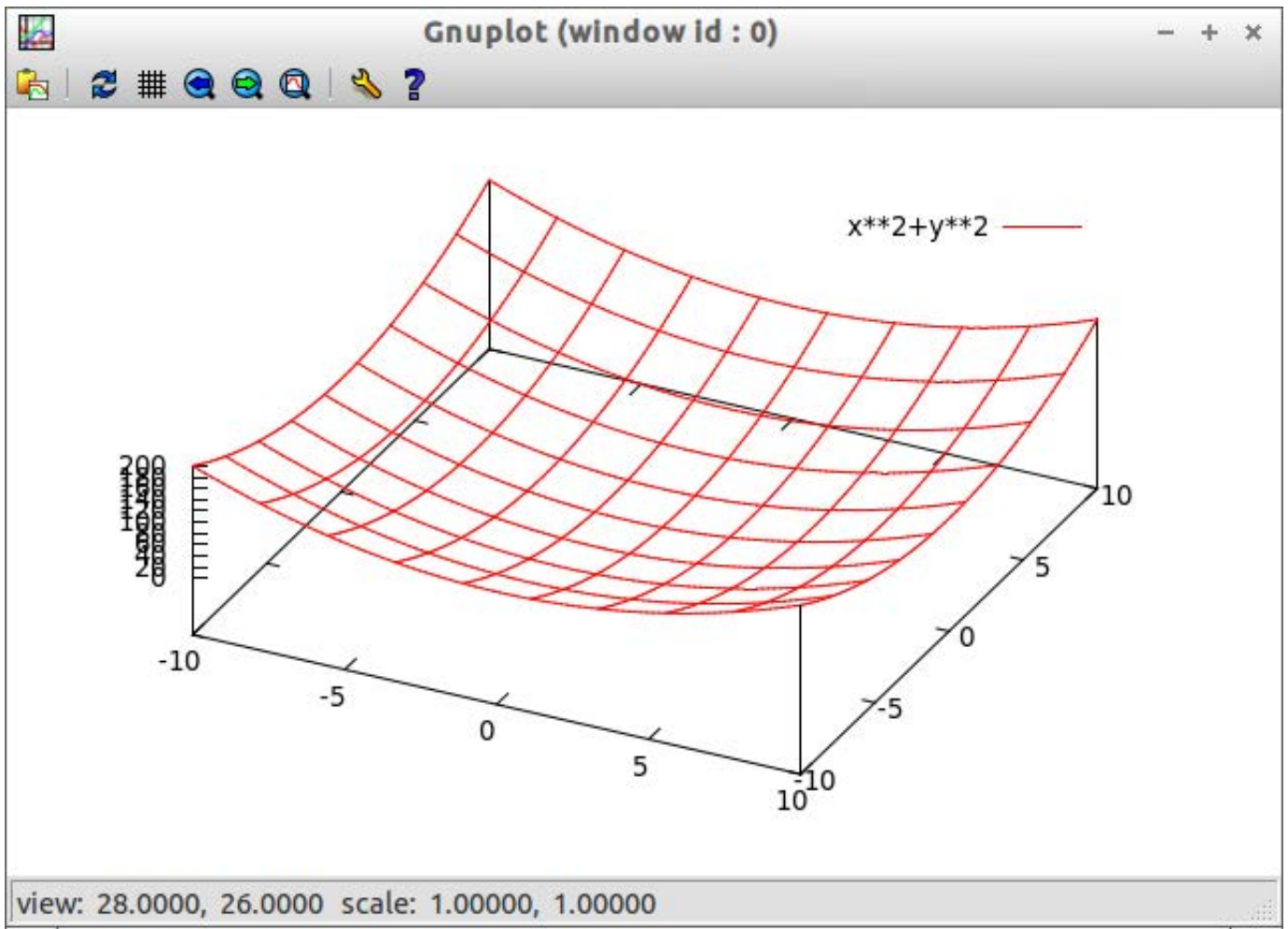


Figure 4. Gnuplot even can handle 3-D plots.

plot functions. As an example, say you have a data file that has the temperature and pressure for each day. You can plot the temperature with:

```
plot "weather.dat" using 1:2 title "Temperature"
```

If you want to get the pressure graph, you would use:

```
plot "weather.dat" using 1:3 title "Pressure"
```

If you want to plot all three columns, you can use:

```
splot "weather.dat"
```

There are two ways of customizing your plots when using gnuplot. The first is to use options to the `plot` and `splot` commands. In this case, you define things like the title of the plot, the axes or the style. The styles available can

be lines, points, linespoints, impulses, dots, steps, fsteps, histeps, errorbars, xerrorbars, yerrorbars or xyerrorbars. To use one of the styles, you can include the option with the `with` keyword. So, if you want to plot both the lines and points of your graph, you could add `with linespoints` to your `plot` command. You also can use shortcuts for these options. For `with`, you can use `w`. For the `title` option, you can use `t`. For the `using` option shown earlier, you can use `u`.

The second option for customizing your plots is to use the `set` command. With this command, you are free to set the values for several graphing options. Using the second option, you can set all types of options, like the title, xlabel, xrange, xticks or key, among other options. For example, you can set the y-range with:

```
set yrange [20:500]
```

After setting the various plotting options, you need to tell gnuplot to redraw the plot you are working on. You can do this with the command:

```
replot
```

Many of these set options also use shortcuts. For example, the shortcut version of the above command is:


```
set yr [20:500]
```

Gnuplot is not only a capable utility to plot data and functions, but it also can do some analysis on the data being plotted. For example, you can get gnuplot to do curve fitting on the data. To do so,

Embedded Server

Standard SIB

- Fanless x86 1GHz CPU
- 1 GB DDR2 RAM On Board
- 4 GB Compact Flash Disk
- 10/100/1000 Base-T Ethernet
- Two RS-232 Ports
- Four USB 2.0 Ports
- Mini-PCIe
- Audio In / Out
- Power Supply Included
- Analog SVGA 3D Video
- VESA Hole Pattern
- Optional Wireless LAN
- Locked Compact Flash Access
- No Moving Parts
- XPE or Linux with Eclipse IDE
- Dimensions: 4.9" x 4.7" x 1.7" (125 x 120 x 44mm)



2.6 KERNEL

The EMAC Server-In-a-Box (SIB) is a low cost, small footprint, yet powerful server. Like all EMAC SIBs, the Standard SIB has no moving parts and features a rugged enclosure design making it an ideal choice for most industrial applications. The Standard SIB has a secure locking cover for securing the flash media, while still offering easy removal for updates and backing up the system.

http://www.emacinc.com/servers/Standard_sib.htm

Since 1985


OVER

28

YEARS OF

SINGLE BOARD

SOLUTIONS



EMAC, inc.

EQUIPMENT MONITOR AND CONTROL

Phone: (618) 529-4525 • Fax: (618) 457-0110 • Web: www.emacinc.com

[UPFRONT]

you first need to define a function, as well as some initial guesses before calling the `fit` command. An example would look like this:

```
f1=a1*tanh(x/b1)
a1=300; b1=0.005;
fit f1(x) 'data_file.dat' using 1:2 via a1,b1
```

This tells gnuplot to try to fit the data from the columns 1 and 2 from the file `data_file.dat` to the function defined by `f1(x)`.

When you have an environment created for a particular research area, you can save all of the settings you may have set up with the command `save`. This command essentially saves off all of the gnuplot commands you issued to the text file. This text file can be loaded into a new gnuplot session with the `load` command. This will take all of the commands saved to the “save” file and re-run them in the new session.

You always can see what options have been set by using the command `show`. This command shows you what values have been set within the current session. To see all of the options, use the command `show all`. When you are playing with options, you sometimes can get yourself into an

odd condition. Just remember that you always can reset any values created with the `set` by using the `reset` command. This command resets these session options to their default values.

Sometimes you may need to interact with the system on which gnuplot is running. In those cases, you need to start a shell session from gnuplot. There are two ways to do so. The first is to use the command `system`. In this case, you can hand in a string containing the system commands that need to be run outside of gnuplot. The other option is to use the command `!`. This command actually is just a shortcut for the command `system`, and the commands can be used interchangeably.

This article has covered only the most basic functions available in gnuplot. It's definitely worth your time to look deeper into the documentation to see what else it can do for you in analyzing your data. Even if you don't use gnuplot directly, learning more about it will help you when you use other applications like octave. Take this article as a jumping-off point and explore just what is possible in data analysis.

—JOEY BERNARD

Native(ish) Netflix!

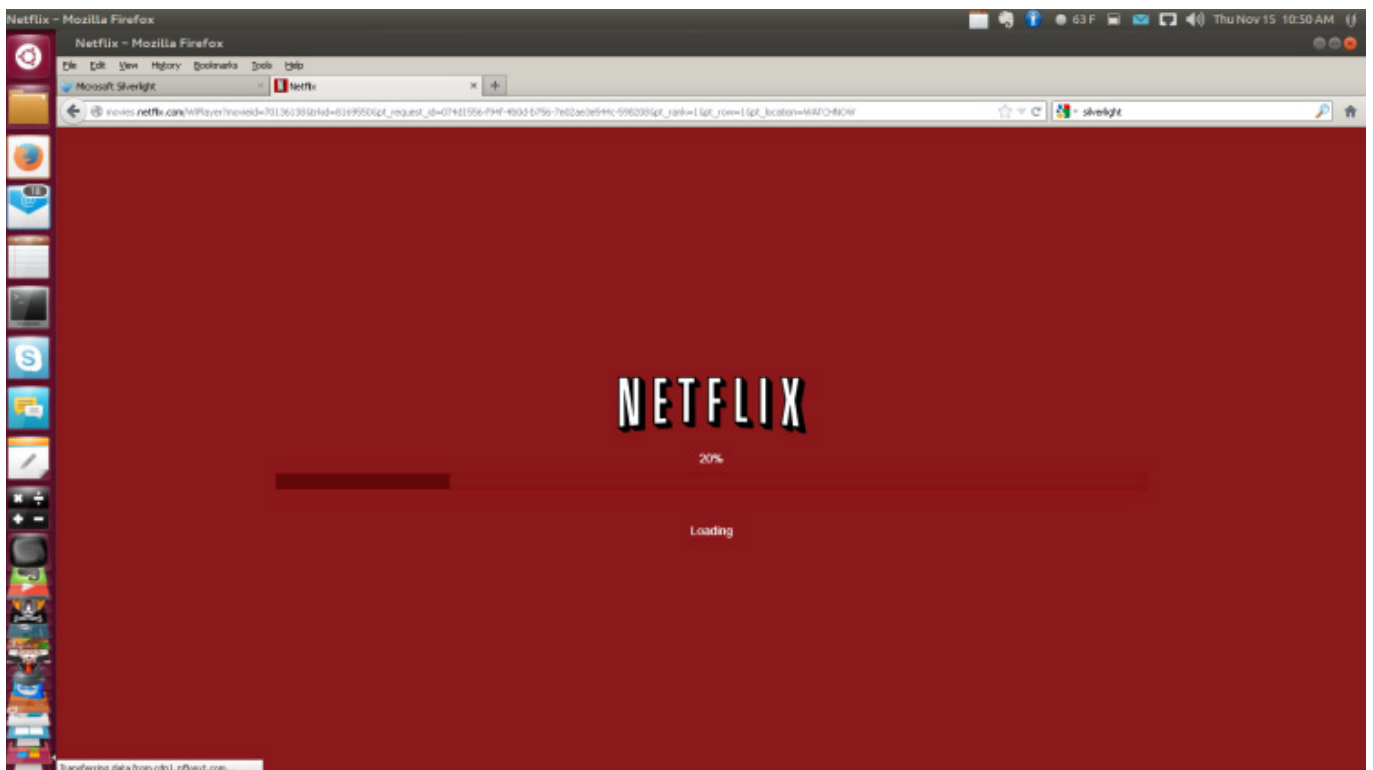
The folks over at <http://www.iheartubuntu.com> recently put up a challenge to the Linux community to get Netflix to work natively under our beloved OS. Thankfully, Erich Hoover stepped up to the challenge and patched the Wine Project in a way to allow Firefox/Silverlight to be installed and actually work with Netflix's DRM'd Silverlight!

The process is a little complex, and it involves patching source code before compiling, but Erich plans to

create a PPA with all the compiling already done. Eventually, he intends to create a standalone Netflix-playing app that incorporates all the pieces of Wine and Silverlight. Thankfully, Erich didn't wait until the project was complete before sharing his success. If you want to play native(ish) Netflix on your Linux desktop without virtualization, check out his instructions at:

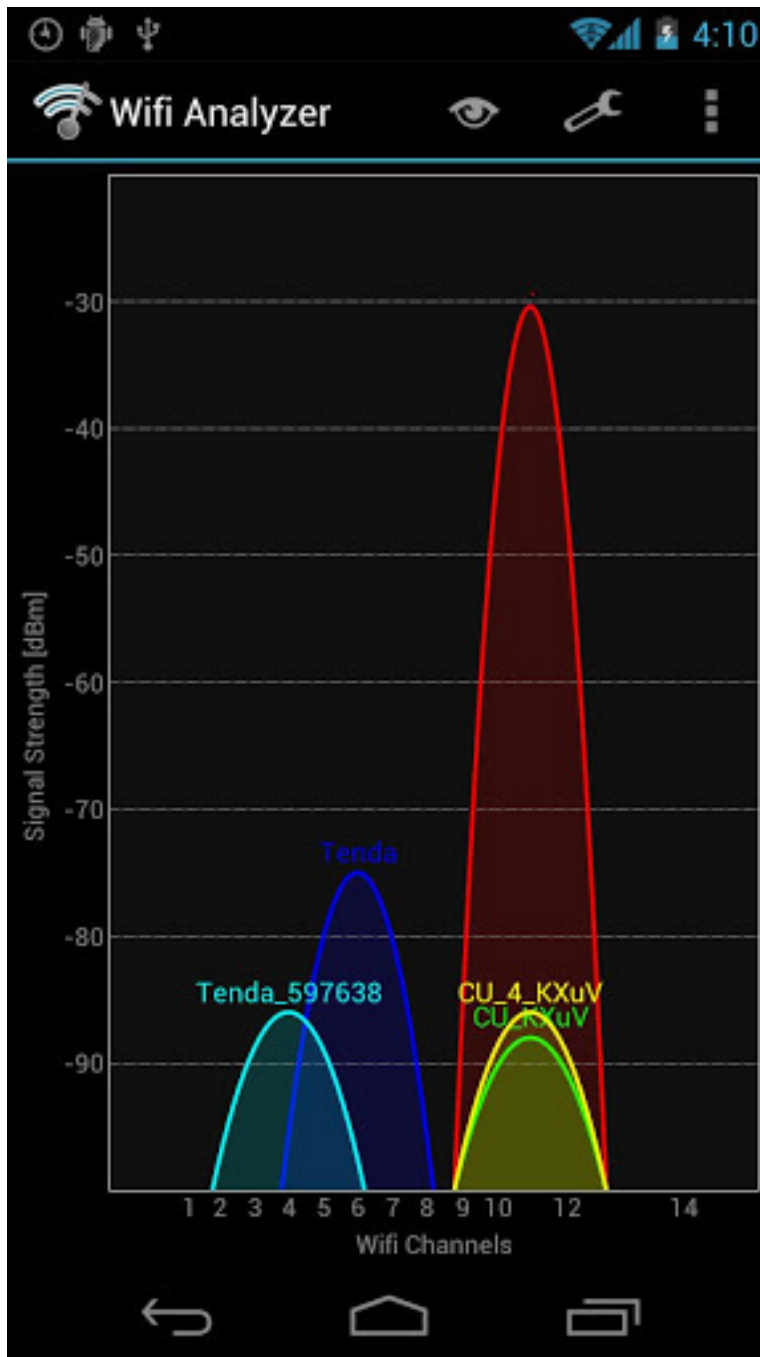
<http://www.iheartubuntu.com/2012/11/netflix-on-ubuntu-is-here.html>.

—SHAWN POWERS



Screenshot from <http://www.iheartubuntu.com>

Android Candy: WiFi Analyzer



Screenshot from the Google Play Store

I have a new day job, and as part of the hiring package, I was issued a smartphone. I'm a little bitter that it doesn't include a tethering plan, but that doesn't upset me nearly as much as the lack of Wi-Fi analysis apps. See, my new job issued me an iPhone. I really like the iPhone (it's true, I can't lie), but in order to scan Wi-Fi, I'd have to jailbreak my phone!

Thankfully, the world of Android has no such silly limitation. If you've ever wanted to scan for access points or check signal levels in different parts of your house, WiFi Analyzer is an excellent (and free!) tool. WiFi Analyzer will show currently available access points, graph signal strength as you walk around the house, and even give information on signal quality.

I've used WiFi Analyzer to determine the best

placement of access points when deploying a building-wide wireless infrastructure. I've used it to pick the best channel for my home access points. I've even walked down the road with it to see what my neighbors use as SSIDs. (That last one might be a little creepy, but really, if people name

their wireless networks after Teletubbies, you want to keep an eye on them.)

Because it's incredibly useful, completely free and not available on iOS, WiFi Analyzer gets this month's Editors' Choice award. Check it out at <http://a.farproc.com/wifi-analyzer>.

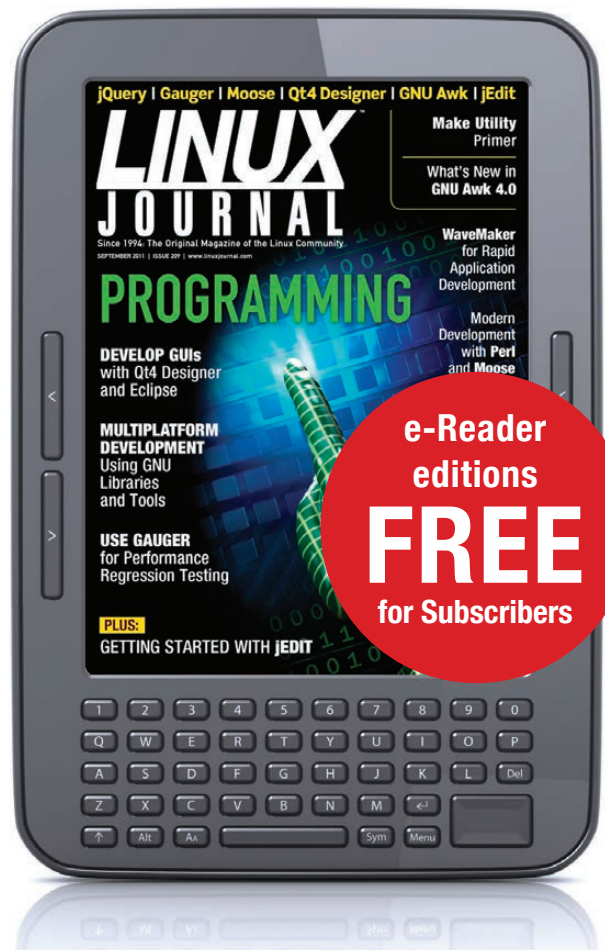
—SHAWN POWERS

LINUX JOURNAL

on your
e-Reader

Customized
Kindle and Nook
editions
now available

LEARN MORE





REUVEN M.
LERNER

Real-Time Messaging

Want to send messages to all the browsers connected to your site? The pub-sub paradigm, run through Web sockets, might be just the solution.

Back in the 1980s, BSD UNIX introduced the idea of a “socket”, a data structure that functioned similarly to a file handle, in that you could read from it or write to it. But, whereas a file handle allows a program to work with a file, a socket is connected to another process—perhaps on the same computer, but quite possibly running on another one, somewhere else on the Internet. Sockets brought about a communications revolution, in no small part because they made it easy to write programs that communicated across the network.

Today, we take that for granted. Dozens or hundreds of sockets are open on my computer at any given moment, and I don’t know if they’re communicating with local or remote programs. But, that’s just the point—it’s so easy to work with sockets, we no longer think of networked programs as anything special or unusual. The people

who created sockets couldn’t possibly have imagined the wide variety of protocols, applications and businesses that were built using their invention.

My point is not to praise sockets, but to point out that the inventors of a technology, particularly one that provides infrastructural support and a new abstraction layer, cannot know in advance how it’ll be used.

In that light, consider a new network communication protocol called Web sockets, part of the standards known collectively as HTML5. To me, at least, Web sockets are the most undersold, least discussed parts of the HTML5 suite, with the potential to transform Web browsers into a fully fledged application platform.

Web sockets don’t replace HTTP. Rather, much like BSD sockets, they provide bidirectional, long-term communication between two computers. The “bidirectional” and “long-term”

aspects distinguish Web sockets from HTTP, in which the client sends a request, the server sends a response, and then the connection is terminated. Setting up a Web socket has very little overhead—and once communication is established, it can continue indefinitely.

Now that Web sockets exist, and are even supported by a growing number of browsers, what can you do with them? That question is still hard to answer, in no small part because Web sockets are so new. After all, if you had asked someone in the 1980s what you could do with BSD sockets, it's unlikely that streaming video would have come to mind.

That said, there are some applications for which Web sockets are already showing their advantage. In particular, applications that benefit from real-time data updates, such as stock-market tickers, now can receive a steady stream of data, rather than perform repeated Ajax calls to a server. Real-time chat systems are another example of where Web sockets can shine, and where HTTP has not performed adequately. Indeed, any Web application that handles or displays a constant flow of data can benefit from Web sockets.

