A PENGUIN-POWERED RADIO STATION IN DC

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community

WATCH: ISSUE OVERVIEW

JULY 2016 | ISSUE 267
http://www.linuxjournal.com

ANDROID BROWSER SECURITY

What You Should Know

+

A Crash Course on **Planning Security Exercises**

Delve Into **Complex String Processing**

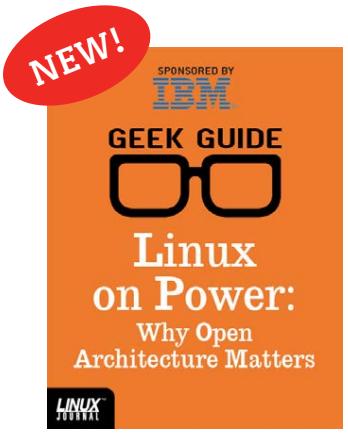
How to Set Up **WordPress with nginx**

Turn an Old PC into a **Virtual-Machine Host**

Machine Learning with Python

Author:
Reuven M. Lerner

Sponsor:
Intel



Linux on Power: Why Open Architecture Matters

Author:
Ted Schmidt

Sponsor:
IBM

Hybrid Cloud Security with z Systems

Author:
Petros Koutoupis

Sponsor:
IBM

LinuxONE: the Ubuntu Monster

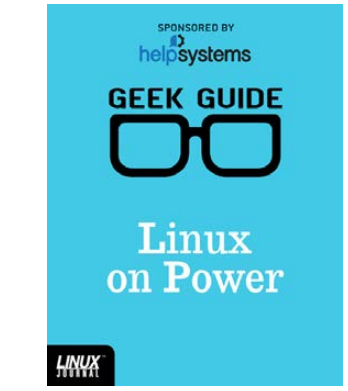
Author:
John S. Tonello

Sponsor:
IBM

Ceph: Open-Source SDS

Author:
Ted Schmidt

Sponsor:
SUSE



Linux on Power

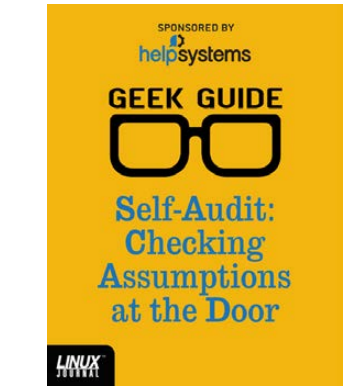
Author:
Ted Schmidt

Sponsor:
HelpSystems

SSH: a Modern Lock for Your Server?