But you can go even farther than that. Remember, Web sockets provide communication between a single server

and a single client. There are, however, numerous applications in which the server might want to “broadcast” information to a large number of clients simultaneously. You can imagine how this could work with Web sockets, creating a Web socket connection between a server and each of the clients, and then sending messages to each of the clients, perhaps by iterating over the array of Web sockets and invoking `send()` on each one.

This is certainly possible, but implementing such a system yourself would be time-consuming and difficult, and might not scale easily. Fortunately, now there are third-party services that (for a fee) will handle such connections for you. Such publish-subscribe (“pub-sub”) systems make it possible for a server to send to any number of clients almost simultaneously, opening the door to all sorts of Web applications.

In this article, I review the basics behind Web sockets and then move forward to demonstrate a simple application that uses the pub-sub paradigm. Even if you don't currently need this sort of functionality in your Web application, I have no doubt you'll eventually encounter a situation that can benefit from it. When the time comes, you'll hopefully realize that it's not too difficult to put it into place.

Working with Web Sockets

Web sockets, as with the rest of the HTML5 standard, have to do with programming within the browser—which, of course, happens in JavaScript or a language that compiles into JavaScript. To create a new Web socket, you simply say:

```
var ws = new WebSocket("ws://lerner.co.il/socket");
```

The beauty of this API is its simplicity. I don't know about you, but I'm tired of protocols that expect me to remember which parameter represents the hostname, which the protocol and which the port (if any). In the case of Web sockets, as you would expect from a Web standard, you pass all of that along in a URL whose protocol is defined as "ws" or "wss" (for SSL-encrypted Web sockets). Also notice that you don't have to define the Web socket as being read-only, write-only or read/write; Web sockets are all bidirectional.

You can send data to the other side of your Web socket by invoking the "send" method:

```
ws.send("Hello");
```

Or, if you want to send something a bit more complicated, it's typical to use JSON:

```
var stuff_to_send = {a:1, b:2};  
ws.send(JSON.stringify(stuff_to_send));
```

What happens when your Web socket receives some data? Nothing just yet. You have to tell the browser that each time it receives data, it should do something with that data, such as display it on the screen. You tell it what to do with a callback, as you might expect in a functional language, such as JavaScript. That is, you tell the Web socket that when it receives data, it should execute a function, passing the received data as a parameter to that function. A simple example might be:

```
ws.onmessage = function(message) {  
    alert("Received ws message: '" + message.data + "'");  
};
```

You also can do something more interesting and exciting with the data:

```
ws.onmessage = function(message) {  
    $("#wsdata").html(message.data);  
};
```

Of course, the incoming data isn't necessarily a string. Instead, it might be a bunch of JSON, in which case, it might contain a JavaScript object with fields. In such a case, you could say:

```
ws.onmessage = function(message) {
```

```
parsed_message = JSON.parse(message)
$("#one").html(parsed_message.one);
$("#one").html(parsed_message.two);
};
```

Now, it's important to remember that the Web sockets protocol is a distinct protocol from HTTP. This means that when I say I want to connect to `ws://lerner.co.il/socket`, I need to be sure I'm running a Web socket server on `lerner.co.il` that responds to items at that URL. This is not the same thing as Apache, nginx or whatever your favorite HTTP server is.

So, when I say here that your browser connects to a server, you need to provide such a server. The Resources section of this article describes a number of systems that make it possible and fairly straightforward to create a server for Web sockets.

Pub-Sub

As you can see, working with Web sockets is fairly straightforward. But, what happens if you want to send messages to multiple clients? For example, let's say your company deals with stocks, and you want the home page of your company's Web site to show the latest value of certain stocks and stock indexes, updated continuously.

The simplest and seemingly most

straightforward way is to use the strategy I described above—namely, that the server can store its Web sockets in an array (or similar data structure). At a set interval, the server then can execute `ws.send()` to each of the clients, either sending a simple text string or a JSON data structure with one or more pieces of information. The client, upon receiving this data, then executes the `onmessage` callback function, which then updates the user's browser accordingly.

This approach has a number of problems, but the main one that bothers me is the lack of a real abstraction layer. As application developers, you want to send the message, rather than consider how the message is being sent, or even who is receiving it. This is one way of looking at the publish-subscribe (pub-sub) design pattern. The publisher and subscriber aren't connected to each other directly, but rather through a middleman object or server that takes care of the connections. When the publisher wants to send a message, it does so through the broker, which then uses the existing Web socket connection to send a message to each client.

Now, this might sound something like a message queue, which I described about a year ago in this space. But message queues and pub-sub systems work quite differently

from each other, and they are used for different purposes.

You can think of a message queue as working something like e-mail, with a single sender and a single recipient. Until the recipient retrieves the message, it waits in the message queue. The order and timing in which messages appear isn't necessarily guaranteed, but the delivery of the message is.

By contrast, a pub-sub system is something like a group IM chat. Everyone who is connected to the pub-sub system and is subscribing to a particular channel, receives the messages sent to that channel. If you happen not to be subscribed at the time a message is sent, you won't receive it; there is no mechanism in pub-sub for storing or replaying messages.

If you were interested in giving people real-time stock updates from your home page, the pub-sub way of doing that would be to have each client register itself as a subscriber with the pub-sub server. The server then would send new messages at regular intervals to the pub-sub server, which would pass them along to the appropriate clients. Remember, in a pub-sub system, the publisher doesn't know how many subscribers there are or when they enter/leave the system. All the publisher needs to know is the API for sending data to the pub-sub server, which passes

things along to the appropriate clients.

Implementing Pub-Sub

Pub-sub has long existed outside the Web and is a fairly standard architecture for "broadcasting" information to a variety of clients. And, you can create a pub-sub system on your own, but there are at least two commercial services—Pusher and PubNub are the best known—that make it trivially easy to implement real-time messaging within your Web application. Pusher uses Web sockets, substituting a Flash-based solution when a browser doesn't support them. PubNub uses a different system, known as "HTTP long polling", which avoids the problem of browser support for Web sockets. Both are worth consideration if you're looking for a commercial pub-sub service, but I use Pusher here (as well as in my own consulting work), partly because I prefer to use Web sockets, and partly because Pusher lets you tag each message with an event type, giving you a richer mechanism for sending data.

Because Pusher is a commercial service, you need to register with it before you can use it. It has a free "sandbox" system that is more than viable for systems in development. Once you go beyond its limits of 20 connections and 100,000 messages

per day, you need to pay a monthly fee. After signing up with Pusher, you need three pieces of information to implement the application:

- A key: this is the equivalent of your user name. It's used by the publisher to send messages and by the client to retrieve messages. This cannot be a secret, because it'll be in the HTML files that your users display in their browsers.
- A secret: the equivalent of a password. This will not be used on the client, or subscriber, side of things. But, it will be used on the server (publisher) side, which is sending data to the client. This ensures that only you are sending data.
- Finally, each application that you use with Pusher has its own application ID, a unique numeric code. If you have different applications running with Pusher, you need to register for additional application IDs.

Once you have those three pieces of information, you can begin to create your Web application. The simple application that you'll create here is a Web page that displays the latest information about a

particular stock. Which stock? Whichever one the publisher has decided to show you. Such updates, of stock names and values, will be sent to your browser via pub-sub, letting you update the page without needing a refresh or any input from the user.

To get this all to work, you'll need an application in three parts. To start off with, let's create a trivially simple Web application using the Ruby-based Sinatra framework. Here is the entire application, which I put in a file named `stock.rb`:

```
#!/usr/bin/env ruby

require "sinatra"
require 'erb'

get "/" do
  erb :index
end
```

This program says that the Web application responds to GET requests on the `/` URL, and nothing more. Requests for any other URLs, or with any other methods, will be met with errors. If you are asked to display the `/` URL, you'll display the ERb (embedded Ruby) file, named `views/index.erb`. You can start your Web application by typing:

```
./stock.rb
```

On my system, I get the following response:

```
== Sinatra/1.3.3 has taken the stage on 4567
   ➔for development with backup from Thin
>> Thin web server (v1.5.0 codename Knife)
>> Maximum connections set to 1024
>> Listening on 0.0.0.0:4567, CTRL+C to stop
```

In other words, if I now make a request for `http://localhost:4567/`, I'll get an error, because the template is not in place. Creating a subdirectory named "views", I then can create the file `index.erb` within it, which is shown in Listing 1.

As you can see, `index.erb` is a simple HTML file. Its body consists of a headline and a single paragraph:

```
<p>Current value of <span id="name">NAME</span> is
   ➔<span id="price">PRICE</span>.</p>
```

The above line is the primitive stock ticker. When your publishing system will send a new stock name and price, you will update this line to reflect that message.

Just as you used a callback to handle incoming messages on your Web socket, you also will need to define a callback to handle messages sent by the publisher to your Pusher "channel", as it is known. (Each application can have any number of channels, and each

channel can have any number of events. This allows you to distinguish between different types of messages, even within the same application.)

In order to do this, you need to load the JavaScript library (from `pusher.com`), and then create a new Pusher object with the key of the account you have created:

```
var pusher = new Pusher('cc06430d9bb986ef7054');
```

You then indicate that you want to subscribe to a particular channel, the name of which does not need to be set in advance:

```
var channel = pusher.subscribe('stock_ticker');
```

Finally, you define a callback function, indicating that when you receive a message of type "update_event" on the `stock_ticker` channel, you want to replace the HTML in the body of this document:

```
channel.bind('update_event', function(data) {
    $("#name").html(data['name']);
    $("#price").html(data['price']);
});
```

Notice that I'm using jQuery here in order to replace the HTML on the page. In order for that to work, I've also brought in the jQuery library,

Listing 1. index.erb

```
<!DOCTYPE html>
<!-- -*-html-*- -->

<head>
  <title>Stock Market</title>

  <script src="http://ajax.googleapis.com/ajax/libs/
↳jquery/1.8.0/jquery.min.js"
↳type="text/javascript"></script>
  <script src="http://js.pusher.com/1.12/pusher.min.js"
↳type="text/javascript"></script>
  <script type="text/javascript">
    var pusher = new Pusher('KEY_FROM_PUSHER');

    var channel = pusher.subscribe('stock_ticker');
    channel.bind('update_event', function(data) {

      $("#name").html(data['name']);
      $("#price").html(data['price']);

    });
  </script>

</head>
<body>
  <h1>Stock Market</h1>

  <p>Current value of <span id="name">NAME</span> is
↳<span id="price">PRICE</span>.</p>

</body>
```


downloading it from Google's servers.

With this HTML page in place, and my Sinatra application running, I'm now ready to receive messages. I run the Sinatra application and point my browser to localhost:4567. I should see the static version of the page, with NAME and PRICE in the paragraph.

Publishing a message is almost as easy as receiving one. Different applications will have different use

cases. Sometimes, you will want to send a message from the Web application itself, indicating that a new message has been posted to a forum or that the number of signed-in users has changed. In other cases, you'll want these updates to come from an external process—perhaps one that is running via cron or is monitoring the database separately from the Web application.

For this particular example, I wrote a

Listing 2. update-stocks.rb

```
#!/usr/bin/env ruby

COMPANIES = %w(ABC DEF GHI JKL MNO)

require 'pusher'

Pusher.app_id = APP_ID_FROM_PUSHER
Pusher.key = 'KEY_FROM_PUSHER'
Pusher.secret = 'SECRET_FROM_PUSHER'

loop do
  company = COMPANIES.sample
  price = rand 100
  Pusher['stock_ticker'].trigger('update_event',
  ➤ { :name => company, :price => price})

  sleep 5
end
```

small Ruby program, `update-stocks.rb`, which is shown in Listing 2. This program uses the “pusher” gem, provided free of charge by the Pusher people. You then choose one of the companies in your list (the constant array `COMPANIES`), then choose a random number up to 100. Next, you send the message to all of the subscribers on the “stock_ticker” channel, indicating that you’ve sent an “update_event”. Because of the decoupled nature of communication between publisher and subscriber, you won’t get an error message if you misspell the channel or event name. Rather, the message will be delivered to no one. Thus, you will want to be particularly careful when writing these and ensure that the same names are used in your client and your server.

Conclusion

Web sockets are going to change the Web dramatically, but it’s not yet clear how or when. Being able to update a large number of client displays almost simultaneously using pub-sub is already changing the way people see Web apps—and as you can see from this small example application, it isn’t very difficult to do. Pub-sub isn’t appropriate for all applications, but if you are sending the same data to many people, and if they might want to receive updates automatically into their browsers, this is an easy and straightforward way to do it. ■

Reuven M. Lerner is a longtime Web developer, consultant and trainer. He is also finishing a PhD in learning sciences at Northwestern University. His latest project, SaveMyWebApp.com, went live this spring. Reuven lives with his wife and children in Modi’in, Israel. You can reach him at reuven@lerner.co.il.

Resources

You can learn about Web sockets from a variety of sources. The W3C’s API and definition are at <http://www.w3.org/TR/2009/WD-websockets-20090423>, in a document that is surprisingly readable. Another good source of information is the book *Programming HTML5 Applications* written by my colleague Zach Kessin and published by O’Reilly.

Web socket servers have been written in nearly every language you can imagine. I found a relatively up-to-date list, with links, on Wikipedia, under the “Web socket” entry, and thus, I’ll not try to reproduce it here.

You can learn more about Pusher at <http://pusher.com> or a popular competitor, PubNub, at <http://pubnub.com>.



DAVE TAYLOR

Counting Cards: *Cribbage*

Dave takes on the challenge of capturing game logic in a shell script.

I've spent the past few months reviewing shell scripting basics, so I think it's time to get back into an interesting project. It's always a good challenge to capture game logic in a shell script, particularly because we're often pushing the envelope with the capabilities of the Bash shell.

For this new project, let's model how a deck of cards works in a script, developing specific functions as we proceed. The game that we'll start with is a two- or three-player card game called *Cribbage*. The basic functions we'll create also will be easily extended to simple *Poker* variants and other multi-card evaluation problems.

If you aren't familiar with *Cribbage*, you've got time to learn more about the game, because I won't actually get to any game-specific elements until next month. Need a good place to learn? Try this: <http://www.bicyclecards.com/card-games/rule/cribbage>.

The first and most obvious challenge with any card game is modeling the deck of cards. It's not just the deck, however, it's the challenge of shuffling too. Do you need to go through the deck multiple times to randomize the results? Fortunately, that isn't necessary, because you can create a deck—as an array of integer values—in sequential order and *randomly pick cards from the deck* instead of worrying about shuffling the deck and picking them in sequential order.

This is really all about arrays, and in a shell script, arrays are easy to work with: simply specify the needed index in the array, and it'll be allocated so that it's a valid slot. For example, I simply could use `deck[52]=1`, and the deck array will have slots `0..52` created (though all the other elements will have undefined values).

Creating the ordered deck of cards,

I don't care what game you're playing, a hand like 3H, 4D, 5D, 9H, 9H and 9H is going to get you in trouble!

therefore, is really easy:

```
for i in {0..51}
do
    deck[$i]=$i
done
```

Since we're going to use the value -1 to indicate that the card has been pulled out of the deck, this would work just as well if everything were set to any value other than -1, but I like the symmetry of `deck[$i]=$i`.

Notice also the advanced for loop we're employing. Early versions of Bash can't work with the `{x..y}` notation, so if that fails, we'll need to increment the variable by hand. It's not a big hassle, but hopefully this'll work fine.

To pick a card, let's tap into the magic `$RANDOM` variable, a variable that has a different value each time you reference it—darn handy, really.

So, picking a card randomly from the deck is as easy as:

```
card=${deck[$RANDOM % 52]}
```

Note that to avoid incorrect syntactic

analysis, it's a good habit always to reference arrays as `${deck[$x]}` rather than the more succinct `deck[$x]`.

How do you know whether you've already picked a particular card out of the deck? I don't care what game you're playing, a hand like 3H, 4D, 5D, 9H, 9H and 9H is going to get you in trouble! To solve this, the algorithm we'll use looks like this:

```
pick a card
if it's already been picked before
    pick again
until we get a valid card
```

Programmatically, remembering that a value of -1 denotes a card that's already been picked out of the deck, it looks like this:

```
until [ $card -ne -1 ]
do
    card=${deck[$RANDOM % 52]}
done
echo "Picked card $card from the deck"
```

The first card picked isn't a problem, but if you want to deal out 45 of the

52 cards, by the time you get to the last few, the program might well bounce around, repeatedly selecting already dealt cards, for a half-dozen times or more. In a scenario where you're going to deal out the entire deck or a significant subset, a smarter algorithm would be to count how many random attempts you make, and when you've hit a threshold, then sequentially go through the deck from a random point until you find one that's available—just in case that random number generator isn't as random as we'd like.

The piece missing in the fragment above is the additional snippet of code that marks a given card as having been picked so that the algorithm identifies twice-picked cards. I'll add that, add an array of six cards I'm going to deal, and also add a variable to keep track of the array index value of the specific card chosen:

```
for card in {0..5} ; do
  until [ ${hand[$card]} -ne -1 ]
  do
    pick=$(( $RANDOM % 52 ))
    hand[$card]=${deck[$pick]}
  done
  echo "Card ${card} = ${hand[$card]}"
  deck[$pick]=-1      # no longer available
done
```

You can see that I've added the use of a "pick" variable, and because the

equation appears in a different context, I had to add the `$(())` notation around the actual random selection.

There's a bug in this code, however. Can you spot it? It's a classic mistake that programmers make, actually.

The problem? The `until` loop is assuming that the value of `hand[n]` is `-1` and remains so until a valid card randomly picked out of the deck is assigned to it. But the value of an array element is undefined when first allocated—not good.

Instead, a quick initialization is required just above this snippet:

```
# start with an undealt hand:
for card in {0..5} ; do
  hand[$card]=-1
done
```

We're almost actually ready to deal out a hand and see what we get. Before we do, however, there's one more task: a routine that can translate numeric values like 21 into readable card values like "Nine of Diamonds" or, more succinctly, "9D".

There are four suits and 13 possible card values in each, which means that the `div` and `mod` functions are needed: `rank = card % 13` and `suit = card / 13`.

We need a way to map suit into its mnemonic: hearts, clubs, diamonds and spades. That's easy with another array:

```
suits[0]="H"; suits[1]="C"; suits[2]="D"; suits[3]="S";
```

With that initialized, showing a meaningful value for a given card is surprisingly straightforward:

```
showcard()
{
    suit=$(( $1 / 13 ))
    rank=$(( ( $1 % 13 ) + 1 ))
    showcardvalue=$rank${suits[$suit]}
}
```

Actually, that's not quite right, because we don't want results like 11H or 1D; we want to convert 1 into an Ace, 11 into a Jack and so on. It's the perfect use for a case statement:

```
case $rank in
    1) rank="A" ;;
    11) rank="J" ;;
    12) rank="Q" ;;
    13) rank="K" ;;
esac
```

Now we're ready to deal a hand and see what we get:

```
for card in {0..5} ; do
    until [ ${hand[$card]} -ne -1 ]
    do
        pick=$(( $RANDOM % 52 ))
        hand[$card]=${deck[$pick]}
```

```
done
```

```
showcard ${hand[$card]} # sets 'showcardvalue'
```

```
echo "Card ${card}: $showcardvalue"
```

```
deck[$pick]=-1 # no longer available
```

```
done
```

And the result of running this? Here are a few iterations:

```
$ sh cribbage.sh
```

```
Card 0: 5D
```

```
Card 1: 5C
```

```
Card 2: JS
```

```
Card 3: QD
```

```
Card 4: 4D
```

```
Card 5: JD
```

```
$ sh cribbage.sh
```

```
Card 0: 10C
```

```
Card 1: 5D
```

```
Card 2: KC
```

```
Card 3: 7S
```

```
Card 4: 4S
```

```
Card 5: 8C
```

Cool. Now that we have the basics of how to model a deck and deal a hand of unique cards, we can start with the interesting elements—next month. In the meantime, your homework is to learn *Cribbage*. ■

Dave Taylor has been hacking shell scripts for more than 30 years. Really. He's the author of the popular *Wicked Cool Shell Scripts* and can be found on Twitter as @DaveTaylor and more generally at <http://www.DaveTaylorOnline.com>.



KYLE RANKIN

More PXE Magic

Learn how to add graphical PXE menus to your PXE server and boot Ubuntu and Debian releases.

When writing this month's column, I realized this will begin my fifth year writing Hack and / for *Linux Journal*. I enjoy writing this column, so thanks to everyone who follows it. For those of you who either e-mail the editor or me directly, thanks for the feedback. (And, for those of you who e-mail me more in-depth questions, I'm sorry I can't always get back to you with full responses. Hopefully, some of those questions will be fodder for future columns.)

This month, I've decided to follow up on a topic I wrote about not in this column directly, but as a feature article called "PXE Magic" in the April 2008 issue. In that article, I talk about how to set up a PXE server from scratch, including how to install and configure DHCP and TFTP. Ultimately, I even provide a basic pxelinux configuration to get you started. Since then, PXE menus with pxelinux have become more

sophisticated and graphical and could seem a bit intimidating if you are new to it. In this column, I explain how to piggyback off of the work the Debian and Ubuntu projects have done with their PXE configuration to make your own fancy PXE menu without much additional work. I know not everyone uses Debian or Ubuntu, so if you use a different distribution, hold off on the angry e-mail messages; you still can use the PXE configuration I'm showing here for your distro, provided it gives some basic examples of how to PXE boot its installer. Just use these steps as a launching off point and tweak the PXE config to work for you.

Simple Ubuntu PXE Menu

If this is your first time configuring a PXE server, for the first step, I recommend following my steps in the "PXE Magic" article to install and configure DHCP and TFTP (it's available on the *Linux Journal*

Web site if you don't have your copy of the magazine handy at <http://www.linuxjournal.com/article/9963>). Otherwise, if you have existing servers in place, just make sure that DHCP is configured to point to your TFTP server (if it's on the same machine, that's fine). And, if you already have any sort of pxelinux configuration in your tftpboot directory, I recommend that you back it up and move it out of the way—I'm going to assume that your entire /var/lib/tftpboot (or /tftpboot on some systems) directory is empty to start with. For the rest of this article, I reference /var/lib/tftpboot as the location to store your PXE configuration files, so if you use /tftpboot, adjust the commands accordingly.

Both Debian and Ubuntu provide a nice all-in-one netboot configuration for each of their releases that makes it simple to PXE boot a particular release yourself. The file is called netboot.tar.gz and is located in a netboot directory along with the rest of the different install images. For instance, the netboot.tar.gz for the i386 Ubuntu 12.04 release (named Precise) can be found at <http://us.archive.ubuntu.com/ubuntu/dists/precise/main/installer-i386/current/images/netboot/netboot.tar.gz>.

To get started, cd to your tftpboot

directory, and then use wget to pull down the netboot.tar.gz file (I'm assuming you'll need root permissions for all of these steps, so I'm putting sudo in front of all of my commands), and then extract the tarball:

```
$ cd /var/lib/tftpboot
$ sudo wget http://us.archive.ubuntu.com/ubuntu/dists/precise/
main/installer-i386/current/images/netboot/netboot.tar.gz
$ sudo tar xzf netboot.tar.gz
$ ls
netboot.tar.gz pxelinux.0 pxelinux.cfg
ubuntu-installer version.info
```

As the ls command shows, an ubuntu-installer directory was created along with pxelinux.0 and pxelinux.cfg symlinks that point inside that ubuntu-installer directory to the real files. Without performing any additional configuration, provided your DHCP and TFTP servers were functioning, you could PXE boot a server with this configuration and get a boot menu like the one shown in Figure 1.

Ubuntu has taken the extra steps of theming its PXE menu with its color scheme and even provided a logo. Unlike the PXE menu I demoed in my previous "PXE Magic" article, this menu functions more like a GUI program. You can use the arrow keys to navigate it, the Enter key to select



Figure 1. Ubuntu Precise PXE Boot Menu

a menu item and the Tab key to edit a menu entry.

Multi-OS PXE Menu

If all you were interested in was PXE booting a single version of Ubuntu or Debian, you would be done. Of course, what if you wanted the choice of either the 32- or 64-bit versions of a particular release, or what if you wanted to choose between a few

different releases? Although you could just overwrite your tftpboot directory every time you wanted to change it up, with only a few extra tweaks to the config, you easily can host multiple releases with the same menu.

Move Precise to a Submenu

To get started, let's clean out any existing files in the /var/lib/tftpboot directory. Let's use the i386 Precise

netboot.tar.gz to begin, but let's tweak how the files are organized by isolating precise in its own directory:

```
$ cd /var/lib/tftpboot
$ sudo mkdir precise
$ cd precise
$ sudo wget http://us.archive.ubuntu.com/ubuntu/dists/precise/
└─main/installer-i386/current/images/netboot/netboot.tar.gz
$ sudo tar xzf netboot.tar.gz
```

All of the interesting PXE configuration can be found inside the ubuntu-installer/i386 directory, so make a copy of those files back in the root tftpboot directory so you can edit them:

```
$ cd /var/lib/tftpboot
$ sudo cp -a precise/ubuntu-installer/i386/boot-screens
└─precise/ubuntu-installer/i386/pxelinux.0
└─precise/ubuntu-installer/i386/pxelinux.cfg .
```

Unfortunately, all of the configuration files under the boot-screens directory you copied reference ubuntu-installer/i386/boot-screens, when you want them to reference just boot-screens, so the next step is to run a quick Perl one-liner to search and remove any instance of ubuntu-installer/i386/ found in the config file:

```
$ cd /var/lib/tftpboot/boot-screens
$ sudo perl -pi -e 's|ubuntu-installer/i386/||' *
```

The specific pxelinux configuration that points to the Ubuntu Precise kernel and initrd can be found under precise/ubuntu-installer/i386/boot-screens/txt.cfg. If you were to look at that file, it would look something like this:

```
default install
label install
    menu label ^Install
    menu default
    kernel ubuntu-installer/i386/linux
    append vga=788 initrd=ubuntu-installer/i386/
    └─initrd.gz -- quiet
label cli
    menu label ^Command-line install
    kernel ubuntu-installer/i386/linux
    append tasks=standard pkgsel/language-pack-patterns=
    └─pkgsel/install-language-support=false vga=788
    └─initrd=ubuntu-installer/i386/initrd.gz -- quiet
```

What you want to do is make a copy of this config file under your root-level boot-screens directory, but because you extracted the tarball into a directory named precise (instead of the root directory), you need to do another search and replace, and add precise in front of any reference to the ubuntu-installer directory. Otherwise, the paths to the kernel and initrd will be wrong:

```
$ cd /var/lib/tftpboot/boot-screens
```

```
$ sudo cp ../precise/ubuntu-installer/i386/boot-screens/txt.cfg
↳precise-i386.cfg

$ sudo perl -pi -e 's|ubuntu-installer|precise/ubuntu-installer|g'
↳precise-i386.cfg
```

When you are done, the `/var/lib/tftpboot/boot-screens/precise-i386.cfg` file should look something like this:

```
default install

label install

    menu label ^Install

    menu default

    kernel precise/ubuntu-installer/i386/linux

    append vga=788 initrd=precise/ubuntu-installer/i386/initrd.gz

    ↳-- quiet

label cli

    menu label ^Command-line install

    kernel precise/ubuntu-installer/i386/linux

    append tasks=standard pkgsel/language-pack-patterns=
    ↳pkgsel/install-language-support=false vga=788
    ↳initrd=precise/ubuntu-installer/i386/initrd.gz -- quiet
```

Finally, open up `/var/lib/tftpboot/boot-screens/menu.cfg` in your favorite text editor. This file contains the bulk of the configuration that has to do with the PXE menu system, and the file should look something like this:

```
menu hshift 13
menu width 49
menu margin 8

menu title Installer boot menu^G
```

```
include boot-screens/stdmenu.cfg

include boot-screens/txt.cfg

include boot-screens/gtk.cfg

menu begin advanced

    menu title Advanced options

    include boot-screens/stdmenu.cfg

    label mainmenu

        menu label ^Back..

        menu exit

    include boot-screens/adtxt.cfg

    include boot-screens/adgtk.cfg

menu end

label help

    menu label ^Help

    text help

    Display help screens; type 'menu' at boot prompt to
    ↳return to this menu

    endtext

    config boot-screens/prompt.cfg
```

What you want to do is replace the `include boot-screens/txt.cfg` line with a submenu that points to the new `precise-i386.cfg` file you created. I used the existing advanced submenu as an example to start from. The resulting file should look like this:

```
menu hshift 13
menu width 49
menu margin 8

menu title Installer boot menu^G

include boot-screens/stdmenu.cfg

menu begin precise-i386
```



Figure 2. Precise in a Submenu

```

menu title Precise 12.04 i386
include boot-screens/stdmenu.cfg
label mainmenu
    menu label ^Back..
    menu exit
include boot-screens/precise-i386.cfg
menu end
include boot-screens/gtk.cfg
menu begin advanced
    menu title Advanced options
    include boot-screens/stdmenu.cfg
    label mainmenu
        menu label ^Back..
        menu exit
include boot-screens/precise-i386.cfg
menu end
include boot-screens/gtk.cfg
menu begin help
    menu label ^Help
    text help
    Display help screens; type 'menu' at boot prompt to
    ↪return to this menu
endtext
config boot-screens/prompt.cfg

```

When you PXE boot now, you should see a menu option labeled Precise 12.04 i386, as shown in Figure 2. When you select that option and press Enter, you then can access the standard install options like before.

Add Precise 64-Bit

Now that you have the 32-bit Precise install working, let's add the 64-bit release as well. You'll basically perform the same initial steps as before, after you remove any existing netboot.tar.gz files.

The netboot.tar.gz file is structured so that it will be safe to extract it in the same precise directory:

```
$ cd /var/lib/tftpboot/precise
$ sudo rm netboot.tar.gz
$ sudo wget http://us.archive.ubuntu.com/ubuntu/dists/precise/
main/installer-amd64/current/images/netboot/netboot.tar.gz
$ sudo tar xzf netboot.tar.gz
```

Since you already copied over the boot-screens directory, you can skip ahead to copying and modifying the 64-bit txt.cfg, so it gets pointed to the

New: Intel Xeon E5 Based Clusters

Benchmark Your Code on Our Xeon E5 Based
Tesla Cluster with:
AMBER, NAMD, GROMACS, LAMMPS, or Your Custom CUDA Codes



Upgrade to New Kepler GPUs Now!

Microway MD SimCluster with
8 Tesla M2090 GPUs
8 Intel Xeon E5 CPUs and InfiniBand
2X Improvement over Xeon 5600 Series



GSA Schedule
Contract Number:
GS-35F-0431N

right directory:

```
$ cd /var/lib/tftpboot/boot-screens
$ sudo cp ../precise/ubuntu-installer/amd64/boot-screens/txt.cfg
└─precise-amd64.cfg
$ sudo perl -pi -e 's|ubuntu-installer|precise/ubuntu-installer|g'
└─precise-amd64.cfg
```

Now, open up `/var/lib/tftpboot/boot-screens/menu.cfg` again, and add an additional menu entry that points to the `precise-amd64.cfg` file you created. The file ends up looking like this:

```
menu hshift 13
menu width 49
menu margin 8

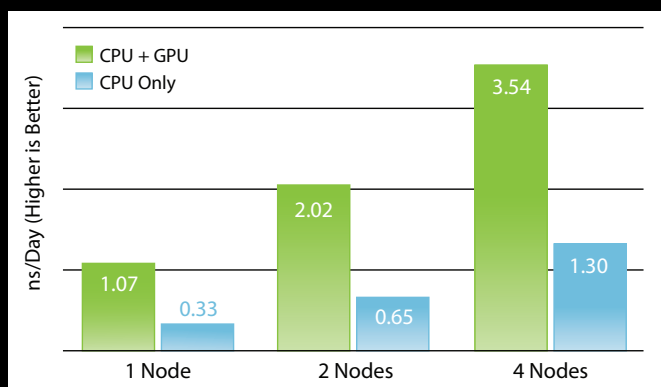
menu title Installer boot menu^G
include boot-screens/stdmenu.cfg
menu begin precise-i386

    menu title Precise 12.04 i386
    include boot-screens/stdmenu.cfg
    label mainmenu
        menu label ^Back..
        menu exit
    include boot-screens/precise-i386.cfg
menu end
```

Harness Microway's Proven GPU Expertise

Thousands of GPU cluster nodes installed.
Thousands of WhisperStations delivered.

- ▶ Award Winning BioStack – LS
- ▶ Award Winning WhisperStation Tesla – PSC with 3D



NAMD F1-ATP Performance Gain

Configure Your WhisperStation or Cluster Today!
www.microway.com/tesla or 508-746-7341




```

menu begin precise-amd64
    menu title Precise 12.04 amd64
    include boot-screens/stdmenu.cfg
    label mainmenu
        menu label ^Back..
        menu exit
    include boot-screens/precise-amd64.cfg
menu end

include boot-screens/gtk.cfg

menu begin advanced
    menu title Advanced options
    include boot-screens/stdmenu.cfg
    label mainmenu
        menu label ^Back..
        menu exit
    include boot-screens/adtxt.cfg
    include boot-screens/adgtk.cfg
menu end

label help
    menu label ^Help
    text help
    Display help screens; type 'menu' at boot prompt to
    ↪return to this menu
    endtext
    config boot-screens/prompt.cfg

```

Add a New Ubuntu Release

So, you were happy with your 12.04 PXE menu, and then Ubuntu released 12.10 Quantal, so now you want to add the 32-bit version of that to your menu. Simply adapt the steps from before to this new release. First, create a directory to store the new release, and pull down and extract the

new netboot.tar.gz file:

```

$ cd /var/lib/tftpboot
$ sudo mkdir quantal
$ cd quantal
$ sudo wget http://us.archive.ubuntu.com/ubuntu/dists/quantal/
↪main/installer-i386/current/images/netboot/netboot.tar.gz
$ sudo tar xzf netboot.tar.gz

```

Next, copy over the quantal txt.cfg file to your root boot-screens directory, and run a Perl one-liner on it to point it to the right directory:

```

$ cd /var/lib/tftpboot/boot-screens
$ sudo cp ../quantal/ubuntu-installer/i386/boot-screens/txt.cfg
↪quantal-i386.cfg
$ sudo perl -pi -e 's|ubuntu-installer|quantal/ubuntu-
installer|g'
↪quantal-i386.cfg

```

Finally, edit /var/lib/tftpboot/boot-screens/menu.cfg again, and add the additional menu entry that points to the quantal-i386.cfg file you created. The additional section you should put below the previous submenus looks like this:

```

menu begin quantal-i386
    menu title Quantal 12.10 i386
    include boot-screens/stdmenu.cfg
    label mainmenu
        menu label ^Back..
        menu exit

```

```
include boot-screens/quantal-i386.cfg
menu end
```

The resulting PXE menu should look something like Figure 3. To add the 64-bit release, just adapt the steps from the above Precise 64-bit release to Quantal. Finally, if you want to mix and match Debian releases as well, the steps are just about the same, except you will need to track down the Debian

netboot.tar.gz from its project mirrors and substitute precise for Debian project names like squeeze. Also, everywhere you see a search and replace that references ubuntu-installer, you will change that to debian-installer. ■

Kyle Rankin is a Sr. Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *The Official Ubuntu Server Book*, *Knoppix Hacks* and *Ubuntu Hacks*. He is currently the president of the North Bay Linux Users' Group.



Figure 3. Now with Three Options



SHAWN POWERS

The Secret Password Is...

If your password is as easy as 123, we need to talk.

The first password I ever remember using when I started in system administration was “.redruM” (no quotes). It was by far the craftiest, most-impossible-to-guess password ever conceived by a sentient being. Sadly, a mere 17 years later (wow, it’s been a long time!) that password probably could be brute-force compromised in ten minutes—with a cell phone.

Since retinal scans still mainly are used in the movies to set the scene for gruesome eyeball-stealing, for the foreseeable future (pun intended), we’re stuck with passwords. In this article, I want to take some time to discuss best practices and give some thoughts on cool software designed to help you keep your private affairs private. Before getting into the how-to section, let me openly discuss the how-not-to.

The Things You Shall Not Do

It’s a bad idea to write your password on a sticky note and affix it to your monitor.

Yes, it sounds like a joke, but this

happens every day—in almost every business. In fact, sometimes tech folks are guilty of this cardinal sin because they’ve changed passwords for users and need to let them know their new passwords. Seeing your password written or typed out should cause you physical pain and distress. Displaying it on your monitor is just wrong.

It’s a bad idea to use any of the following as your password, or at least as your entire password:

- Your pet’s name, current or past.
- Your child’s name or nickname.
- Your car’s name, model or a car you want.
- Birth dates of any people you know.
- Name of your college/high-school mascot.
- Anything related to your hobbies.

- Your address in any form.
- Your telephone number, past or present.
- Your mother's maiden name (this is less secure than .redruM).
- Any of the following: password, 123456, abc123, letmein, love, iloveyou, sex, god, trustno1, master, asdfjkl;, qwerty, password123, secret, jesus or ninja.

If I've just described your password or, heaven forbid, actually listed it in the last bullet point (some of the most common passwords), you need to keep reading. Don't change your password yet though, as I'm going to discuss best practices next, but even if you don't read another word, you can't leave your password like it is—really.

The Things You Shall Try to Do

When it comes to passwords, the longer and more complex, the better. Unfortunately, there is an inverse relationship between the quality of a password and a person's ability to remember it. Logically, one would find the balance between easy to remember and sufficiently complex, but because some people forget how to spell their own names, using some

tricks of the trade is necessary—preferably, combining the tricks.

The Sentence-Mnemonic Method

If I were to tell you my password is "sipmnwnoilbinetb" and that I can remember it every time, you'd probably be impressed. Watch, I'll type it again without looking back: sipmnwnoilbinetb.

Am I really a cyborg with an eidetic memory? Maybe, but in this case, I've just used the sentence-mnemonic method to remember my password. In reality, when I type that password, I'm saying in my head, "Sometimes I pick my nose when no one is looking, but I never eat the boogers."

This particular mnemonic is good for a couple reasons. One, it's easy to remember. Two, it's a *horrible lie*, so no one would ever guess that's what I'm typing. And three, because it's embarrassing, it's unlikely that I'd say it out loud while typing. For most people, just using this method for passwords would be an improvement over their current practice. For the best security, however, it's important to add other complexity.

Substitutionary Complexication

Anyone who was a geek in the 1990s knows that all the cool kids would use numbers in their user names. Whether it was l33th@ck3r or z3r0c00l (or

shawnp0wers), substituting numbers and characters for letters does add a layer of complexity. It's certainly not enough on its own—don't think the crafty use of an @ symbol or a few "3"s for "e"s will keep you safe—but if you add that to the mnemonic method, it certainly will help. "slpmnwn1il,blnetb" looks similar to the eye to my password above, but it is much more resistant to a brute-force attack.

Compound Words

In addition to the above-mentioned methods for increasing complexity, a great way to make your password even more secure basically is to have two passwords separated by a string of numbers or characters. Continuing with our booger-picking example above, what if instead of using a comma to separate the phrases, I used a short string of numbers? On its own, something like 6229 is horribly insecure, but if you do something like "slpmnwn1il6229blnetb", it becomes a really impressive password that is simple to remember. Because I'm talking about the middle of a character string, using an easy-to-remember number is acceptable here.

Based on just a few tricks, I've managed to come up with an excellent password that is easy to remember and not terribly difficult to

type. Yay! I'm done! Well, yes and no.

Hey, That's My Luggage Combination!

The problem is that most people log in to more than one computer system or Web site. Some Web site designers have started to adopt an OpenID sort of authentication system, which allows authentication without actually using a separate password, but that isn't the case everywhere. At least in the near future, we'll be stuck with logins and passwords for multiple Web sites. In a perfect world where Web sites store only well-encrypted passwords, and bad guys never steal password databases, a single well-made password would suffice. That is not the world we live in.

It seems every day there's a company whose Web site has been compromised, and passwords have been leaked. Granted, it's often fun to see what sorts of passwords other people use, but it's a sinking feeling to find your password on the list of compromised—especially if it's the same password you use everywhere. The problem is, coming up with a new password for every Web site is difficult to manage.

If you're consistent and sneaky enough, you might be able to have a "pattern" that only you know. For example:

■ wlvljdc_lapmn = when I visit

Linux Journal dot com, I always pick my nose.

- wlvadc_lapmn = when I visit Apple dot com, I always pick my nose.
- wlvwpdo_lapmn = when I visit Wikipedia dot org, I always pick my nose.

Yes, looking at them side by side, it's easy to tell what the pattern is, but if only one is compromised, it's not terribly clear. Also, in the above examples, I used what letters made sense to me, but they don't line up

with syllables, rather with how the word separation occurs in my head.

One Ring to Rule Them All

For many security-conscious readers, possibly even you, these lessons in good password practice may make you angry. For you, if a password isn't 128-characters long, with a combination of letters, symbols, numbers and fairy spells, it's not good enough. I understand—really, I do. Sadly, I also understand that most of the world still thinks "abc123" is a perfectly cromulent password. For you, my cyborg friend, there are

Powerful: Rhino



Rhino M4700/M6700

- Dell Precision M4700/M6700 w/ Core i7 Quad (8 core)
- 15.6"-17.3" FHD LED w/ X@1920x1080
- NVidia Quadro K5000M
- 750 GB - 1 TB hard drive
- Up to 32 GB RAM (1866 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1190
- E6230, E6330, E6430, E6530 also available

- High performance NVidia 3-D on an FHD RGB/LED
- High performance Core i7 Quad CPUs, 32 GB RAM
- Ultimate configurability — choose your laptop's features
- One year Linux tech support — phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:



Tablet: Raven



Raven X230/X230 Tablet

- ThinkPad X230/X230 tablet by Lenovo
- 12.1" HD LED w/ X@1366x768
- 2.6-2.9 GHz Core i7
- Up to 16 GB RAM
- 750 GB hard drive / 180 GB SSD
- Pen/finger input to screen, rotation
- Starts at \$2050
- T430, T530, W530 also available

Rugged: Tarantula



Tarantula CF-31

- Panasonic Toughbook CF-31
- Fully rugged MIL-SPEC-810G tested: drops, dust, moisture & more
- 13.1" XGA TouchScreen
- 2.4-2.53 GHz Core i5
- Up to 8 GB RAM
- 320-750 GB hard drive / 512 GB SSD
- CF-19, CF-52, CF-H2 also available

EmperorLinux
...where Linux & laptops converge

www.EmperorLinux.com
1-888-651-6686



password management tools.