Author:
Federico Kereki

Sponsor:
Fox Technologies



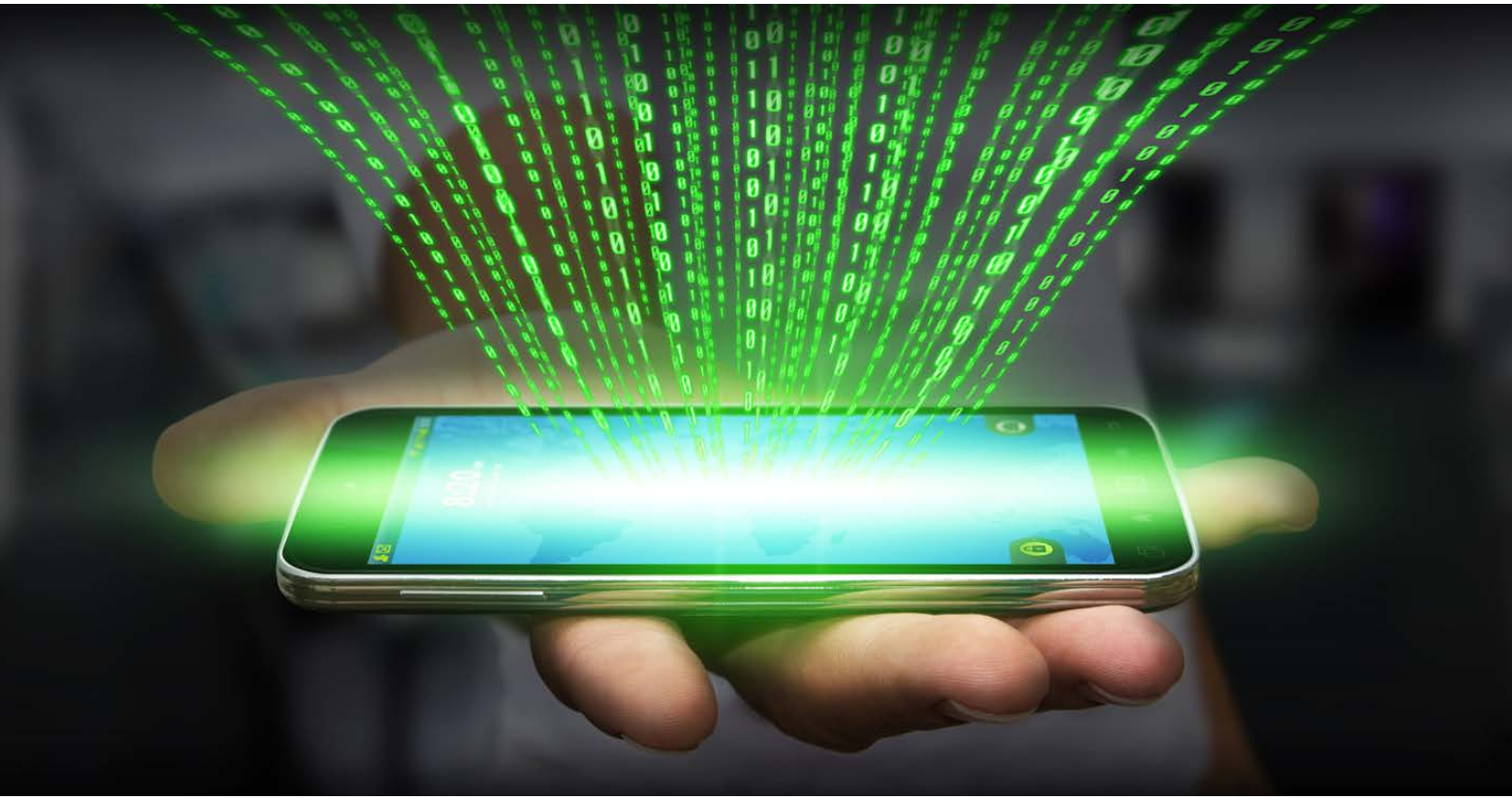
Self-Audit: Checking Assumptions at the Door

Author:
Greg Bledsoe

Sponsor:
HelpSystems

CONTENTS

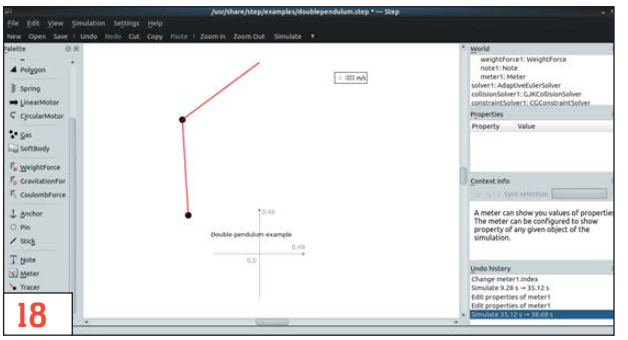
JULY 2016
ISSUE 267

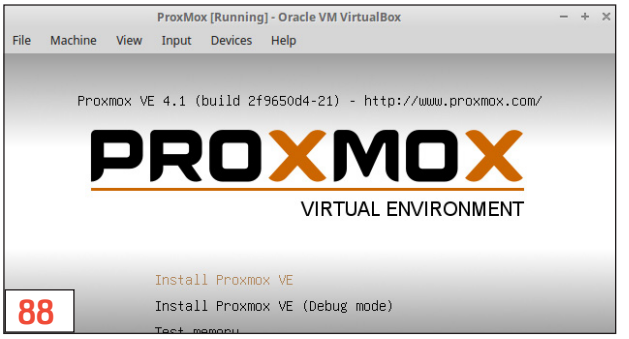


FEATURES

COLUMNS

IN EVERY ISSUE





ON THE COVER

- **A Penguin-Powered Radio Station in DC, p. 80**
- **Android Browser Security—What You Should Know, p. 68**
- **Turn an Old PC into a Virtual-Machine Host, p. 88**
- **A Crash Course on Planning Security Exercises, p. 42**
- **Delve into Complex String Processing, p. 36**
- **How to Set Up WordPress with nginx, p. 26**

LINUX JOURNAL (ISSN 1075-3583) is published monthly by Belltown Media, Inc., PO Box 980985, Houston, TX 77098 USA.
Subscription rate is $29.50/year. Subscriptions start with the next issue.