When every site has a password like "af&6fw^faew^@f88*hIDSLjfe8wlsfyy&&8s0##~", it goes beyond simple mnemonics to remember. Thankfully, there are tools like KeePassX, which is an excellent password manager for Linux, discussed at length by Anthony Dean in the May 2010 issue (<http://www.linuxjournal.com/content/keepassx-keeping-your-passwords-safe>).

The idea behind programs like KeePassX, or the popular browser-based LastPass, is that you can keep your passwords as complex, and even as random, as you like. The programs keep your passwords encrypted and require a master password to unlock them. (When creating a master password, it's very important to follow some sort of complexity strategy, like I outlined earlier in this article.)

With a password manager, you can let your brain keep track of a single password, knowing you can retrieve whatever ultra-safe password you need for a site or computer at any time. Granted, this means relying on a program to keep track of your information, so you'll have to use the program to retrieve it, but with programs like LastPass, there are applications for pretty much every

operating system, browser and smartphone in existence. It is usually the only practical way to keep truly random passwords in order. If you can train yourself to use a program or service to manage passwords, it can change the way you think of security. It also can keep you safe if a particular account is hacked. The system is only as secure as the master password, however, so be sure that's a good one!

Not Quite a Retina Scan...

Thankfully, some companies are taking an honest look at users and realizing password security isn't something they can force feed. Regardless of articles like this, people still will use the names of their dogs to secure their bank accounts. Some companies have begun to use two-step authentication, which adds a physical response to a password challenge.

Someone certainly can steal your password, but what if in order to log in to your e-mail account, you not only had to enter your password correctly, but also had to respond to a text message sent to your phone? It certainly would eliminate the long-distance hacks, because it's unlikely hackers even would *know* your cell-phone number, much less be able to respond to a text message sent to it.

Two-step, or two-factor, authentication

isn't terribly popular yet, but the concept is powerful. If we can continue to come up with complex, yet convenient methods for proving authentication, we will make the world safer and safer. That doesn't mean we can become lax on how we create our passwords, however. Because at least for the near future, secure passwords are the only way to keep our data private.

So Class, What Did You Learn?

You all learned that Shawn apparently picks his nose—at every Web site he visits. Seriously though, hopefully this article has helped you figure out your

own method for creating passwords. Please don't use my exact method, but rather use it to come up with your own. Until we can have retinal scanners on every laptop, we're going to have to secure our passwords the old-fashioned way, like barbarians. So remember, "Sdrphn,iwoae!" (Shawn doesn't really pick his nose, it was only an example.)■

Shawn Powers is the Associate Editor for *Linux Journal*. He's also the Gadget Guy for LinuxJournal.com, and he has an interesting collection of vintage Garfield coffee mugs. Don't let his silly hairdo fool you, he's a pretty ordinary guy and can be reached via e-mail at shawn@linuxjournal.com. Or, swing by the #linuxjournal IRC channel on Freenode.net.

Read *LINUX JOURNAL* on your Android device

Download the app
now in the
**Android
Marketplace.**



www.linuxjournal.com/android

For more information about advertising opportunities within *Linux Journal* iPhone, iPad and Android apps, contact Rebecca Cassity at +1-713-344-1956 x2 or ads@linuxjournal.com.

ROSA Desktop 2012

The fact that Russia's ROSA Labs once collaborated with Mandriva is evident in the company's latest release, ROSA Desktop 2012. Nevertheless, since breaking from Mandriva, ROSA Labs has forked the distro onto its own

unique development path. ROSA Desktop 2012 is an LSB-compliant distro that features a customized KDE desktop. The free edition sports only free software; the Extended Edition includes nonfree components and proprietary software, such as codecs. ROSA Labs says that by developing ROSA Desktop 2012 with its own software development and build environment—ROSA ABF—the company is able to achieve unmatched technological independence, high quality and up to five years of technical support. Examples of new features include EFI/UEFI support, improved hardware detection and improved compatibility with Windows 8. Supported languages include English, French, German, Italian, Portuguese, Romanian, Russian, Spanish and Ukrainian.

<http://www.rosalab.com>



Wind River Intelligent Network Platform

WIND RIVER

The targets in the company's cross hairs for the recently introduced Wind River Intelligent Network Platform are equipment providers who—in the face of exploding traffic—are in need of powerful building blocks to help them build faster, smarter and safer networks. The solution is designed to deliver improved network intelligence, security and speed, and the scalability to support the move to software-defined networking. Wind River claims a 1100% improvement in IP-forwarding and up to a 500% improvement in throughput for UDP and performance boost for TCP. Wind River Intelligent Network Platform is composed of software that manages a consolidated control and data plane system, as well as software engines that provide fast packet acceleration and content inspection. Additional engines for specialized needs, such as accelerated traffic flow-classification, will follow. The platform comes configured with Wind River Linux and a comprehensive suite of integrated development tools.

<http://www.windriver.com>



Digital Defense Inc.'s SecurED Training

For most employees, security is something they leave for folks like us to deal with. With its new SecurED training modules, Digital Defense Inc. (DDI) is employing the oldest trick in the

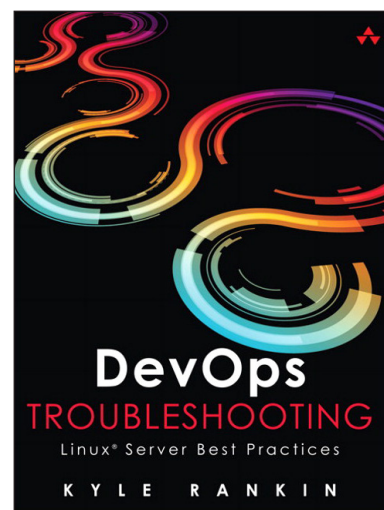
book—humor—to get nongeeks to realize the importance of collaborative enterprise security. DDI has partnered with Emmy-award-winning comedy writer T. Sean Shannon to develop training modules, each 5–7 minutes in length, that promote a culture of security awareness. Humorous situations are used to maintain attention and increase the “stickiness factor”. The modules can be accessed via a PC, laptop, iPad/tablet or mobile device and are accessed through an organization’s LMS. Designed as a year-long program to be viewed monthly, the 12 modules cover topics like password security, acceptable computer use, safe browsing, dangers of social-media sites, preventing viruses and malware and installing software from unknown sources.

<http://www.ddifrontline.com>

Kyle Rankin's *DevOps Troubleshooting* (Addison-Wesley)

The best perk in working for a magazine is that shameless plugs are free. In all seriousness, *Linux Journal* readers certainly will be interested to know that our own Hack and / columnist, Kyle Rankin, has written a new book titled *DevOps Troubleshooting: Linux Server Best Practices*. The purpose of *DevOps* is to give developers, QA and admins a common set of troubleshooting skills and practices so they can collaborate effectively to solve Linux server problems and improve IT performance, availability and efficiency. Kyle walks readers through using DevOps techniques to troubleshoot everything from boot failures and corrupt disks to lost e-mail and downed Web sites. They'll also master indispensable skills for diagnosing high-load systems and network problems in production environments. Addison-Wesley Professional is the publisher (and royalty provider) for *DevOps*. So, Kyle, about those royalties...

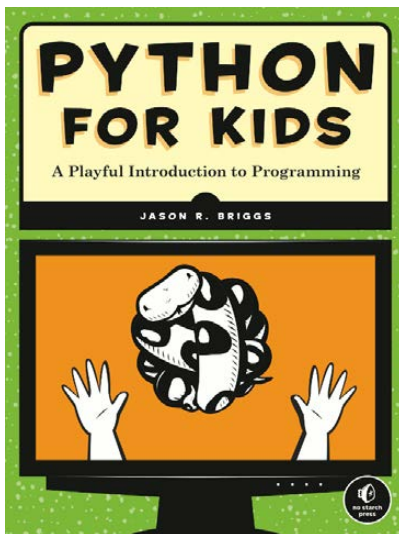
<http://www.informit.com>



Wireload Inc.'s YippieMove API

With the release of the new YippieMove API, Wireload Inc.'s YippieMove is upgrading itself from e-mail migration service to e-mail migration solution provider. Since its inception, YippieMove has been actively slaying the beasts involved in moving e-mail history between different e-mail vendors. With the release of the YippieMove API, YippieMove is aiming to become the go-to e-mail migration service for vendors to integrate with. The move opens up a new market for third-party software developers and enables e-mail vendors and ISPs to create fully automatic inbound e-mail migrations for new accounts. Vendors or ISPs potentially could add a simple step to their sign-up process that would enable users to bring over their e-mail archives from any provider or solution to the newly created account in just a few clicks. The custom-built technology has been built from scratch and perfected by YippieMove during the past four years.

<http://www.yippiemove.com>



Jason R. Briggs' *Python for Kids* (No Starch Press)

Veteran programmer Jason R. Briggs' inaugural venture into book publishing, *Python for Kids: A Playful Introduction to Programming*, aims to inspire the same love of computing that he experienced decades ago hacking his Radio Shack TRS-80. *Python for Kids*, published by No Starch Press, is a lighthearted introduction to the Python programming language, full of fun examples and original color illustrations. Briggs begins the book with the basics of how to install Python

and write simple commands. In bite-sized chapters, he explains essential programming concepts. By the end of the book, readers have built simple games and created cool drawings with Python's graphics library, Turtle. Each chapter closes with offbeat exercises that challenge readers to put their newly acquired knowledge to the test.

<http://www.nostarch.com>



STEC Inc.'s Linux-Based SSD Solutions

We Linuxers always are happy to see new vendors embrace the Linux and open-source paradigm. STEC Inc. is also finding much more to like than it ever imagined after recently developing Linux drivers for its PCI Express-based solid-state drives. STEC reports that coupling the new open-source Linux with the company's high-performance s1120 PCIe Accelerator SSD card dramatically broadens the range of applications for the device. Furthermore, the combination results in "very promising performance results never seen before", such as a boost to Oracle application performance to more than 160,000 transactions per minute. Other improvements include an improvement in application response times by maximizing I/Os per second on a single server and a lower TCO by reducing data-center operational and capital expenditures.

<http://www.stec-inc.com>

txtr beagle

If you are an e-book fan in North America, Kindles or NOOKs probably are the among the first devices that come to mind. If the Berlin-based txtr has its way, however, you soon may find yourself reading e-books on the company's new beagle e-book reader. The txtr beagle, with its 5" screen, 1/4" profile and 4.5oz weight, is billed as the smallest and lightest e-book reader on the market. txtr says the beagle will run for a year on AAA batteries and does not require cables or chargers. It is the first companion reader to receive e-books sent from a smartphone, says the device developer. The Android-based txtr beagle is part of the overall Adobe-certified txtr eReading platform, but does not count as an extra device. txtr believes that these characteristics, as well as a price point for telecom providers potentially as low as \$13.00, are the raw materials for a truly global, mass-market e-book reader.



<http://www.txtr.com>

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o Linux Journal, PO Box 980985, Houston, TX 77098. Submissions are edited for length and content.

ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Cryptography provides stronger security at much smaller key sizes than an RSA. This article explains how it works and how to use it with OpenSSH.

JOE HENDRIX

When it comes to public key cryptography, most systems today are still stuck in the 1970s. On December 14, 1977, two events occurred that would change the world: Paramount Pictures released *Saturday Night Fever*, and MIT filed the patent for RSA. Just as *Saturday Night Fever* helped popularize disco through its choreography and soundtrack, RSA helped popularize cryptography by allowing two parties to communicate securely without a shared secret.

Public key techniques, such as RSA, have revolutionized cryptography and form the basis for Web site encryption via SSL/TLS, server administration via SSH, secure e-mail and IP encryption (IPsec). They do this by splitting the shared secret key used in traditional cryptography into two parts: a public key for identifying oneself and a secret key for proving an identity electronically. Although the popularity of disco has waned, most Web sites today that use encryption still are using RSA.

Since the 1970s, newer techniques have been developed that offer better security with smaller key sizes than RSA. One major breakthrough is the development of cryptography based on the mathematical theory of elliptic curves, called ECC (Elliptic Curve Cryptography). Although ECC has a reputation for being quite complex, it has been integrated into popular open-source cryptographic software including OpenSSH and OpenSSL, and it's not inherently any more difficult to use than RSA. In this

article, I describe ECC and show how it can be used with recent versions of OpenSSH and OpenSSL.

Not all cryptographic algorithms are equal. For a fixed key or output length, one algorithm may provide much more security than another. This is particularly true when comparing different types of algorithms, such as comparing public and symmetric key algorithms. To help make sense of this, the National Institute of Standards and Technology (NIST) reviews the academic literature on attacking cryptographic algorithms and makes recommendations on the actual security provided by different algorithms (see Table 1 from 2011).

Note: for new applications, I think AES-128 should be used over triple DES even if 128-bit security isn't needed. Attacks have been found on SHA-1, and NIST now estimates that SHA-1 provides only 69 bits of security in digital signature applications.

If the system you are designing is expected to protect information only until 2030, NIST recommends

Table 1. NIST Recommended Key Sizes

BITS OF SECURITY	SYMMETRIC KEY ALGORITHM	CORRESPONDING HASH FUNCTION	CORRESPONDING RSA KEY SIZE	CORRESPONDING ECC KEY SIZE
80	Triple DES (2 keys)	SHA-1	1024	160
112	Triple DES (3 keys)	SHA-224	2048	224
128	AES-128	SHA-256	3072	256
192	AES-192	SHA-384	7680	384
256	AES-256	SHA-512	15360	512

DEPARTMENT OF DEFENSE REQUIREMENTS

Although NIST guidance is well respected, the Department of Defense has stronger requirements for classified information. For the Defense Department, 128 bits is only good enough for protecting information classified SECRET. Use of RSA isn't approved, and TOP SECRET information requires use of AES-256, SHA-384 and ECC with a 384-bit key size. Furthermore, systems must use two separate encryption implementations for protection. For example, use both IPsec and TLS, so that the information is still protected by one layer if a flaw in the other is found. Although this may not be very practical for most Internet applications, it's interesting to see what the requirements are when security is paramount.

that you use cryptography providing at least 112 bits of security. For applications that need longer-term protection, NIST recommends at least 128 bits of security.

Just because NIST makes these recommendations, doesn't mean that applications follow them. Many Web sites, including on-line banks, still will use SHA-1 and pair it with AES 128 and a 1024- or 2048-bit RSA key. According to NIST, achieving true 128-bit security means that the RSA key should be at least 3072 bits—a size most Internet certificate authorities don't even offer. At present, Verisign will sell you an SSL certificate that it claims offers "256-bit security", because you can use it with AES-256. The signature itself uses

SHA-1 and a 2048-bit RSA key.

At present, the security on the Internet is still sufficiently weak that it almost always will be easier to find a vulnerability that allows an attacker to bypass security rather than directly attack the encryption. However, it is still worthwhile to be aware of how much security the overall encryption implementation provides. In cryptography, more bits are usually better, but an implementation is only as strong as its weakest length. Both ECC and SHA-2 represent essential algorithms to getting real 128-bit or 256-bit security.

The Mathematics of Elliptic Curve Cryptography

Elliptic Curve Cryptography has a

reputation for being complex and highly technical. This isn't surprising when the Wikipedia article introduces an elliptic curve as "a smooth, projective algebraic curve of genus one". Elliptic curves also show up in the proof of Fermat's last theorem and the Birch and Swinnerton-Dyer conjecture. You can win a million dollars if you solve that problem.

To get a basic understanding of ECC, you need to understand four things:

1. The definition of an elliptic curve.
2. The elliptic curve group.
3. Scalar multiplication over the elliptic curve group.
4. Finite field arithmetic.

Essentially, elliptic curves are points on that satisfy an equation with the form:

$$y^2 = x^3 + ax + b$$

Figure 1 shows a picture of an elliptic curve over the real numbers where a is -1 and b is 1 . Elliptic curves satisfy some interesting mathematical properties. The curve is symmetric around the x axis, so that if (x,y) is a point on the curve, then $(x,-y)$ is also on the curve. If you draw a line

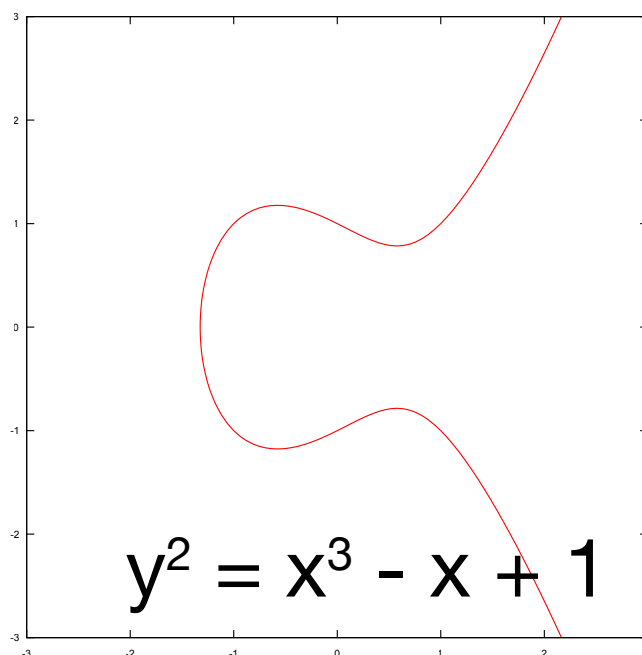


Figure 1. Elliptic Curve over Real Numbers

between any two points on the line with different x coordinates, they will intersect the line at a unique third point. Finally, for each point on the curve, if you draw a straight line tangent to the curve from that point, it will intersect the curve once again at another point.

Mathematicians use these properties to form a structure called a group from the points on the elliptic curve. A group consists of a set of elements containing a special point (denoted $\mathbf{0}$), an operation for negating an element (denoted $-x$), and an operation for adding two elements (denoted $x + y$). The elements in the group defined by an elliptic curve consist of the

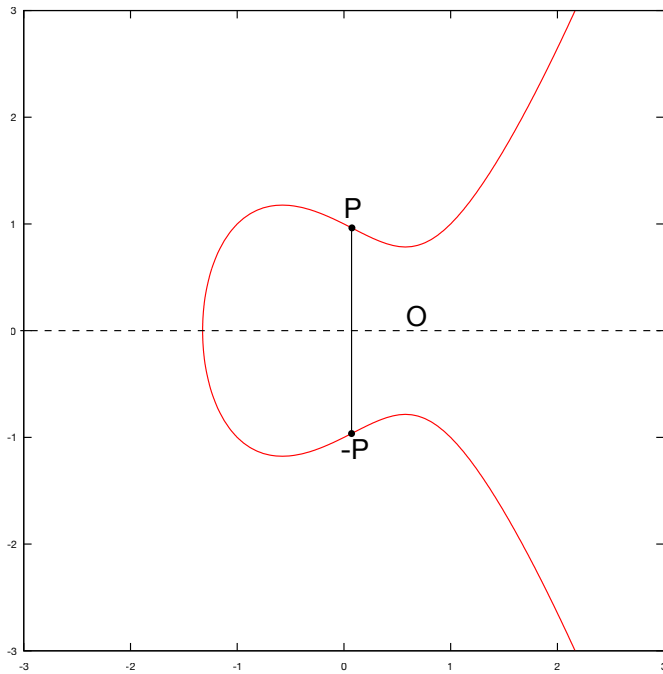


Figure 2. Negating a Point

points on the curve plus an additional point for **0** that is not on the curve, but as you'll see below is easiest to visualize as a line on the x-axis. To negate a point, you just negate the y-coordinate of the point, and adding a point to its negation is defined to return **0** (Figure 2). To add two points P and Q with different x-coordinates, draw a line connecting the two points and extending beyond them. This line should intersect the curve at a third point. The sum $R = P + Q$ is the negation of the third point. Finally, to add a point P to itself, draw the line tangent to P (Figure 3). The sum $R = 2P$ is the negation of the point

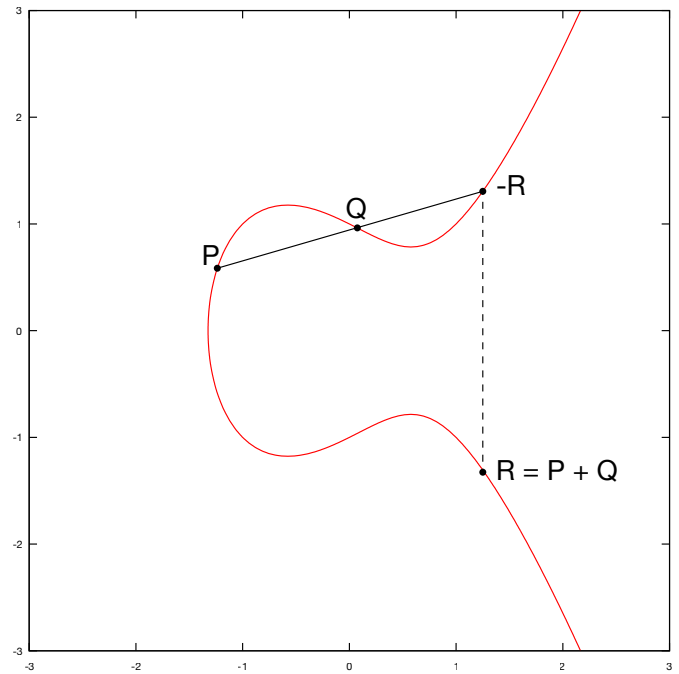


Figure 3. Adding Two Points

that line intersects (Figure 4).