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 • Adam Monsen

President	Carlie Fairchild publisher@linuxjournal.com
Publisher	Mark Irgang mark@linuxjournal.com
Associate Publisher	John Grogan john@linuxjournal.com
Director of Digital Experience	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



Sharpen your Android skills at

AnDevCon

The Android Developer Conference

World's Largest

BOSTON

August 1-4, 2016

Sheraton Boston

"Simply the best Android developer conference out there! A must-go if you do Android development."

—Florian Krauthan, Software Developer, Hyperwallet

Get the best Android developer training anywhere!

- Choose from more than 75 classes and in-depth tutorials
- Meet Google and Google Development Experts
- Network with speakers and other Android developers
- Check out more than 50 third-party vendors
- Women in Android Luncheon
- Panels and keynotes
- Receptions, ice cream, prizes and more!

www.AnDevCon.com

A BZ Media Event

Better Than We Found It

SHAWN POWERS

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.

Technology is supposed to make our lives easier. It's supposed to automate those things that take our valuable time, and it promises to make those things we still do faster and more efficient. Unfortunately though, it hasn't given us more free time, it's just allowed us to cram even more activities into our already busy schedules. Thankfully, some technology really does make our lives better. If we're willing to learn and change, technology really can be that invaluable aid in our lives. The trick is to make technology work for us, and not the other way around.

For example, in last month's issue I wrote about the Raspberry Pi IP camera I created in order to live-stream my bird feeder. Since then, I've learned about a new tool called UV4L that makes my Raspberry Pi an even better camera. Learning about new technology and new ways to use existing technology is the secret to getting the most out of our increasingly digital world.

Reuven M. Lerner describes a better way to host WordPress installs using nginx. Building on last month's introduction to nginx, Reuven shows how scalable it can be in a real-world scenario. Dave Taylor follows with a fascinating look at how to automate the creation of content. Using "spinning", Dave shows how to create content that can be used for evil, but is

VIDEO: Shawn Powers runs through the latest issue.

also brilliant in its execution. If you've ever wanted to peer into the mind of a spammer, Dave will introduce you to the robotic version of one.

We can never get "good enough" at security to be fully secure, but with the help of Susan Sons, we can do our best to be security-focused and prepared. Much like a school has fire drills, Susan explains the concept (and plan for realization) of running security exercises. Having a staff that is not only aware, but also practiced at dealing with security can be invaluable. Susan gives us a very practical plan for making that happen.

Charles Fisher provides a depressing, but very important look at Android browser security—or more specifically, the lack of security in the Android WebKit browsers. There are so many programs utilizing the Android WebKit libraries that a vulnerability at that level can have security ramifications far beyond the default stock browser. Charles not only explains the problem, but also gives some concrete information on how to deal with the potential security nightmare still in many Android systems.

Alan Peterson explores a better way to do radio. Linux isn't the default for most radio stations, but in Washington, DC, that's changing. Thanks to the power of open-source software and the flexibility of coding on Linux, several radio stations are finding Linux is the answer for secure, scalable management of radio, especially in this increasingly internet-centric world. If you're interested in how radio stations are automating using Linux as their core, you won't want to miss his article!

Finally, John S. Tonello continues his three-part series on making the internet—or more specifically, on creating a tiny virtualized environment for testing internet tools efficiently and effectively. Using VirtualBox and Proxmox, John demonstrates how easy and powerful a virtualized environment truly can be. If you're tired of doing your simulations on piles of old computers, this is a series you'll really enjoy.