Once the group is defined, we can talk about scalar multiplication—the fundamental operation that makes elliptic curves useful in cryptography. The kth scalar multiple of P is the point obtained by adding P to itself k times. This can be done efficiently by representing k as a binary number and using a double-and-add multiplication method. If you are familiar with RSA, scalar multiplication plays a similar role in ECC that modular exponentiation plays in RSA.

The real numbers used in diagrams for explaining ECC are not practical to use in actual implementations.

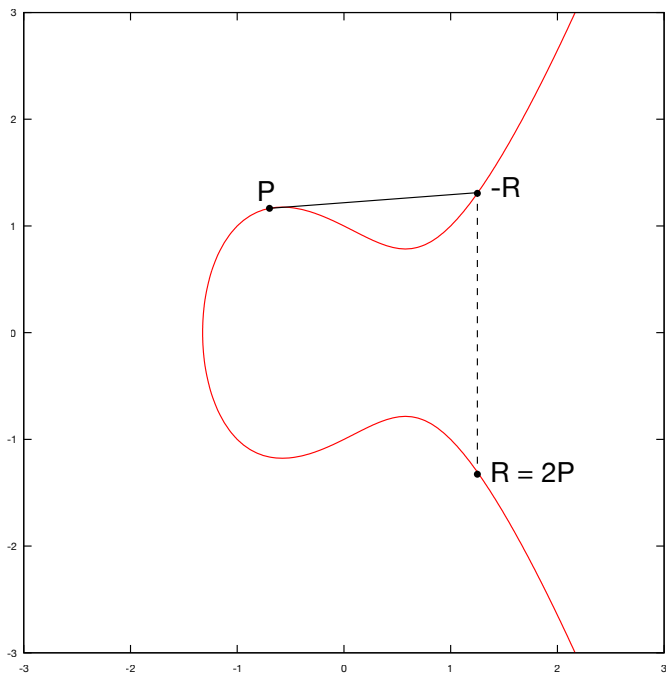


Figure 4. Doubling a Point

Real numbers can have an arbitrary number of digits, and computers have only a finite amount of memory. Most applications, including OpenSSL, use elliptic curves over coordinates that use modular arithmetic, where the modulus is a large prime number. Figure 5 shows the elliptic curve with the same equation as in Figure 1, but where arithmetic is performed modulo 19.

For the different key sizes in Table 1, NIST recommends a specific elliptic curve with a prime modulus for that key size (see the Binary Fields sidebar). For each key size, NIST specifies three things:

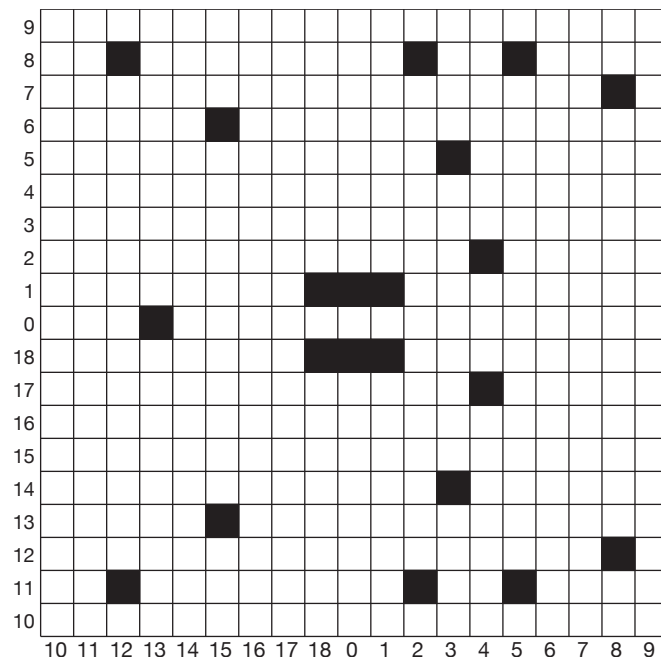


Figure 5. Elliptic Curve over Prime Field (mod 19)

1. The coefficients in the elliptic curve equation.
2. The size of the underlying field for representing x and y .
3. A *base point* that is a point of the curve to be used when calling scalar multiplication.

To see how big the numbers for a 256-bit curve are, the NIST P-256 curve equation has the coefficients $a=-3$ and $b = 41058363725152142129326129780047268409114441015993725554835256314039467401291$.

The coordinates are in a prime field

BINARY FIELDS

For each bit size, NIST also recommends two other elliptic curves over a type of field called a binary field. Although prime fields are more common in software, binary fields are common when implementing ECC in low-power hardware. I focus on prime curves in this article, because that's what OpenSSL uses, and there are a lot more patents on binary curve implementations than prime curves. Unless you have some specific hardware needs and also money to spend on lawyers to deal with patents, I'd recommend sticking to prime curves.

modulo p_{256} where:

$$p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$$

The base point is $G=(x_G, y_G)$ and defined by:

$$x_G = 48439561293906451759052585252797914202762949526041747995844080717082404635286$$

$$y_G = 36134250956749795798585127919587881956611106672985015071877198253568414405109$$

If these numbers look big to you, just think that the 256-bit elliptic curve is equivalent to RSA with 3072-bit numbers. RSA public keys contain more than 12 times the number of digits.

If you'd like to learn more about Elliptic Curve Cryptography, there are many references available. Certicom, a company founded by some of the inventors of ECC, hosts an on-line tutorial at <http://www.certicom.com/ecc-tutorial>. For a more comprehensive understanding

of cryptography, the book *Understanding Cryptography* by Christof Paar, Jan Pelzl and Bart Preneel has a chapter about ECC and also covers the AES and SHA. I've just touched the basic definitions here, and I've not discussed the optimizations used to make a high-performance implementation like the one in OpenSSL. For a quite comprehensive reference on fast ECC algorithms, the "Handbook of Elliptic and Hyperelliptic Curve Cryptography" (<http://www.hyperelliptic.org/HEHCC>) has yet to let me down.

Using Elliptic Curve Cryptography in OpenSSH

A little more than a year ago, OpenSSH 5.7 added support for ECC-based cryptography. Although it's still not in every Linux distribution, support for ECC finally is becoming

widespread enough that it's starting to be worth considering a migration. Support for ECC requires OpenSSH version 5.7 or later and OpenSSL version 0.9.8g or later. OpenSSH can use ECC both to help you authenticate that you really are talking to the server you want and to help the server perform key-based authentication of users.

Host authentication is used by the client to authenticate the server. It is used to detect man-in-the-middle attacks and normally is set up automatically and used by OpenSSH. When OpenSSH is installed, it should create one or more host keys, which normally are stored in `/etc/ssh`. The ECC private key normally is named `ssh_host_ecdsa_key`, and the corresponding public key normally is named `ssh_host_ecdsa_key.pub`. See the man pages for `sshd_config` if you would like to change this path. Just make sure that the private key can be read only by authorized admins; anybody with access to the host private key potentially could impersonate the server.

Client authentication is used to authenticate the client against the server. Using keys to authenticate rather than passwords is both more convenient (because you can use `ssh-agent` or another program

to cache the key) and more secure (because the password is never sent in plain text to the server). If you have used SSH for significant work in the past, you've probably set this up using RSA keys, and the exact same process, namely using `ssh-keygen`, is used to create ECC keys. The only difference is to pass `-t ecdsa` to create the key. The man page for `ssh-keygen` will have more details, and there are many tutorials for setting up SSH keys available on-line if you need a walk-through.

For most people, once encryption software supporting ECC is more widely deployed, converting to ECC should be quick and painless. RSA still is probably "good enough" for most applications, but ECC is significantly more secure, and it may be essential to getting strong security on tiny, low-power, networked devices that are becoming more widespread. Its introduction into open-source tools like OpenSSL and OpenSSH is definitely a great step toward gaining more widespread use. ■

Joe Hendrix is a security researcher who works in Portland, Oregon, for Galois, Inc. His main interest is in applying formal verification techniques to real security problems. He welcomes comments sent to jhendrix@whoisjoe.info.

CONFIGURING ONE-TIME PASSWORD AUTHENTICATION with OTPW

Have you ever wanted to log in to your system from a hotel kiosk or Internet café? That's risky business, but Todd A. Jacobs shows you how to work hard and play safe on public terminals. Todd walks you through the configuration and day-to-day usage of one-time password authentication using OTPW, and even shows you how to integrate it with SSH. Sadly, he offers no advice on what coffee tastes best while connecting safely from a trendy hotspot.

TODD A. JACOBS

Password authentication contains a lot of assumptions about security and trust.

Encrypted SSH tunnels and public key verification are two common ways to ensure that your password is not compromised in transit.

But, what if it's the computer you're currently typing on that can't be trusted?

This isn't just a tinfoil-hat scenario for paranoid penguinistas. There are many everyday situations and common locations where you probably should *not* use your system password, even over a secure tunnel. Examples include:

- A public computer in a hotel, library or Internet café.
- A coworker's virus-infested computer.
- A shared workstation while pair-programming.
- Any place someone could watch you type in your password.

What do all these examples have in common? Essentially, that you're trying to connect to a trusted destination from an untrusted source. This is a complete reversal of what

most authentication systems were designed to address.

Take public key authentication. SSH public key authentication certainly bypasses the password prompt on the remote host, but it still requires you to trust the local machine with your private key password. In addition, once the key is decrypted with your password, the local system has full access to the sensitive key material inside.

Uh-oh—luckily, there's already a solution for this frequently overlooked problem: one-time passwords.

The combination of SSH and one-time passwords is powerful:

- The SSH protocol provides encryption of the login sequence across the network.
- A good SSH client allows you to inspect the remote host's public key fingerprint before entering your credentials. This prevents a rogue host from collecting your one-time passwords.
- The one-time password system ensures that a password can't be reused. So, even if the password is captured in transit, it's worthless to an attacker once you've logged in with it.

OPIE REMOVAL FROM DEBIAN AND UBUNTU REPOSITORIES

Debian began removing OPIE-related packages in early 2011, following some discussions about the security of the binaries, licensing issues and lack of upstream activity.

If you're interested in the details, the following Debian bug reports are relevant:

- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=511582>
- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=622220>
- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=622221>
- <http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=622246>

While the OPIE packages remain in the current Debian stable release at the time of this writing (code-named "Squeeze"), and some unofficial platform ports can be found in the debports repository, OPIE is not available in testing or unstable, and it appears unlikely to be included in the next stable release.

A number of one-time password solutions are available for UNIX-like systems. The two most well-known are S/KEY and OPIE (One-Time Passwords in Everything).

With the recent removal of OPIE from the Debian and Ubuntu repositories, the OTPW one-time password system created by Markus Kuhn provides a viable alternative. Although not a drop-in replacement for OPIE, OTPW offers comparable functionality while providing some interesting features not found in either S/KEY or OPIE.

In particular, OTPW provides:

- Two-factor authentication, consisting of a "prefix password" and a set of autogenerated, disposable suffixes. Even if the list of suffixes falls into the wrong hands, brute force is necessary without the prefix password.
- Protection against certain race conditions and man-in-the-middle attacks through the use of password locks and triplet challenges.

Although not a drop-in replacement for OPIE, OTPW offers comparable functionality while providing some interesting features not found in either S/KEY or OPIE.

- Shared-filesystem support. Because OTPW checks passwords against a list of hashed values stored in a user's home directory, one password list will work for all systems mounting the same \$HOME directory.

Next, I cover installing and using OTPW, with a special focus on integration with OpenSSH.

Package Installation

To make use of OTPW, you need two binaries: `otpw-bin` and `libpam-otpw`. With Debian and Ubuntu, installation is as easy as:

```
sudo apt-get install otpw-bin libpam-otpw
```

If your distribution does not provide OTPW, you can download the source directly from the author's home page. The source tarball does not use GNU autoconf, so you will need to compile and install the binaries manually in accordance with the author's instructions.

Configure PAM

The next step in preparing the system for OTPW is configuration of `libpam-otpw`. A full treatment of PAM is outside the scope of this article, but I cover the most common use cases here.

Changing your PAM configuration can lock you out of your workstation or server, so it's a good idea to keep your existing terminal open until you're sure that things are working correctly. If you have console access, keep a bootable distribution or rescue disk handy. See the [Testing One-Time Password Authentication with SSH](#) sidebar for more information about testing PAM over SSH.

The easiest way to enable OTPW is to put it immediately above `pam_unix` in your `common-auth` configuration file:

```
# /etc/pam.d/common-auth
auth      sufficient  pam_otpw.so
session   optional    pam_otpw.so

auth      sufficient  pam_unix.so nullok_secure
auth      required    pam_deny.so
```

TESTING ONE-TIME PASSWORD AUTHENTICATION WITH SSH

If you are configuring a remote system for OTPW, you should test your PAM stack without closing your current SSH connection. Remember, if you make a mistake with your PAM configuration, you may be unable to authenticate—even with console access—so keep a bootable distribution, such as Knoppix, SystemRescueCD or Finnix handy just in case. Meanwhile, existing logins remain unaffected because they already are authenticated.

In order to test the PAM stack properly, you can't re-use your existing SSH connection. Most recent distributions support SSH multiplexing and persistent connections out of the box, so explicitly disable these options for testing.

In addition, SSH prefers public key authentication by default. So, in order to test OTPW authentication, public key authentication needs to be temporarily disabled too.

The following invocation enables accurate testing of the SSH PAM stack, without making any system changes:

```
read -p 'Hostname: ' REMOTE_HOST &&
SSH_AGENT_PID= SSH_AUTH_SOCK= \
  ssh \
  -o PreferredAuthentications=keyboard-interactive \
  -o ControlPersist=no \
  -o ControlPath=none \
  "$REMOTE_HOST"
```

Once you have confidence that OTPW is working correctly, you also should verify that your other authentication mechanisms (namely SSH public keys and normal system passwords) continue to work as expected.

The order of the PAM libraries is very important. When placing OTPW first, users with an `~/.otpw` file are prompted for a one-time password

first, allowing fallback to standard system passwords if the OTPW login fails. Users without a `~/.otpw` file simply will see the standard

If you're tempted to remove standard system passwords altogether, especially from console logins, please don't.

password prompt.

If you prefer to reverse the order, prompting for a system password before falling back to one-time passwords, just ensure that `pam_deny` comes last:

```
# /etc/pam.d/common-auth
auth      sufficient  pam_unix.so nullok_secure

auth      sufficient  pam_otpw.so
session   optional    pam_otpw.so

auth      required   pam_deny.so
```

If you're tempted to remove standard system passwords altogether, especially from console logins, please don't. On some systems, most notably Ubuntu systems with `ecryptfs`-encrypted home directories, recovering from OTPW mishaps is extremely difficult without standard system passwords.

Modifying `common-auth` is usually the right thing to do on a headless server or console-only system. However, workstations or servers

that provide the X Window System present special problems for one-time password systems.

Some tools or applications won't work properly with OTPW because they can't display the challenge to the user. The typical symptom is usually a password dialog that never completes or seems to ignore user input. In times past, `gksu` and GNOME Display Manager (GDM) had this issue with OPIE. In such cases, the solution is to move OTPW out of `common-auth` and include it only in specific services.

For example, you can add OTPW authentication to SSH connections while using just the standard password prompt for console or GUI logins. You can do this in three easy steps:

1. Delete any lines from `common-auth` that reference `pam_otpw.so`:

```
# /etc/pam.d/common-auth on Debian Squeeze
auth      sufficient  pam_unix.so nullok_secure
auth      required   pam_deny.so
```

2. Create a new OTPW include file for PAM:

```
# /etc/pam.d/otpw
auth          sufficient      pam_otpw.so
session       optional        pam_otpw.so
```

3. Include OTPW immediately *before* common-auth in /etc/pam.d/sshd:

```
# Other stuff ...

# Enable OTPW authentication.
@include otpw

# Standard Un*x authentication.
@include common-auth

# More stuff ...
```

SSH Configuration

In addition to configuring the PAM libraries, OTPW needs the following three settings in the SSH daemon's configuration file:

```
# /etc/ssh/sshd_config
UsePrivilegeSeparation yes
UsePAM yes
ChallengeResponseAuthentication yes
```

These are usually there, but possibly commented out or set to "no", so modify them accordingly. Next, reload the SSH daemon after modifying its

configuration file:

```
# Generic Linux
sudo /etc/init.d/ssh reload

# Debian 6.0.4+
sudo service ssh reload

# Ubuntu 11.04+
sudo reload ssh
```

Generating OTPW Passwords

Once the OTPW PAM module has been configured properly, only users with an ~/.otpw file will be challenged with a one-time password dialog during login. This file contains some metadata about its contents, as well as a list of one-way hashes that will match only a valid response to a challenge.

To create this file, or to re-populate it with new passwords, use the otpw-gen utility. By default, it will create 280 password suffixes, formatted to fit on a single side of US letter-sized (8.5" x 11") paper. Because only the one-way hashes are stored in ~/.otpw, not the passwords themselves, you must capture or print the standard output of this command when the passwords are generated. You will not be able to retrieve the password list after the fact; you'll need to

generate new passwords instead.

Here is what it looks like when you run the command for the first time, piping the output to your default printer:

```
$ otpw-gen | lpr
Generating random seed ...
```

If your paper password list is stolen, the thief should not gain access to your account with this information alone. Therefore, you need to memorize and enter below a prefix password. You will have to enter that each time directly before entering the one-time password (on the same line).

When you log in, a 3-digit password number will be displayed. It identifies the one-time password on your list that you have to append to the prefix password. If another login to your account is in progress at the same time, several password numbers may be shown and all corresponding passwords have to be appended after the prefix password. Best generate a new password list when you have used up half of the old one.

```
Enter new prefix password:
```

```
Reenter prefix password:
```

```
Creating '~/otpw'.
```

```
Generating new one-time passwords ...
```

When generating a new password list, the prompts that appear on standard error are slightly different:

```
Overwrite existing password list '~/otpw' (Y/n)?
```

```
Enter new prefix password:
```

```
Reenter prefix password:
```

```
Creating '~/otpw'.
```

```
Generating new one-time passwords ...
```

The first prompt ensures that you don't accidentally over-write your existing password list; the second prompt asks you for a new password. There's nothing stopping you from reusing the same prefix password on each invocation—the random seed makes duplicate hashes unlikely—but best practice is to use a new prefix each time you regenerate the password list.

If you want to generate a password list on a remote host but print to a local printer, you can do this over your SSH connection as long as you trust your localhost:

```
read -p 'Hostname: ' &&
{
    stty -echo
    ssh "$REPLY" otpw-gen | lpr
    stty echo
}
```

Note the use of `stty` to ensure that your prefix password isn't echoed to the screen. As long

as your prefix password remains secure, you are no worse off using an untrusted printer than you are if your password list falls into the wrong hands. This is often a valuable security trade-off for frequent travelers.

Finally, to disable OTPW challenges for a given user, just delete the `.otpw` file in that user's home directory.

Using OTPW to Log In

Once you have your password list in hand, you're ready to use one-time password authentication for your SSH connection. Assuming that you don't have any identities loaded into your SSH agent, your dialog should look similar to this:

```
$ ssh localhost
Password 015:
```

The prompt with the digits is the OTPW challenge. To respond, find the matching challenge ID on the password sheet you printed earlier. Next, enter your prefix password followed by the string that follows the challenge ID.

Using "foo" as a prefix password, the following suffix list was generated. Your list and suffixes will be different, even if you use the same prefix password.

```
OTPW list generated 2012-05-06 13:40 on localhost
```

```
000 Swvv JGk5 004 =qfF q2Mv 008 sb5P h94r 012 o5aH +/GD 016 8eLV VxuA
001 xPZR :ceV 005 B=bq =mHN 009 WBSR smty 013 QMZ% +bm8 017 vjFL K4VU
002 Sj%n 9xD3 006 RrNx sJXC 010 Xr6J F+Wv 014 j=LO CMmx 018 Km8c 8Q3K
003 s7g8 NE%v 007 sd=E MTqW 011 fNKT vo84 015 fWI% MB9e 019 z8ui %eQ3
```

```
!!! REMEMBER: Enter the PREFIX PASSWORD first !!!
```

To respond to this challenge successfully, type:

```
foo fWI% MB9e
```

at the prompt. The spaces are optional; OTPW ignores them if present.

If you answered the challenge correctly, login will proceed. Otherwise, you will be prompted with the standard system login. At this point, you can enter your standard system password, or press Return to give OTPW another try. After the system-defined number of password attempts (usually three), the login will fail and return you to the command prompt:

```
$ ssh localhost
Password 013:
Password:
Password 013:
Password:
Password 013:
Password:
Permission denied (publickey,password,keyboard-interactive).
```

OTPW AND ENCRYPTED HOME DIRECTORIES

The `ecryptfs` filesystem presents special problems for SSH and OTPW. By default, distributions like Ubuntu unwrap the special passphrase required to mount an encrypted home directory with the user's system password.

This is handled by the `pam_ecryptfs.so` module, which is included through `/etc/pam.d/common-auth` and others. If you authenticate using anything other than your system password, the module prompts you for a system login password in order to mount the encrypted home directory.

In practice, this means that your system password is exposed on untrusted terminals when mounting your remote home directory. This is obviously not ideal.

The best way to avoid this is to leave a console session running at all times. For example, log in at the console using your system password, and then lock the screen. As long as your console session remains active, your home directory remains mounted. As a result, you can use OTPW authentication without further changes to the system, and you won't reveal your system password during login or mounting.

However, if you still want to be able to use OTPW for SSH logins when a console session isn't running—and understand the security implications of doing so—here's how it's done.

First, you need to create a wrapper script for calling `otpw-gen`:

```
#!/bin/bash
set -e
otpw-gen "$@"
mv ~/.otpw /usr/local/lib/otpw/$LOGNAME/
ln -s /usr/local/lib/otpw/$LOGNAME/.otpw ~/
```

The wrapper should be placed in your path and made executable.

Next, place `otpw4ecryptfs.sh` (listed below) in `~/bin` or `/usr/local/sbin`:

```
#!/bin/bash

# Purpose:
```

```
# Enable OTPW for all users on systems with
#   ecryptfs-mounted home directories.

set -e

# Expose the underlying directories that may be
# hidden by ecryptfs.
sudo mkdir -p /mnt/real_home
sudo mount -o bind /home /mnt/real_home

# Collect all non-system users.
users=$(
    awk -F: '$1 != "nobody" \
        && $3 >= 1000 \
        && $3 < 65534 \
        {print $1}' /etc/passwd
)

# Enable OTPW for each non-system user.
for user in $users; do
    sudo mkdir -p /usr/local/lib/otpw/$user
    sudo touch /usr/local/lib/otpw/$user/.otpw
    sudo chown -R $user: /usr/local/lib/otpw/$user
    sudo chmod 755 /mnt/real_home/$user
    ln -sf /usr/local/lib/otpw/$user/.otpw \
        /mnt/real_home/$user/
    ln -sf /usr/local/lib/otpw/$user/.otpw \
        /home/$user/
done < /etc/passwd

sudo umount /mnt/real_home
```

When you run the script, it creates OTPW files that are readable by `pam_otpw.so` even when the user's home directory is unmounted.

Please note that this script gives read and execute permissions to all users' home directories so that `pam_otpw.so` can read the OTPW password files. This is not inherently a risk, but users who rely on more restrictive directory permissions may want to tighten up the permissions of files and folders in their home directories immediately afterward.

Finally, all users should run `otpw-gen-wrapper.sh` to populate and maintain their OTPW password list. Always use the wrapper instead of calling `otpw-gen` directly, or password generation will break the symlinks required for proper operation.

The use of triplets in particular can exhaust your unused passwords rapidly, so it's a good idea to regenerate the password list whenever you fall below a minimum amount.

To prevent simultaneous logins, or when SSH is interrupted during OTPW authentication, OTPW may lock a password. When a password is locked, your next login attempt will present a triplet challenge that requires one prefix and three suffixes to respond:

```
$ ssh localhost
Password 004/011/005:
```

Given the same password list as before, enter your triplet response as a single line, with or without spaces. The following shows how the response is composed (note that the first line below is just an informational aid; you would type only the second line below, without the pipe characters):

```
prefix | suffix 004 | suffix 011 | suffix 005
foo    | =qfF q2Mv | fNKT vo84 | B=bq =mHN
```

Once you have successfully responded to a triplet challenge,

login will proceed and the `~/.otpw.lock` symlink should be removed, and your next challenge will again be a single challenge ID number.

In some cases, the lock is not removed properly. If you continue to be prompted for a triplet, you can remove the lockfile manually:

```
rm ~/.otpw.lock
```

Users with encrypted home directories that are not already mounted before login will need to take a few additional steps. See the OTPW and Encrypted Home Directories sidebar for an example.

Check for Remaining Passwords

If your password list is exhausted, you will no longer be able to use OTPW to log in until a new list is generated. Likewise, if your password list doesn't contain at least three unused responses, you will not be able to use OTPW to log in when `~/.otpw.lock` exists, because there

Listing 1. otpw-stats.sh

```
#!/bin/bash

# 30 unused passwords seems like a reasonable, if
# arbitrary, floor to ensure randomness and a small
# cushion against triplet exhaustion. Feel free to
# adjust this number to suit your needs.
MIN_PASSWORDS=30
OTPW_LIST="$HOME/.otpw"

# Stop processing if OTPW isn't set up for this
# user.
[ -f "$OTPW_LIST" ] || exit

# The top two lines of an OTPW file are meta-data.
TOTAL_PASSWORDS=$((`wc -l < "$OTPW_LIST"` - 2))
# Lines with dashes represent used passwords.
USED_PASSWORDS=$(egrep '^-' "$OTPW_LIST" | wc -l)
# The number of passwords remaining is a calculated
# value.
PASSWORDS_LEFT=$((TOTAL_PASSWORDS - USED_PASSWORDS))

cat << EOF
OTPW Password Statistics
-----
    Passwords used: ${USED_PASSWORDS:=0}
    Passwords left: $PASSWORDS_LEFT

EOF

if [ $PASSWORDS_LEFT -le $((TOTAL_PASSWORDS / 2)) ]
then
    echo "It's time to generate new OTPW passwords."
elif [ $PASSWORDS_LEFT -le $MIN_PASSWORDS ]; then
    echo "Remaining passwords at critical levels."
    echo "It's time to generate new OTPW passwords."
fi
```

are not enough challenge IDs to issue a triplet.

In addition, some of the security of OTPW comes from the randomness of the remaining challenges. The use of triplets in particular can exhaust your unused passwords rapidly, so it's a good idea to regenerate the password list whenever you fall below a minimum amount.

The OTPW author recommends regenerating the password list when less than half the original passwords remain unused, but doesn't define a minimum bound for number of passwords required for adequate randomness of challenges. A small number of unused passwords makes you more vulnerable to brute-force attacks, since there are fewer challenges to present.

The `pam_otpw.so` PAM module is supposed to inform the user when unused passwords fall below half of those generated. However, the PAM session functionality doesn't seem to work on Debian or Ubuntu. In addition, even if it worked, the module doesn't establish a floor

to ensure sufficient randomness of challenges.

The `otpw-stats.sh` script shown in Listing 1 provides this missing functionality. It also allows you to define a sensible minimum for unused passwords by adjusting the `MIN_PASSWORDS` variable at the top of the script.

Add `otpw-stats.sh` to your `~/.profile` (or other shell startup script) to provide feedback at login:

```
# Only run script when logging in via SSH.  
[ -n "$SSH_CONNECTION" ] && ~/bin/otpw-stats.sh
```

Conclusion

OTPW provides a one-time password implementation that compares favorably against OPIE and S/KEY. It is easy to integrate with SSH on most Linux systems, and remains possible to use on Ubuntu systems with encrypted home directories. ■

Todd A. Jacobs is a veteran IT consultant with a passion for all things Linux. He spends entirely too much time making systems do things they were never designed to do. He spends the rest of his time being immensely grateful for a wife who supports his geektastic projects.

Resources

OTPW Source: <http://www.cl.cam.ac.uk/~mgk25/otpw.html>

March 3-6, 2013 → San Francisco

Get the scoop on
SharePoint 2013!



Register Early and SAVE!



The Best SharePoint Training!

Choose from over
90 Classes & Workshops!

Check out these **NEW!** classes,
taught by the industry's best experts!

How to Install SharePoint 2013 Without
Screwing It Up

Todd Klindt  and Shane Young 

What IS SharePoint Development?
Mark Rackley

SharePoint Performance: Best Practices
from the Field
Jason Himmelstein

Creating a Great User Experience in
SharePoint
Marc Anderson 

Ten Best SharePoint Features You've
Never Used
Christian Buckley 

Understanding and Implementing
Governance for SharePoint 2010
Bill English

Building Apps for SharePoint 2013
Andrew Connell 


SharePoint Solutions with SPServices
Marc Anderson 

Lists: Used, Abused and Underappreciated
Wes Preston 

Planning and Configuring Extranets in
SharePoint 2010
Geoff Varosky

Creating Simple Dashboards Using
Out-of-the-Box Web Parts
Jennifer Mason 

Integrating SharePoint 2010 and Visual
Studio Lightswitch
Rob Windsor 

Solving Enterprise Search Challenges with
SharePoint 2010
Matthew McDermott 

Getting Stuff Done! Managing Tasks with
SharePoint Designer Workflows
Chris Beckett 

SharePoint 2013 Upgrade Planning for the
End User: What You Need to Know
Richard Harbridge

Ten Non-SharePoint Technical Issues
That Can Doom Your Implementation
Robert Bogue 

SharePoint MoneyBall: The Art of Winning
the SharePoint Metrics Game
Susan Hanley

Intro to Branding SharePoint 2010 in the
Farm and Online
Randy Drisgill  and John Ross 

How to Best Develop Requirements for
SharePoint Projects
Dux Raymond Sy 



Check out more than
55 exhibiting companies!

A BZ Media Event



Follow us: twitter.com/SPTechCon

SPTechCon™ is a trademark of BZ Media LLC. SharePoint® is a registered trademark of Microsoft.

Lots more online!

www.sptechcon.com

Wi-Fi Mini Honeypot

**Do you have an old,
unused wireless router
collecting dust?
Have some fun and
make a Wi-Fi honeypot
with it!**

MARCIN TEODORCZYK

Recently, I've been playing with some new wireless gear. It's nothing special: 200mW Atheros-based transceiver and 18dBi yagi antenna. I'm living in an apartment in a city of about 640,000 people. I've pointed the antenna to a window and passively received about 30 wireless ESSIDs, three of which were unsecured (open) and six secured with WEP (easily crackable). I haven't connected to any of them, of course, but that gave me some ideas.

What if I deployed a wireless access point deliberately open? Some people

information about attackers as you want. Such honeypots are especially useful in large networks as early threat indicators, but you also can play with them on your home network, just for fun and research.

You can build a wireless honeypot with old hardware, some spare time and, of course, a Linux-based solution. OpenWrt (<https://openwrt.org>) and DD-WRT (<http://www.dd-wrt.com/site/index>) are the two most popular Linux-based firmware projects for routers. I use them and some old spare routers in this article to show

Building a very basic wireless honeypot shouldn't take you more than an hour or two.

eventually will connect and try to use it for Internet access—some might be malicious, and some might think that it's a hotspot. And, what if I deployed a similar access point, but secured with easily crackable WEP this time? Well, in my humble opinion, it's not possible to unconsciously crack WEP. If somebody that I don't know connects to this AP, I've just been attacked. All I need to do is to monitor.

That's exactly a wireless honeypot: fake access point, deliberately unsecured or poorly secured and monitored, so you can get as much

you how to build three kinds of honeypots: a very basic one that logs only information about packets sent by users into its memory, a little more sophisticated one with USB storage that logs a few more details about malicious clients to the storage, and finally, a solution that redirects HTTP traffic through a proxy that not only can log, but also interfere with communication.

Basic Honeypot with DD-WRT

Building a very basic wireless honeypot shouldn't take you more

than an hour or two. Just grab your old router and pick up the firmware. Be sure to look at supported routers

for both DD-WRT and OpenWrt. In my case, it came up that the router is supported only by DD-WRT, as

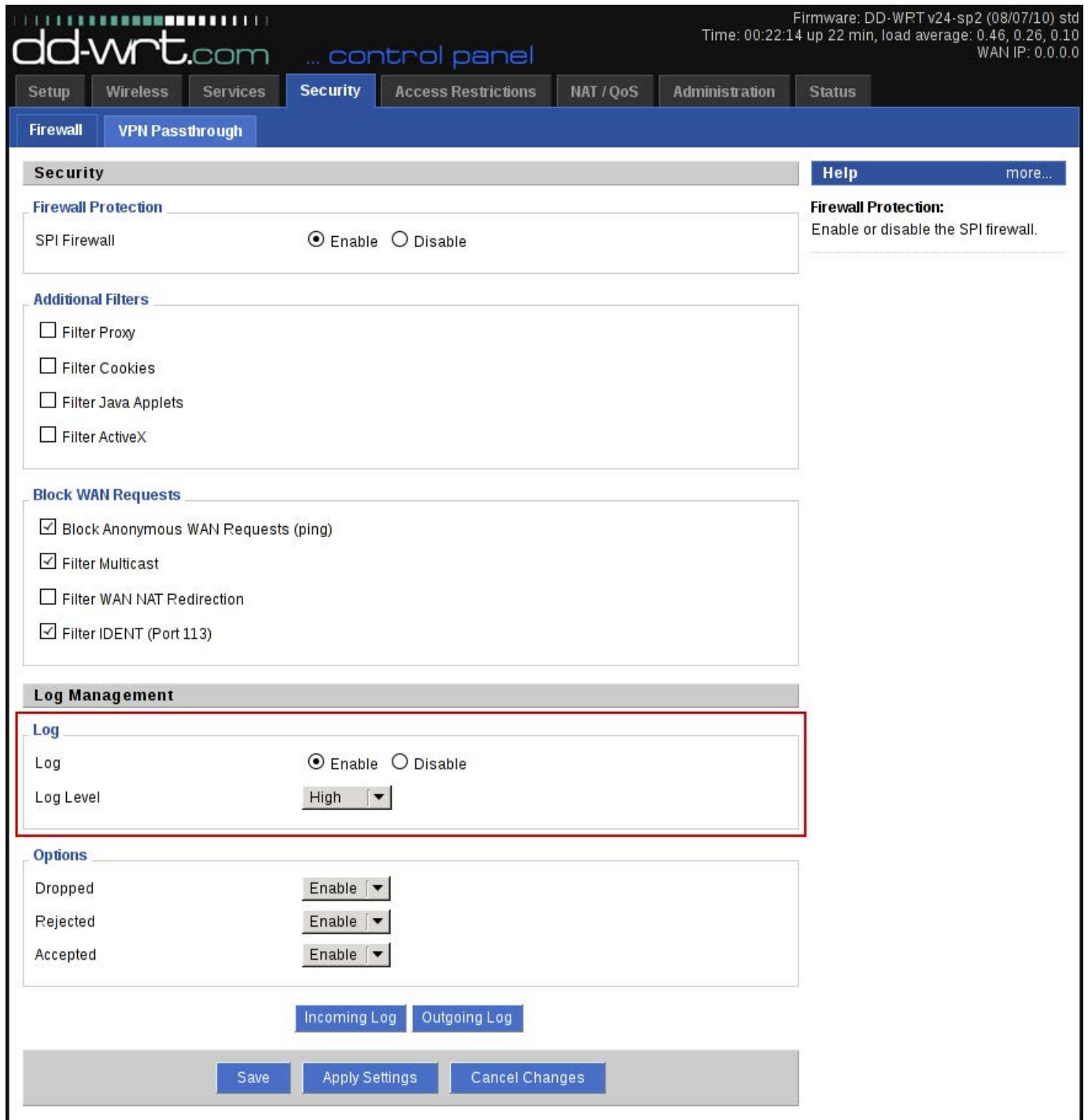


Figure 1. Enabling System Logging

it has 32MB of RAM and 4MB of Flash memory. OpenWrt's hardware requirements are a little bigger.

Next, flash your router (that's the risky part). Basically, you need to download the firmware for your machine and upload it to the memory. On some routers, it's as easy as clicking a button on the Web interface. On others, you have to connect through a serial cable, for example. Remember, this step can be dangerous. Make a backup first and be sure to read the instructions carefully on the DD-WRT/OpenWrt sites.

After successfully flashing your router, you should see an enhanced (as compared to the original one) Web interface. Now, set up SSH access and wireless network parameters. If you don't know how, you can find detailed instructions on the DD-WRT home page. As it is going to be a honeypot, I would suggest WEP, which should attract potential attackers. At the same time, it won't be so vulnerable to false positives—people with devices automatically connecting to an open network.

If you can log in as root and see the prompt, you're ready for the next step: enabling system logging. You can do this using the Web interface: Services→Services→System Log and

Security→Log Enable (Figure 1).

You also can set a few ESSIDs instead of just one: Wireless→Basic Settings→Virtual Interfaces. After that, your honeypot will be seen as a few networks—at least at first glance. This increases the probability of attacks, especially when there are many other networks in your neighborhood.

Remember, you don't have to connect your honeypot to the Internet. In fact, you shouldn't, as you have no control of what potential users might do with the Internet access. After configuring it as described above, test whether it logs your connections. DD-WRT writes the log in /var/log/messages by default. You can check it using SSH. Here's an example fragment of such a log:

```
Jan  1 00:43:03 orange user.warn kernel: ACCEPT IN=br0
➤OUT= MAC=00:26:5a:a1:bc:86:00:0c:f1:11:43:0e:08:00
➤SRC=192.168.2.2 DST=192.168.2.1 LEN=84 TOS=0x00 PREC=0x00
➤TTL=64 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=22535 SEQ=1
Jan  1 00:43:04 orange user.warn kernel: ACCEPT IN=br0
➤OUT= MAC=00:26:5a:a1:bc:86:00:0c:f1:11:43:0e:08:00
➤SRC=192.168.2.2 DST=192.168.2.1 LEN=84 TOS=0x00 PREC=0x00
➤TTL=64 ID=0 DF PROTO=ICMP TYPE=8 CODE=0 ID=22535 SEQ=2
```

If you can see your packet info logged, just leave the router and wait, looking at the log from time to time.

Unfortunately, with such small

resources, you can't do much more—at least within a few hours. This basic honeypot would log only packet headers, IPs and MAC addresses. You can see how a ping command is logged in the previous example. Generally, all the information you can collect is when somebody with a specified MAC and IP try to use your network—that's not much.

Logging Associations to USB Storage with OpenWrt

You can build a little more-advanced wireless honeypot with OpenWrt. Using it, you'll be able to log not only packets, MAC addresses and IP addresses, but also wireless associations, authentications, disassociations, deauthentications and timestamps. With a little effort, you also can expand your honeypot logging capabilities to use USB storage—that gives you a lot more space for logs.

My second router has 32MB of RAM, 8MB of Flash memory and USB support. On such hardware, you easily can install OpenWrt in a similar way as DD-WRT. Detailed instructions are available on the OpenWrt site. After installing it, setting up a wireless access point and logging in via SSH as root, you need to install

a few more packages.

First, you'll need USB storage support:

```
opkg update
opkg install kmod-usb-ohci
opkg install kmod-usb2
insmod usb-ohci
insmod usbcore
insmod ehci-hcd
```

Now, after connecting a pendrive, `dmesg` should show it to you, for example, as `/dev/sda`. Make a directory for mounting your storage: `mkdir /storage`. Then mount it: `mount /dev/sda1 /storage`. You'll use it later for gathered data.

Next, you must decide what traffic to log. Let's assume you want to log all traffic forwarded by the router. To do this, use `netfilter` and `iptables` (<http://www.netfilter.org>): `iptables -I FORWARD -j LOG`, just as you would do in a typical Linux distribution.

Listing 1 shows an example fragment of a log stored on the pendrive. It was generated by the user associating, authenticating, requesting IP through DHCP and connecting to `google.pl:80`.