We all know Linux makes the world a better place. Whether you're talking about its presence in the mobile space, its dominance in the cloud or just its efficiency in a server room, Linux is invaluable. In this issue, we explore lots of ways Linux and open source are making our technology more and more beneficial, and along the way, we get to learn cool new things. We hope you enjoy this issue as much as we've enjoyed putting it together!■

RETURN TO CONTENTS

PREVIOUS
Current_Issue.tar.gz

NEXT
Editors' Choice

diff -u

What's New in Kernel Development

Stephan Müeller has been working on updating the code that implements **/dev/random**. As new technology has become popular, such as solid-state drives, fully virtualized systems and highly parallelized systems, the ability to find enough entropy to produce truly random numbers has declined.

Good random number generation is crucial on modern systems in an era where security exploits are commonplace and governments throughout the world often are the ones wearing the black hats. It's also important for the Linux kernel to guarantee a sufficient source of random numbers on all systems, regardless of any particular hardware configuration. This can prove complex to implement, as the kind of entropy available on a given system is often related to the kind of hardware available to it.

Stephan implemented **LRNG** (Linux Random Number Generator) to address the existing problems associated with /dev/random. The big goals were to provide a good source of entropy even during boot-up and to reduce entropy-related slowdowns on parallelized systems, where security measures must be implemented across the full set of CPUs, whether physical or virtual.

There were a number of objections to Stephan's code, none of which are likely to keep it out of the kernel. The reason is that the old mechanism for generating random numbers is out of date, so any incremental improvement will be better than the status quo. Also, LRNG is intended to exist side by side with /dev/random for the present, so no features will be overtly lost.

But, for example, **Theodore Ts'o** objected that some of Stephan's entropy sources weren't providing true entropy and should be removed from the calculation. Stephan was happy to comply.

Another objection was that **glibc** might not export the **getrandom()** system call, as it was a Linux-specific call and wouldn't exist on operating systems like the **Hurd**. In which case, Ted said, there might have to be a special **liblinux** library alongside glibc, that could catch all the Linux-specific entries.

Just as an aside, it's so cool that glibc is taking account of projects like the Hurd and other operating systems. Now that Linux runs the world (seriously—it does), it would be tempting for glibc and other projects to cater only to Linux. But no! We still see similar struggles as were going on in 1995 and thereabouts. It's like the perennial **C compiler** debate: "The compiler should produce *this* machine code!" "No, the kernel should use *this* source construction!" "That source construction sucks! It doesn't work with other compilers!" "*What* other compilers! If it's a problem, use #ifdef!" "We're trying to get *away* from #ifdefs, not pepper the code

with them!" "Well, la dee da!" "La dee da yourself!" And so on.

A prime attack vector against any operating system is the concept of knowing where in RAM a particular data structure or code sequence resided. It won't necessarily give you a full security exploit on its own, but it can combine with other security holes to trick a system into thinking it's doing something secure, when really control has been handed over to Dr. Evil.

The solution is to make sure that no one ever can tell what's in a given portion of **RAM**. But, that's tricky. The kernel itself has to know where everything is, so there has to be some rhyme and reason to the RAM layout, but it has to be a rhyme and reason that user code can't figure out.

Thomas Garnier recently came out with **ASLR** (Address Space Layout Randomization) for **x86-64** systems. It's essentially an enhancement to security features that have been in the kernel since 2005. Security-centric Linux distributions have been rolling their own enhancement patches for a while now, and Thomas wanted to bring that level of security to the official kernel.

No one had any major objections, so it looks like the code will go forward into the official tree. This won't eliminate security problems on Linux, but it'll greatly strengthen Linux's defenses against certain attack vectors.

These are rough days for Linux security. Linux essentially runs everything in the whole world except consumer desktops. It's a huge target. All the nations of the world are engaged in massive amounts of cyber warfare against each other and in some cases against their own citizens. Nongovernmental hactivist groups also derive a significant portion of their power and voice from Linux security exploits. And, the entire corpus of source code is freely available for everyone to pore over in search of the elusive zero-day exploit.

—Zack Brown

Android Candy: Waze Redux

Back in 2014, I highlighted Waze, which is a turn-by-turn GPS navigation program created by a startup in Israel. That company was bought by Google, but it still remains independent, at least for now. (It does share some data behind the scenes, but it functions differently when it comes to routing.)

(Image from https://www.waze.com)

Although I had very bad luck with Waze early on, I recently used it on a cross-country trip, and it was amazing. I still have unpleasant memories of Waze trying to force me to turn off an overpass and having it rout me to an off-ramp only to get back on the on-ramp immediately. I'm happy to say, however, those issues seem to be resolved. In fact, it was a very pleasant experience!

Not only was the navigation reliable (and murder-free), but it has a unique way of saving time by taking less-traveled routes. Last year when I was driving through Atlanta, Georgia, I got stuck in traffic for hours using my Garmin GPS. This year, Waze took me into corn fields in order to avoid traffic jams in Nashville. I'll admit, I was a bit worried when GPS advised me to turn on a poorly maintained country road, but in the end, it saved me hours of monotonous city traffic.

The TL;DR truth is, Waze has gotten to the point where it's now my favorite GPS app. Plus, if your passenger is bored, it's fun to report speed traps and hazards on the road. All that input makes for better driving, which makes family vacations far more enjoyable! Check it out in the Google Play store today. Waze is still free, and you'd be silly not to give it a test drive.—Shawn Powers

Message for You, Sir!

In my Open-Source Classroom column in the May 2016 issue, I discussed how to set up Gmail as your SMTP provider for outgoing email. The problem with email is that sometimes the sheer quantity of it makes important messages slip past my radar. So for really important error messages, I like to get SMS messages. Thankfully, I get several orders of magnitude fewer text messages than I do email messages. That means if a text comes, I almost always notice, and I always check it. ("Inbox Zero" is far more attainable with my SMS inbox!)

Most providers have an email gateway for sending text messages, but figuring out what format to use for what number is frustrating. I prefer a simple way to add a one-liner into a script that will send a text message. Thankfully, Ian Webster provides TextBelt free of charge! You can download the source code and host it yourself if you prefer, but Ian graciously offers a running instance of TextBelt that is free as in beer and free as in speech

to use. To send a text, simply use `curl`:

```
curl -X POST http://textbelt.com/text -d number=5551234567 -d
 ➥'message=Warning! Your server is on fire! Also,
 ➥you are out of milk.'
```

The code above will send a text message to any of the major carriers in the US. You'll get a response on the command line showing either success or failure. There's also an international gateway that is available; see http://textbelt.com for more information.

Note: there are a still a few goofy carriers. My Android phone is on Cricket Wireless, and although I do get a success message, the text never arrives. I suspect this is because Cricket recently changed its email gateway; hopefully the TextBelt code will be updated shortly. The moral of the story is, test first before relying on the service to work! —Shawn Powers

A Better Raspberry Pi Streaming Solution

In last month's issue (June 2016), I described my Raspberry Pi outdoor camera build. Since then, however, I've discovered a different way to stream video from it. Although capturing images with "raspistill" and serving them out via Web server is perfectly fine, I'd prefer to have an actual video stream coming from the little RPi. Thankfully, there's UV4L.

I had to add a line to my sources.list file in order to download the software, but it was well worth it. Add this to your /etc/sources.list file:

```
deb http://www.linux-projects.org/listing/uv4l_repo/raspbian/
 ➥wheezy main
```

And then you'll need to add the key:

```
curl http://www.linux-projects.org/listing/uv4l_repo/lrkey.asc
 ➥| sudo apt-key add -
```

Once that is done, simply install the program:

```
sudo apt-get update && sudo apt-get install uv4l uv4l-raspicam \
uv4l-raspicam-extras uv4l-server uv4l-uvc uv4l-xscreen \
uv4l-mjpegstream uv4l-dummy
```

And finally, you can add a line like this to your crontab that will turn your RPi into a streaming IP camera! These are just the settings

I use; check out the man pages for your options:

```
@reboot /usr/bin/uv4l -nopreview --auto-video_nr --driver
 ➥raspicam --encoding jpeg --quality 90 --metering matrix
 ➥--drc low --width 1280 --height 720 --framerate 10
 ➥--server-option '--port=9090' --server-option
 ➥'--max-queued-connections=10' --server-option
 ➥'--max-streams=5' --server-option '--max-threads=15'
```

Browse to http://raspberry.pi.ipaddress:9090 to see the various things UV4L provides.—Shawn Powers

Stepping into Science

In past articles, I've looked at several libraries or specialist applications that can be used to model some physical process or another. Sometimes though you want to be able to model several different processes at the same time and in an interactive mode. This is especially helpful in educational situations where you are trying to learn how those processes work. So in this article, I introduce an application named Step from the Edu section of the KDE Project (https://edu.kde.org/step).