This honeypot is a little more advanced, although you still don't have much control over user activity on the Internet. You either shouldn't

Listing 1. Example Log Generated with OpenWrt and Stored on a Pendrive

```
Oct 15 10:17:01 white daemon.info hostapd: wlan0:
  ➔STA 00:0c:f1:11:43:0e IEEE 802.11: authenticated
Oct 15 10:17:01 white daemon.info hostapd: wlan0:
  ➔STA 00:0c:f1:11:43:0e IEEE 802.11: associated (aid 1)
Oct 15 10:17:01 white daemon.info hostapd: wlan0:
  ➔STA 00:0c:f1:11:43:0e WPA: pairwise key handshake completed (RSN)
Oct 15 10:17:03 white daemon.info dnsmasq-dhcp[1106]:
  ➔DHCPDISCOVER(br-lan) 192.168.1.99 00:0c:f1:11:43:0e
Oct 15 10:17:03 white daemon.info dnsmasq-dhcp[1106]:
  ➔DHCPOFFER(br-lan) 192.168.1.99 00:0c:f1:11:43:0e
Oct 15 10:17:03 white daemon.info dnsmasq-dhcp[1106]:
  ➔DHCPREQUEST(br-lan) 192.168.1.99 00:0c:f1:11:43:0e
Oct 15 10:17:03 white daemon.info dnsmasq-dhcp[1106]:
  ➔DHCPACK(br-lan) 192.168.1.99 00:0c:f1:11:43:0e red
Oct 15 10:17:14 white user.warn kernel: IN=br-lan OUT=eth0.2
  ➔SRC=192.168.1.99 DST=209.85.148.105 LEN=60 TOS=0x00
  ➔PREC=0x00 TTL=63 ID=59445 DF PROTO=TCP
  ➔SPT=49958 DPT=80 WINDOW=14600 RES=0x00 SYN URGP=0
Oct 15 10:17:14 white user.warn kernel: IN=eth0.2 OUT=br-lan
  ➔SRC=209.85.148.105 DST=192.168.1.99 LEN=60 TOS=0x00
  ➔PREC=0x00 TTL=51 ID=6488 PROTO=TCP SPT=80 DPT=49958
  ➔WINDOW=5672 RES=0x00 ACK SYN URGP=0
Oct 15 10:17:14 white user.warn kernel: IN=br-lan
  ➔OUT=eth0.2 SRC=192.168.1.99 DST=209.85.148.105 LEN=52
  ➔TOS=0x00 PREC=0x00 TTL=63 ID=59446 DF PROTO=TCP
  ➔SPT=49958 DPT=80 WINDOW=229 RES=0x00 ACK URGP=0
Oct 15 10:17:14 white user.warn kernel: IN=br-lan
  ➔OUT=eth0.2 SRC=192.168.1.99 DST=209.85.148.105
  ➔LEN=200 TOS=0x00 PREC=0x00 TTL=63 ID=59447 DF PROTO=TCP
  ➔SPT=49958 DPT=80 WINDOW=229 RES=0x00 ACK PSH URGP=0
Oct 15 10:17:15 white user.warn kernel: IN=eth0.2 OUT=br-lan
  ➔SRC=209.85.148.105 DST=192.168.1.99 LEN=52 TOS=0x00
  ➔PREC=0x00 TTL=51 ID=6489 PROTO=TCP SPT=80
  ➔DPT=49958 WINDOW=106 RES=0x00 ACK URGP=0
Oct 15 10:17:15 white user.warn kernel: IN=eth0.2 OUT=br-lan
  ➔SRC=209.85.148.105 DST=192.168.1.99 LEN=561 TOS=0x00
  ➔PREC=0x00 TTL=51 ID=6490 PROTO=TCP SPT=80
  ➔DPT=49958 WINDOW=106 RES=0x00 ACK PSH URGP=0
```

connect the router to the Internet, filter the traffic with iptables and/or set up a proxy between your router and the Internet. Or, you can set up a proxy on your router!

OpenWrt and Tinyproxy

If your machine has enough resources, you can go one step further and use a proxy on your router. With this, you will be able to monitor, filter

Listing 2. Tinyproxy Configuration with Domain Filtering, Stealth Mode and Custom Log Localization

```
config 'tinyproxy'
    option 'User' 'nobody'
    option 'Group' 'nogroup'
    option 'Port' '8888'
    option 'Listen' '192.168.1.1'
    option 'Timeout' '600'
    option 'DefaultErrorFile' '/usr/share/tinyproxy/default.html'
    option 'StatFile' '/usr/share/tinyproxy/stats.html'
    option 'Logfile' '/storage/tinyproxy.log'
    option 'LogLevel' 'Connect'
    option 'MaxClients' '100'
    option 'MinSpareServers' '5'
    option 'MaxSpareServers' '20'
    option 'StartServers' '10'
    option 'MaxRequestsPerChild' '0'
    list 'Allow' '192.168.1.0/24'
    list 'Allow' '127.0.0.1'
    option 'ViaProxyName' 'tinyproxy'
    option 'DisableViaHeader' '1'
    option 'FilterDefaultDeny' '1'
    option 'Filter' '/storage/filter'
    list 'ConnectPort' '443'
    list 'ConnectPort' '563'
    option 'enable' '1'
```


and modify HTTP traffic. Squid is an example of full-blown proxy solution. If you have a router that is capable of running it, go ahead. If you (like me) don't, you'll have to stick with a solution with fewer requirements.

An example of such a solution is tinyproxy. To install tinyproxy in your OpenWrt, run:

```
opkg update
opkg install tinyproxy luci-app-tinyproxy
```

Listing 3. tinyproxy.log

```
CONNECT Oct 22 16:17:59 [1242]: Connect (file descriptor 7):
  ↳crimson.lan [192.168.1.200]
CONNECT Oct 22 16:17:59 [1242]: Request (file descriptor 7):
  ↳GET / HTTP/1.1
NOTICE Oct 22 16:17:59 [1242]: Proxying refused on filtered
  ↳domain "www.google.com"
CONNECT Oct 22 16:18:05 [1243]: Connect (file descriptor 7):
  ↳crimson.lan [192.168.1.200]
CONNECT Oct 22 16:18:05 [1243]: Request (file descriptor 7):
  ↳GET / HTTP/1.1
CONNECT Oct 22 16:18:05 [1243]: Established connection to host
  ↳"www.linuxjournal.com" using file descriptor 8.
CONNECT Oct 22 16:18:06 [1244]: Connect (file descriptor 7):
  ↳crimson.lan [192.168.1.200]
CONNECT Oct 22 16:18:06 [1244]: Request (file descriptor 7):
  ↳GET /pixel/p-a3K3N6enFe9wA.gif HTTP/1.1
NOTICE Oct 22 16:18:06 [1244]: Proxying refused on filtered
  ↳domain "pixel.quantserve.com"
CONNECT Oct 22 16:25:52 [1246]: Connect (file descriptor 7):
  ↳crimson.lan [192.168.1.200]
CONNECT Oct 22 16:25:52 [1246]: Request (file descriptor 7):
  ↳GET / HTTP/1.1
NOTICE Oct 22 16:25:52 [1246]: Proxying refused on filtered
  ↳domain "www.google.com"
```

Then, configure and run it with:

```
uci set tinyproxy.@tinyproxy[0].enable=1
uci commit
/etc/init.d/tinyproxy enable
/etc/init.d/tinyproxy restart
```

From now on, your tinyproxy should listen by default on port 8888 on your localhost. You can check this with the `netstat` command. Since you want to accept connections not only from localhost, but also from LAN, you'll have to change the configuration a little bit. Also, in our case, it's better to

run it in so-called stealth mode—that means no added headers in HTTP. You can find the tinyproxy configuration in the `/etc/config/tinyproxy` file. Listing 2 shows an example of such a configuration. Notice that logfile is specified to be in the `/storage` directory, which is our pendrive. Another important option is `list 'Allow'`. These are the IPs that are allowed to connect to the tinyproxy. You should specify your LAN network or a part of it.

Tinyproxy also lets you filter requests by domain. You can specify a blacklist or a whitelist of domains

Listing 4. tinyproxy.html

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  ➔"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Information</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<h1>Information</h1>
{clienthost} <br /><br />
You shouldn't use this network for web access.
</body>
</html>
```

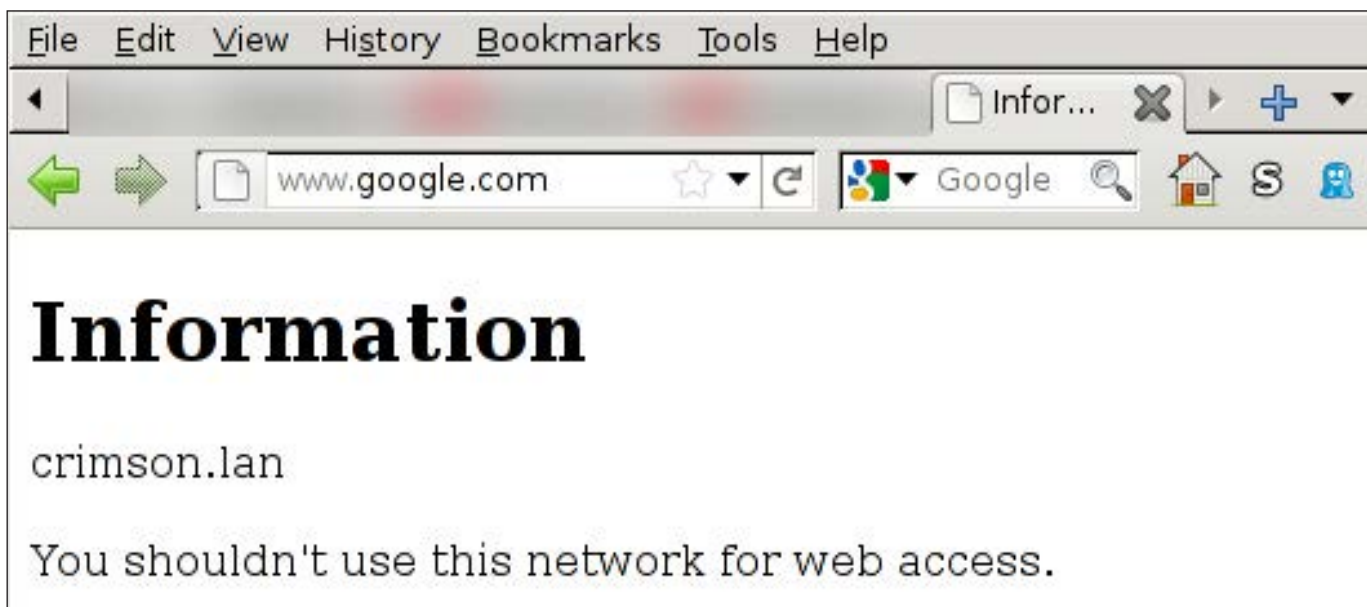


Figure 2. Web Site Template

in the Filter file. In our configuration, this is `'/storage/filter '`. Also, here we notify tinyproxy to treat this file as a whitelist (`FilterDefaultDeny 1`), meaning that requests only for specified domains will be allowed. That way, you can forbid attackers from accessing the Internet with their browsers or let them access only specified domains. An example of a `/storage/filter` file could be:

```
linuxjournal.com
```

That would let them visit only the *Linux Journal* Web site.

Keep in mind that this will block only HTTP requests; all the other traffic will be allowed if you haven't

blocked it elsewhere.

Finally, you must tell your router to forward all HTTP traffic to your new proxy. As usual, you can do this with iptables, but first you need to install the `iptables-mod-nat-extra` package:

```
opkg install iptables-mod-nat-extra
iptables -A PREROUTING -t nat -p tcp --destination-port 80
-j REDIRECT --to-port 8888
```

From now on, all HTTP requests should be forwarded through tinyproxy and logged to `/storage/tinyproxy.log`. Listing 3 shows a fragment of such a log. You can see what connections the user tried to make and what has been filtered by the proxy.

When tinyproxy filters a connection request, it displays an information page describing what happened. You also can make use of this to hide tinyproxy's presence or to inform or deceive your attackers (or make a joke). The Web site template is in `/usr/share/tinyproxy/default.html`. You can see a slightly modified version of this in Listing 4 and Figure 2. This doesn't tell users about tinyproxy and the reason for seeing this page; instead, it politely informs them that they shouldn't use this network for Internet access.

Go with the Evolution

The next step in the evolution would be a full-blown wireless honeypot. You can make one using a machine that can run a typical Linux distribution. Then install, for example, *dionaea* (<http://dionaea.carnivore.it>), use a wireless card configured to run as an access point and forward all traffic to your localhost, on which the attacker will see fake services.

Remember, if you want a really good honeypot, make sure that it looks as close as possible to reality. That means, for example, that you might use some dummy clients just to simulate traffic. Or, use WPA instead of WEP. It all depends on your environment.

Also, it is important to be familiar with your country's laws. Make sure it's not prohibited to sniff your attackers' data. Think about whether it's wise to make an Internet connection available for them. Maybe it would be better not to connect your router's WAN port to anything at all, connect it to your machine simulating the Internet or connect it to the Internet but filter the traffic with iptables?

Finally, don't be discouraged by the DD-WRT or OpenWrt systems. They are based on Linux and are very similar in use, but because of the small resources available, they're stripped down. There are no manual pages and slightly different utilities that you may know from your Linux distribution, even though they are named the same. And, the documentation isn't always accurate. If you have any problems, both projects' wikis are very helpful.

And, last but not least, have fun building your solution and (especially) with browsing the collected data! ■

Marcin Teodorczyk is a GNU/Linux user with more than 12 years of experience. For the past four years, he's been using Arch Linux exclusively on his personal computers. Marcin has an M.Sc. degree in IT and works as a security officer. In his spare time, writes articles for IT magazines or...juggles.



ACHIEVEMENT UNLOCKED! MASTER OF THE WEB

1-UP YOUR ABILITIES

PHP - PYTHON - RUBY - JAVA - .NET
HTML5 - JAVASCRIPT - MOBILE

EXPAND

YOUR SKILLS WITH EXPERTS
FROM ACROSS THE GLOBE.

EXPLORE

DIVERSE TECHNOLOGIES WITH
160 PRESENTATIONS.

EXPERIENCE

THE BEST OF WEB COMMUNITY AND
CULTURE.

- **START**
REGISTER ON CONFOO.CA
BEFORE JAN. 20
FOR A DISCOUNT

FOLLOW
@CONFOOCA

SPONSORED BY:



FooLab



Phonegap Application Development

Phonegap: the easy way to develop smartphone applications.

MIKE DIEHL

How many times have you heard, “there’s an app for that”? But sometimes, there actually isn’t “an app for that”, or the apps that do exist don’t meet your needs. As Linux users, we tend to like to scratch our own itches, and if that means we write some code to do it, so be it. However, writing code to run on an Android phone or tablet has a bit of a learning curve, and it’s even worse on Apple products. Fortunately, Phonegap provides a simple way to create standalone apps for Android, iPhone, WebOS, Blackberry and Windows Phone, among others. You just need to be reasonably proficient in HTML, JavaScript and CSS, and you can develop native apps for the majority of smartphones currently in use. And, the same code base can run, with obvious

limitations, on any Web browser.

Developing native code for Android is relatively easy. You’ll have to learn to use Android’s XML-based screen layout mechanism, and you’ll have to learn Java. For iPhone, you’ll need to learn Objective C. If you want to develop for Windows Phone, you’ll need to learn C# as well. Instead, you simply could use Phonegap and maintain a single code base in HTML/JavaScript/CSS. This is the definition of a “no-brainer”.

Before I go much further, I need to clear up a potential source of confusion. Phonegap initially was developed by a company named Nitobi, which subsequently was acquired by Adobe. In 2011, Nitobi/Adobe donated the Phonegap code base to the Apache Foundation. As a result of this

contribution, they needed to ensure that the intellectual property was unfettered by trademark ambiguity, so they renamed the Phonegap project to Cordova. The Apache Foundation is in the process of migrating from Phonegap to Cordova, so I refer to this project as Cordova here.

Getting started with Cordova on Android isn't difficult. At the risk of rehashing material that is well documented elsewhere, I'll just outline the process involved. First, you have to install the Android SDK, which is a free download from the Android site and is very well documented. The Android SDK integrates with the Eclipse IDE, so you will need to have a fairly recent version of Eclipse as well. The SDK documentation will walk you through the whole process, from downloading the software to building and running the sample application. The SDK lets you run your program in an emulator or on a real Android device, if you have one.

Installing Cordova is also fairly straightforward and well documented. The only difficulty I had with the entire process is that I wasn't very familiar with Eclipse and stumbled a bit. The Cordova installation process culminates with building and running the sample Cordova application. The sample application demonstrates

much of Cordova's API and is worth looking at.

I found the process of creating a new Cordova project to be a bit kludgy. The process involved creating a new Android project first, then making a two-line modification to a Java program, pasting in a dozen lines of XML into another file, and well, you get the idea. All of the changes made sense, but seemed a bit error-prone. Finally, I decided to copy the example project and strip it down to its bare necessities. This is the approach that I recommend; it worked like a champ for me.

A Cordova application has three main pieces. There is an architecture-specific binary piece that actually communicates directly with the device's hardware. Then there is a Java-based abstraction layer that sets up your application's runtime environment and presents a JavaScript API for your application to use. The third part is your HTML/JavaScript/CSS code, and this is the only part that you normally need to be concerned with. All of these pieces get linked together at build time to form a native binary executable for the target device.

The Cordova JavaScript API allows your program to access many of the host device's sensors. This means that your application has easy access to the

device's GPS, accelerometer, compass, microphone and speaker. The API provides persistent data storage by allowing access to the device's contact database, its filesystem and a native SQLite database.

Let's look at some code.

For the sake of illustration, I developed a simple application. The application is designed to demonstrate three main features: access to the device's GPS sensor, access to the user's contacts and the ability to make Ajax calls to remote Web services.

The HTML needed to create this application is pretty straightforward. See Listing 1.

The content of the `<head>` section is mostly boilerplate. Note that you import the `cordova.js` and then your `main.js` JavaScript files, and that the order is important. In the `<body>`, you find a graphic that you bring in from a remote server. Then you see input fields for your current GPS coordinates. Next, you have some form fields that will contain information from the phone's contact directory, followed by Previous and Next buttons that allow users to scroll through their contacts. Finally, there are two ``s that will allow the program to display witty comments from a remote Web site and any error messages that might need to be

displayed. Figure 1 shows what the page looks like in a browser.

Listing 2 shows the JavaScript code that makes it all work.

Line 1 is a simple boolean flag that determines whether the script is running on a mobile device or a



Figure 1. Sample Application Running in a Browser

Listing 1. HTML for the Sample Application

```

<html>
<head>
<title>Sample Application</title>
<meta name="viewport" content="width=320; user-scalable=no" />
<meta http-equiv="Content-type" content="text/html; charset=utf-8">
<link rel="stylesheet" href="./master.css" type="text/css"
  ↳media="screen" title="no title">

<script type="text/javascript" charset="utf-8"
  ↳src="cordova-1.9.0.js"></script>
<script type="text/javascript" charset="utf-8"
  ↳src="main.js"></script>
</head>

<body style="{background: beige;}">

<br>
<p>
You are here:<br>
<input name="lon", id="lon" size="15"> Longitude, <br>
<input name="lat", id="lat" size="15"> Latitude.<br>
<p>
<input id="id" name="id"><br>
<input id="name" name="name"><br>
<input id="phone" name="phone"><br>
<input id="email" name="email"><br>
<button onclick="previous_contact();">Previous</button>
<button onclick="next_contact();">Next</button>
<p>
<hr>
"<span id="quote">Linux Rocks!</span>"<br>
<span id="error"></span><br>
<hr>
</body>
</html>

```

Listing 2. JavaScript Code for the Sample Application