The one major limitation is that the simulation runs only in two dimensions. Aside from that, you can model almost any system you can imagine. You can create discrete systems that are made of particles connected either with rigid rods or springs. You can apply external gravitational or electrical forces to your system. There is a molecular dynamics portion that allows you to model gases and liquids, including condensation and evaporation, and there is support for units and error values in your numbers. Several solvers are available to handle the actual calculations, so you should be able to find one that is best for your particular application.

To install Step, you should have a package available within your distribution's package management system. For example, with

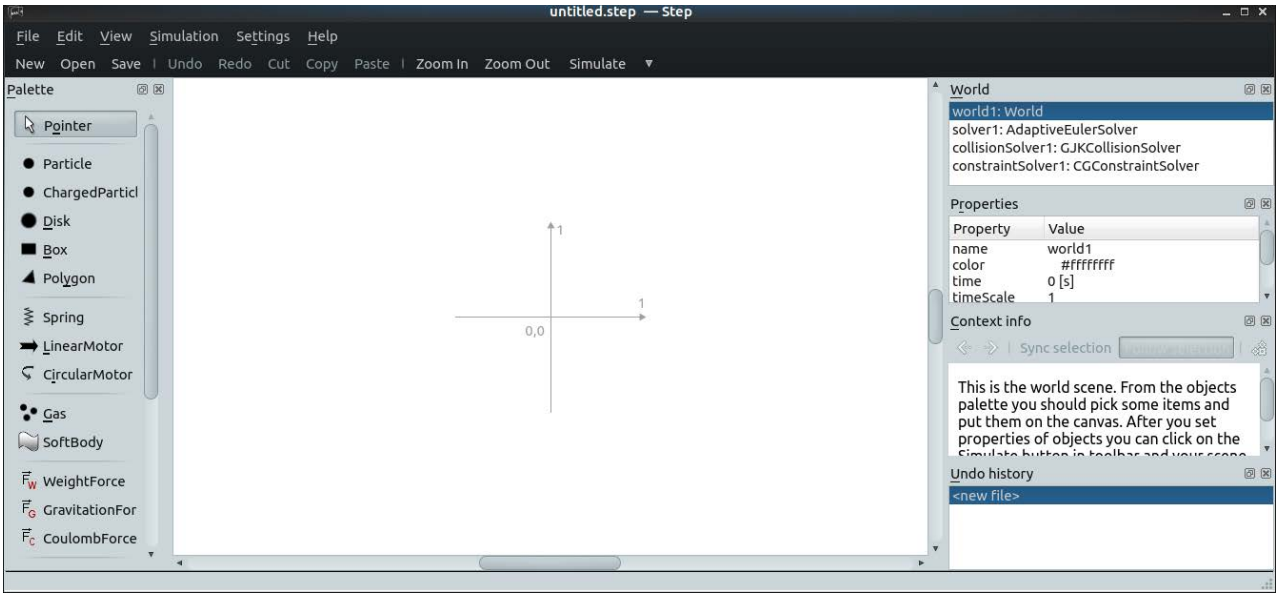


Figure 1. When you start Step, you get a new blank project to start your simulation.

Debian-based distributions, you can install Step with this command:

```
sudo apt-get install step
```

One thing to be aware of is that Step is part of the KDE Project. So, if you install it on a different desktop environment, such as GNOME or Unity, you also will need to install a large number of KDE support libraries. It will run fine on other desktop environments, so you don't actually need to run KDE.

To start Step, you either can find it within your desktop's menu system or open a terminal window and run the `step` command. When it starts up, you'll get a new, empty project in which to build your simulation.