```

1 var mobile = 1;
2 var contacts;
3 var current_contact = 0;
4
5 function init () {
6
7 if (mobile == 1) {
8 navigator.contacts.find(["*"], store_contacts);
9 }
10
11 update();
12 window.setInterval(update, 1000);
13 }
14
15 function update () {
16 var req;
17
18 if (mobile == 1) {
19 navigator.geolocation.getCurrentPosition
    ➤(set_location, location_error);
20 } else {
21 document.getElementById("lat").value =
    ➤Math.floor(Math.random()*46)
22 document.getElementById("lon").value =
    ➤Math.floor(Math.random()*46)
23 }
24
25 req = new XMLHttpRequest();
26
27 if (!req) {
28 alert("Ajax failed!");
29 return false;
30 }
31
32 req.open("GET", "http://example.com/test.html", true);
33 req.onreadystatechange = set_quote;
34 req.send(null);
35
36 return true;
37 }
38
39 function store_contacts (c) {
40 contacts = c;
41 display_contact();
42 return true;
43 }
44
45 function previous_contact () {
46 current_contact = current_contact - 1;
47 if (current_contact < 0) { current_contact = 0; }
48 display_contact();
49 return true;
50 }
51
52 function next_contact () {
53 current_contact = current_contact + 1;
54 if (current_contact > (contacts.length-1)) { current_contact =
    ➤contacts.length-1; }
55 display_contact();
56 return true;
57 }
58
59 function display_contact () {
60 document.getElementById("id").value = " ";
61 document.getElementById("name").value = " ";
62 document.getElementById("phone").value = " ";
63 document.getElementById("email").value = " ";
64
65 document.getElementById("id").value =
    ➤contacts[current_contact].id;
66 document.getElementById("name").value =
    ➤contacts[current_contact].displayName;
67 document.getElementById("phone").value =
    ➤contacts[current_contact].phoneNumbers[0].value;
68 document.getElementById("email").value =
    ➤contacts[current_contact].emails[0].value;
69
70 return true;
71 }
72
73 function set_location (p) {
74 document.getElementById("lat").value = p.coords.latitude;
75 document.getElementById("lon").value = p.coords.longitude;
76 return true;
77 }
78
79 function location_error (e) {
80 document.getElementById("error").innerHTML = e.message;
81 return true;
82 }
83
84 function set_quote (p) {
85 if (!p) { return 1; }
86 if ((p.status) && (p.status > 299)) { return 1; }
87 document.getElementById("quote").innerHTML = this.responseText;
88 return true;
89 }
90
91 if (mobile == 1) {
92 document.addEventListener("deviceready", init, false);
93 } else {
94 window.onload = init;
95 }

```

Web browser. Setting this variable to 0 allows me to run and debug the program in Firefox where I have all of the HTML, DOM and JavaScript development tools that I'm accustomed to using. Setting this variable to 1 targets the program for a mobile device where I can debug the Cordova-specific aspects of my program, knowing that my JavaScript is probably correct.

Lines 91–95 arrange to have the JavaScript `init()` function called when the DOM is loaded and after the Cordova initialization routines have run. These lines also point out a couple oddities about Cordova development. First, there is no way to detect automatically whether the program is running in a browser or on a smartphone. That's why I set that variable, as discussed earlier. Also, Cordova creates its own event that gets triggered when it's ready to begin JavaScript execution; you can't use `window.onload` as you normally would, because this event might trigger before Cordova is ready. Either way, the `init()` function will be called at the appropriate time.

The `init()` function is on lines 5–13. On line 8, you make a call to the `contacts.find` method to get an array of contact objects from the device's contact directory. This array

is then passed, asynchronously, to `store_contacts()`, lines 39–44, which simply stores the array in a global variable. Then, `init()` makes a call to `update()` to initialize the data display and arranges for `update()` to be called every second from then on.

The `update()` function, lines 15–37, is where the fun begins. If the program is running in a browser, you simply populate the Longitude and Latitude fields with random numbers. Having the numbers change like that allowed me to verify that the program was still running. However, if the program is running on a physical device, you use the `geoLocation.getCurrentPosition` method to fetch the real GPS coordinates. If this operation is successful, `set_location()` is called. Otherwise, `location_error()` gets called, and you can display an error message (lines 73–83). The only error I've encountered with the `getCurrentPosition` call was when I actually had the GPS disabled.

Lines 25–36 form an almost embarrassing Ajax call. I've stripped this code down to the least amount of code that would run under Firefox and Cordova. It won't run on IE, and it doesn't do much, if any, error checking. I'm not trying to demonstrate *how* to do an Ajax

call in Cordova. I'm only trying to demonstrate that you *can*. In this case, you're loading some content from a remote server and putting it inside the quote `<div>` discussed earlier. During development, I'd simply change the content of that file on the server to verify that it changed inside the app.

Lines 44–58 are onclick handlers for the two buttons in the application. All these routines do is adjust an array index plus or minus one, as appropriate, and do some bounds checking. Finally, they call `display_contact()` to display the current contact.

The `display_contact()` function (lines 59–72) is the last of the Cordova-specific functions in the program. In lines 60–64, you blank out all of the contact fields in preparation for setting them with new values. I found that if I didn't blank them out first, they would persist into the next record if the next record didn't happen to have a value for a given field. In lines 65–69, you populate the fields with data from the current contact record. Note that both `phoneNumbers` and `e-mails` are arrays of objects, and that for this purpose, you are interested only in the first element.

And there you have it. There's nothing here that would be unfamiliar to the average Web developer, except a really powerful API. But, I've

only touched on what this API can do. Figure 2 shows the application running on my Droid Bionic.

I did some hand waving over a subtle problem with this application that many JavaScript, particularly Ajax, developers have encountered. On most browsers, your program can't

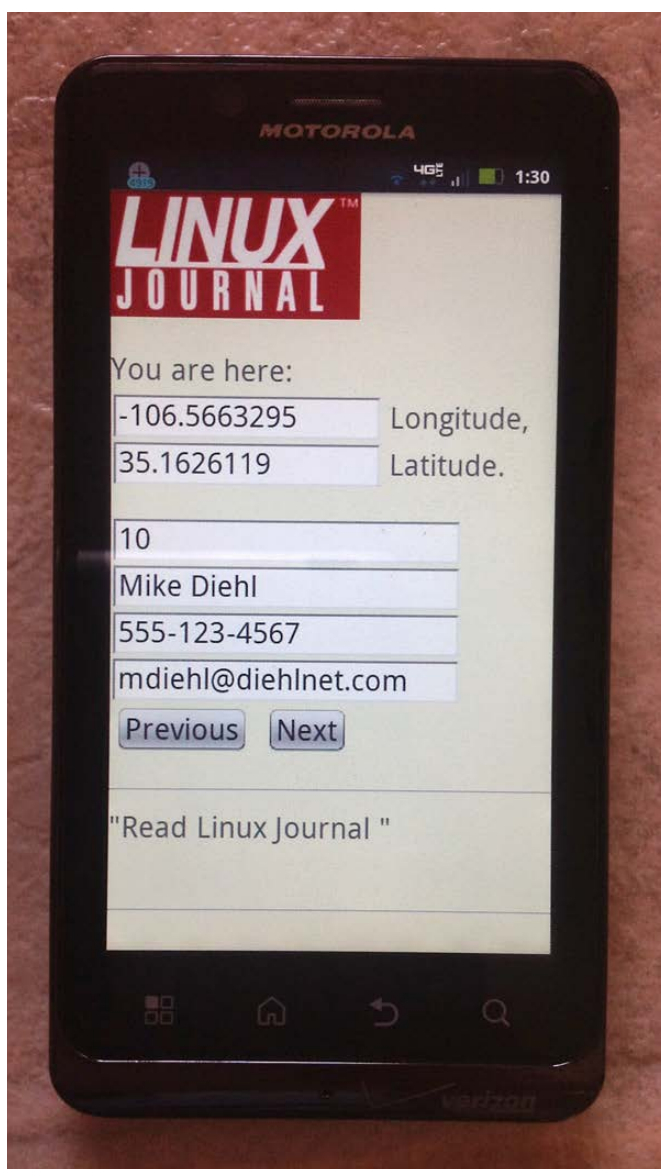


Figure 2. Sample Application Running on Android

load from one domain, and then load content or code from another domain. However, this program is standalone and needed to load content from a potentially arbitrary Web site. Cordova handles this problem by “whitelisting” the domains from which an application is allowed to fetch content. By default, all domains are blacklisted, and, thus, all network access is disabled. A developer can whitelist a domain by editing `/res/xml/cordova.xml` and following the examples given for the `<access>` tag. This is a safe but elegant solution to a potentially nasty problem.

Another interesting possibility is to have your application load all of its HTML and JavaScript from a remote Web server. This easily can be done by making a simple change to `./src/{projectname}/{projectname}.java`. This file has only 20 lines of real code, and the necessary change is pretty intuitive.

Being able to load content from a remote server actually makes development easier. I found it easier to do my development on a remote, publicly accessible server, than to develop on my workstation. This way, I could point my Web browser at the application and get all my HTML, CSS and JavaScript working the way I wanted. Then I got the application fully functional on my Android. Once

it was fully functional, I copied the project to my workstation for the final build. Doing it this way is the only way you’ll be able to test an application that makes any Ajax calls without violating your browser’s cross-site scripting security policy.

As neat as Cordova is, there are a few things I didn’t like. As I mentioned earlier, there is no way to detect automatically whether a program is running in a browser or on a device. Also, I found the whitelisting functionality to be a bit buggy, but not in any way that broke my application. But, the most disheartening thing I found was when I tried to use the camera API. Instead of simply snapping a picture and either returning the data or storing it, the API actually brought up the device’s native camera device as a pop-up. This was extremely intrusive and actually broke my first demonstration app.

I’ve had a lot of fun playing with Cordova, and I’ve barely scratched the surface of what it can do or be extended to do. This has to be the easiest way to get into smartphone application development. ■

Mike Diehl operates Diehlnet Communications, LLC, a small IP phone company. Mike lives in Blythewood, South Carolina, with his wife and four sons. He can be reached at mdiehl@diehlnet.com.



DOC SEARLS

On Infrastructure, Geology and Other Temporary Things

Linux is lasting. This can help inform our understanding of other things we depend on.

Infrastructure, like geology, is temporary. Both work on a LIFO basis. Dunes built by surf and wind are the first landforms to go when the weather gets wild—likewise human constructions, such as beach houses and boardwalks. As I write this, on Halloween 2012, the dunes and boardwalks of the Jersey Shore, where I spent my summers as a kid, have been torn apart and deposited inland by Superstorm (née Hurricane) Sandy, which came ashore two nights ago. A broken gas line feeds a fire that already has burned ten houses in Mantoloking. Nearly every clear summer day, from 1949 to 1961, when I was age 2 to 14, we'd go to Mantoloking Beach. That's where I

learned to swim and ride the surf. Since then, Mantoloking has become an upscale enclave. From what I can tell, all the houses that just burned were put up since I knew the place. Last in, first out.

Infrastructure is the geology we make and re-make for ourselves. Across eastern Massachusetts, where I am now, bike paths have replaced railroads that replaced the tow paths of canals that replaced paths of animals and other natives through the woods. Construction is the world's oldest and biggest business. Much of that work is re-construction. There is no urge more human, Stewart Brand once said, than the urge to alter a permanent structure. Using natural

Even for hackers, attention tends to be partial and provisional, and always subject to rot.

and manufactured materials, we build and re-build, constantly. On the island of Majorca, which is one big piece of marine limestone, it is said that every building block has been lifted by a thousand hands a thousand times. A single block holding up an olive terrace also may have served time in pavement, a lookout tower and the wall of a house. How that works also is a good model for code. A few years ago, on a *Linux Journal* Geek Cruise (I still miss those), I asked Andrew Morton if hackers would still be improving the Linux kernel 200 years from now. He said yes. To me, that makes Linux a form of infrastructure. Something of its nature is geological. That is, its purposes transcend the temporary, and even the specific. It's made for wide usability more than for any specific use. "I don't know what happens outside the kernel, and I don't much care", Linus says (<http://www.linuxjournal.com/article/6427>). "What happens inside the kernel I care about."

Given the nature of human attention, it's amazing that infrastructure gets made at all. Even

for hackers, attention tends to be partial and provisional, and always subject to rot. The sentences we hear verbatim are forgotten within seconds, leaving only meaning behind—and partial meaning at that. Each cycle of human life runs about a century at best. If we're lucky, we might get 60 productive years. Within those years, one's lasting effects are so rare that others treasure them. My own favorite treasure is the George Washington Bridge, which my father helped build, as a cable rigger (<http://www.flickr.com/photos/docsearls/3503894401>). He's long gone, but his work is not. Yet, it too shall pass. The purpose of the bridge—to carry traffic in and out of New York—surely will outlive all of us, but it is unlikely to outlast the geology in which it is anchored. On the New York side, the rock is Manhattan Schist (<http://www.washington-heights.us/history/archives/000457.html>), formed in the Cambrian, about a half-billion years ago. On the New Jersey side, it's the Palisades ([http://en.wikipedia.org/wiki/The_Palisades_\(Hudson_River\)](http://en.wikipedia.org/wiki/The_Palisades_(Hudson_River))):

cliffs that began as an intrusive sill at the end of the Triassic, a little more than 200 million years ago. That's when Pangea began to break up. If you want to see the adjacent geologies of that time, visit the Atlas Mountains of Morocco.

On the stellar scale, the rock flanking the Hudson is young. The solar system is 4.65 billion years old; the universe about 2.8 times older than that. Conveniently, the first half-dozen billion years of matter's existence were enough time to populate most of the periodic table. This required compressing light elements into stars and exploding those stars, over and over, scattering the building materials of new stars and planets in all directions. Most heavy elements involved in Earth's creation sank early to the core. The ones found in Earth's crust—gold, titanium and tungsten, for example—were deposited by meteorites (<http://www.sciencedaily.com/releases/2009/10/091018141608.htm>) after the surface hardened. Surely much of it came during the same Heavy Bombardment (http://en.wikipedia.org/wiki/Late_Heavy_Bombardment) that put a face on the moon, during a 300-million-year span, starting 4.1 billion years ago.

More than 90% of the iron mined so far on Earth was deposited as ferrous sludge on ocean floors a little more than two billion years ago, when life began to bloom and metabolize out the iron then saturated in the seas. This was an event that will not be repeated. If our species' appetite for iron persists long enough, we'll re-mine it from landfills after exhausting its supply in ancient rock. Helium, the second-lightest element and one of the most common in the universe, is currently produced on Earth only by decay of a certain breed of natural gas. At the current usage rate, helium is due to run out in a few dozen years (<http://www.washingtonpost.com/wp-dyn/content/article/2010/10/11/AR2010101104496.html>). As yet, we have no other way to make it, but that doesn't stop us from using it up, just as we use up other non-renewable things, deferring the problem of resource exhaustion to future generations. We are, undeniably, a pestilential species.

Our summer home was about ten miles inland from Mantoloking, in Brick Township, on the edge of the Pine Barrens ([http://en.wikipedia.org/wiki/Pine_Barrens_\(New_Jersey\)](http://en.wikipedia.org/wiki/Pine_Barrens_(New_Jersey))). My parents had an acre and a half there, which they bought in 1948

for \$150. Pop and Uncle Archie, his brother-in-law, cut a driveway to a clearing and brought in a small old shack on a flat-bed truck, which they deposited on a shallow foundation of cinder blocks. They named it The Wanigan, a native term for a portable abode. They drove a well by hand, pounding lengths of galvanized steel pipe down into the ground. Their driver was a capped iron pipe half filled with lead, weighing about 80 pounds. They fit this over the top of the well pipe, then lifted and dropped it, over and over again. After they got water flowing with a hand pump, they built a kitchen around the pump and then a one-hole privy in the back. A few years later Pop added a bedroom, dug a septic tank and built a small indoor bathroom. Electricity arrived early on, but telephony never did. Grandma and other relatives bought adjacent properties, and put up their own little houses. I still remember every foot of the paths between them. For my sister and me, plus countless cousins, it was paradise. The forest floor was a thick mass of blueberries, huckleberries and wintergreen, under a canopy of scrub oak and pitch pine. Clearings and paths were established by deer and other woodland creatures. We went barefoot all summer, running from one

secret place to another, grazing on berries like sheep on grass, and riding our bikes to the country store to buy candy and comic books, which we read and re-read on our bunk beds. Bedtime came when the whip-poor-will called (http://en.wikipedia.org/wiki/Eastern_Whip-poor-will), and we fell asleep to a hubbub of crickets and tree frogs.

The LIFO geology under The Wanigan was sand deposited in the Pliocene, when the whole coast was under water. The sand was easy to dig up, which we often did, to make castles, forts and other structures. Our hands and feet would always turn black when we did. Pop told us this was because countless wildfires destroyed the forest over and over, turning the sand gray with ash. Later, when I looked more deeply into the matter, I found he was right. In fact, the whole forest ecosystem was adapted to fire. Also, apparently, to suburban sprawl, since the Wanigan, Grandma's place and nearly everything around it is now a strip mall.

The main influence on my life in those days was my cousin Ron, who was five years older than me and much more hip to what mattered in the world, such as girls, cars and rock & roll. I was too shy and geeky

to recruit a girlfriend and wasn't old enough to drive, but I could sublimate my yearnings through music. My main source for that was WMCA, then New York's main Top 40 station. I loved to gawk at WMCA's transmitter when we passed it on the New Jersey Turnpike, on our way down to The Shore (pronounced "Da Shaw"). I saw WMCA's three-tower rig as the well from which all cool music came.

It was from this that I became obsessed with the mysteries of wireless. Why did some AM stations have one tower while others had more? Why did AM signals fade under bridges while FM ones didn't? What made signals at the bottom end of the AM band travel farther along the ground than those at the top end? What made the ionosphere reflective of AM and shortwave signals but not of FM and TV signals? Why did TV work best with a roof antenna while AM radio didn't?

At age 12, I got a ham radio license and began to build electronic stuff, sometimes on advice from engineers I met at the transmitters of New York's AM stations, nearly all of which stood, like WMCA's, in stinky swamps along the Hackensack and Passaic Rivers. I'd ride my bike down there from our house in Maywood, knocking on

doors of transmitter buildings, and then asking questions of the engineers while they took readings off meters, threw big scary switches and whacked vegetation away from the bases of towers. At night I'd "DX"—listening to far-away stations—on my Hammarlund HQ-129X ham receiver, which did a great job picking up AM signals, mostly through my 40-meter dipole antenna, which hung like a clothesline between my bedroom and a tree in our backyard. By the time we moved away from Maywood, I had logged about 800 stations, or about 10% of all the stations transmitting in the US at that time. Later, as an adult in North Carolina, I did the same with FM signals, which occasionally refract at a low angle, like a mirage above a hot road, off the same ionospheric layer that reflects AM signals, bringing in clear signals from 800 to 1,000 miles away. The thrill of this was less in signal fishing than in gaining an empirical understanding of how things work.

It was because of that understanding that I blogged this two days ago, while Sandy was headed ashore (<http://blogs.law.harvard.edu/doc/2012/10/29/riding-out-the-storm>):

Given the direction of the storm, and the concentrating effects

of the coastlines toward their convergence points, I would be very surprised if this doesn't put some or all of the following under at least some water:

- All three major airports: JFK, La Guardia and Newark.
- The New York Container Terminal.
- The tower bases of New York's AM radio stations. Most of them transmit from the New Jersey Meadows, because AM transmission works best on the most conductive ground, which is salt water. On AM, the whole tower radiates. That's why a station with its base under water won't stay on the air. At risk: WMCA/570, WSNR/620, WOR/710, WNYC-AM/820, WINS/1010, WEPN/1050, WBBR/1130, WLIB/1190, WADO/1280 and several others farther up the band. WFAN/660 and WCBS/880 share a tower on High Island in Long Island Sound by City Island, and I think are far enough above sea level. WMCA and WNYC share a three-tower rig standing in water next to Belleville Pike by the New Jersey Turnpike and will be the first at risk.

I got most of that right. Two of the three airports took water, and WMCA, WNYC, WINS, WLIB and some number of other stations went off the air when the tide surge rose over their transmission equipment and tower bases. The main reason I called them right is that I've been studying infrastructure in various ways ever since Ron turned me on to rock & roll. It also has occurred to me gradually, during the decade and a half I've been writing here, that my interest in Linux and infrastructure are of a single piece with my interest in geography, aerial photography and fault lines where the converging forces of business, hackery and policy meet. In the parlance of geologists, much of what happens amidst all of it is "not well understood".

According to the *Oxford English Dictionary*, the word *infrastructure* first appeared in the early 1900s (<http://oxforddictionaries.com/definition/english/infrastructure>). According to Google's Ngram Viewer (http://books.google.com/ngrams/graph?content=infrastructure&year_start=1800&year_end=2000&corpus=15&smoothing=3share=), which graphs usage of words in books across time, *infrastructure* did not hockey-stick until the 1960s.

Relatively speaking, academic work on infrastructure is sparse. There are few university departments devoted to infrastructure. (The University of Melbourne's Department of Infrastructure Engineering is one: <http://www.ie.unimelb.edu.au>). It's more of a topic in many fields than a field of study itself. I have come to believe this is a problem. If we'd had a better understanding of infrastructure, we might have suffered fewer losses from Katrina, Sandy and other natural disasters. We might also have a better understanding of why it's nuts to build on barrier dunes, fault lines and places nature likes to burn every few dozen years—unless the inevitability of loss is a conscious part of our deal with nature. Likewise, I believe we'd have a better understanding of human inventions with vast positive externalities, such as those produced by Linux and the Internet, if we also understood their leverage as the same as that provided by infrastructure.

Without that understanding, it's hard to make full sense of modern oddities, such as Google's giant data centers. A few days ago, as I write this, Google took the wraps off of what had been, until then, largely a secret matter. Now it's bragging on

the giant, nameless buildings that together comprise, Google says, "where the Internet lives". Google does its best to pretty up these places with gorgeous photos and copy like, "Steam rises above the cooling towers in The Dalles data center in Oregon. These plumes of water vapor create a quiet mist at dusk." Still, they look like prisons (<http://www.google.com/about/datacenters/gallery/#/places>)—or data-fired power plants, which is basically what they are.

Think about it. These places are no less central to what civilization does today than are power plants and container ports. Yet there is little if any regulatory oversight of them, outside concerns over power use and environmental impact. Nor could there be. Knowledge of what the Internet is, and how it works, is minimal to most of those who use it, and worse than absent in legislative and regulatory circles, both of which have long since been captured by Hollywood and the phone and cable companies. Because of this, the Net is slowly turning into a "service" through which Hollywood and the carriers can bill us in fine detail for "content" and data usage. And, because of this, we risk diminishing

or losing the most productive generator of positive economic externalities the world has ever known. Google is both at war and allied with these forces, and that war is being played out inside these data centers. I believe we'd understand all of this a lot better if we also had a better understanding of what infrastructure is, and what it does.

Sometimesoon I'll re-start work on *The Giant Zero*: my book about infrastructure and the Internet, and how the two overlap. The title comes from a description of the Net as "a hollow sphere in which every point is visible to every other point across an empty space in the middle—a vacuum where the virtual distances are zero". That insight is Craig Burton's, and he provided it in an interview I did for a *Linux Journal* piece in the mid-1990s. Learnings from Linux will be at the core of *The Giant Zero*. Anybody interested in helping build the prose base for that work, please get in touch. I can't, and won't, build it alone. ■

Doc Searls is Senior Editor of *Linux Journal*. He is also a fellow with the Berkman Center for Internet and Society at Harvard University and the Center for Information Technology and Society at UC Santa Barbara.

Advertiser Index

Thank you as always for supporting our advertisers by buying their products!

ADVERTISER	URL	PAGE #
1&1	http://www.1and1.com	21
CONFOO	http://Confoo.ca	101
EMAC, INC.	http://www.emacinc.com	29
EMPEROR LINUX	http://www.emperorlinux.com	61
IXSYSTEMS	http://www.ixsystems.com	7
MICROWAY	http://www.microway.com	54, 55
SCALE	https://www.socallinuxexpo.org/scale11x/	2
SHAREPOINT	http://www.sptechcon.com/	89
SILICON MECHANICS	http://www.siliconmechanics.com	3

ATTENTION ADVERTISERS

The *Linux Journal* brand's following has grown to a monthly readership nearly one million strong. Encompassing the magazine, Web site, newsletters and much more, *Linux Journal* offers the ideal content environment to help you reach your marketing objectives. For more information, please visit <http://www.linuxjournal.com/advertising>.