In the center of your window is the main pane where your system of particles and forces is displayed. On the left-hand side is a palette of elements you can use to build the system you want to model. On the right-hand side, you can see a series of panes that give information about the system as a whole, along with details about specific selected elements.

It might be rather daunting to look at all of this functionality and

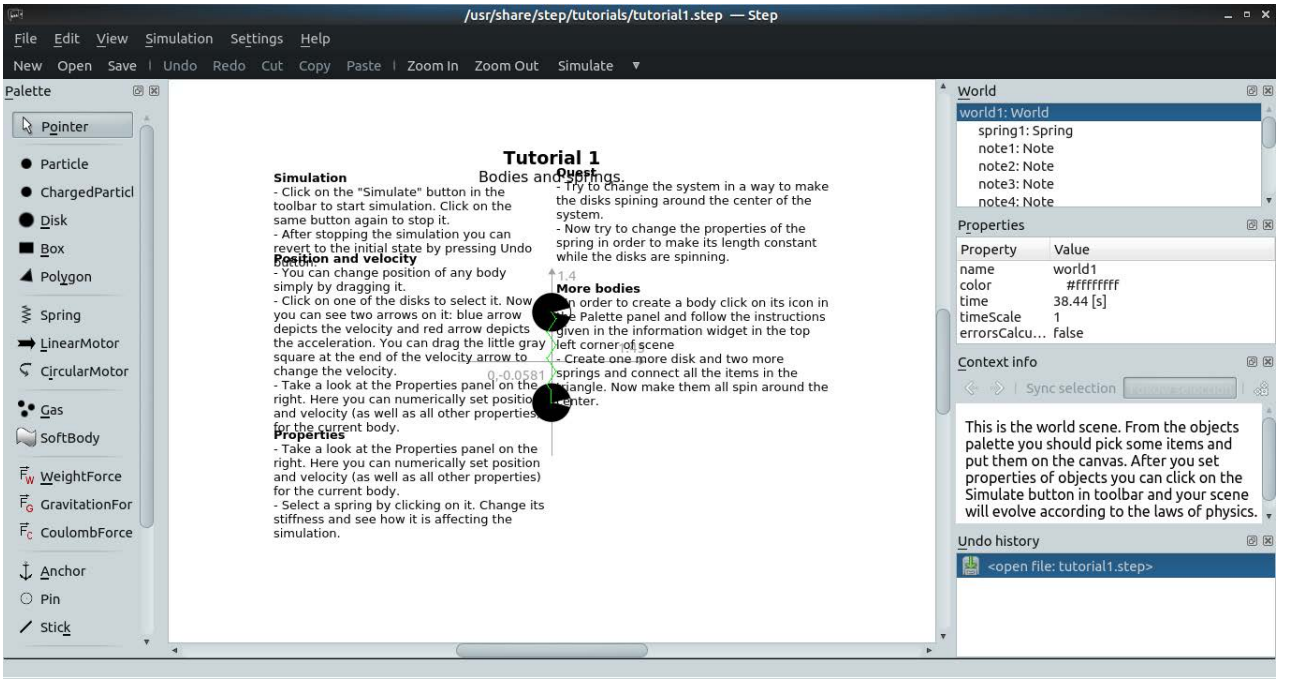


Figure 2. Step comes with a set of tutorials to walk you through some of the available functionality.

have to start with a completely blank canvas. Luckily, Step comes with a number of tutorials to walk you through the first steps of building and simulating systems within Step. You can access them by clicking the menu item File→Open Tutorial. This pops up a dialog window where you can select and load one of the five provided tutorials. When you select one, you get a system of elements along with a description of activities that you can follow along with to help you learn a bit more about each of Step's sections.

When you want to move on to developing your own models, several example projects are available that cover a large number of physical systems. You can access them from the File→Examples→Open Example menu item. This will pop open a dialog where you can select and load systems, such as the double pendulum model.

These examples provide a starting point that you can alter to create your own model. In order to make changes, you either can select the element in question within the main pane or you can select from the top pane on the right-hand side. Either of these steps will populate the middle pane on the right-hand side with the details for that element. Then you can go ahead and make your alterations. For example, for a particle, you can change the position, velocity or mass. Once you have your own system built, you can save it as a Step file by clicking File→Save As.

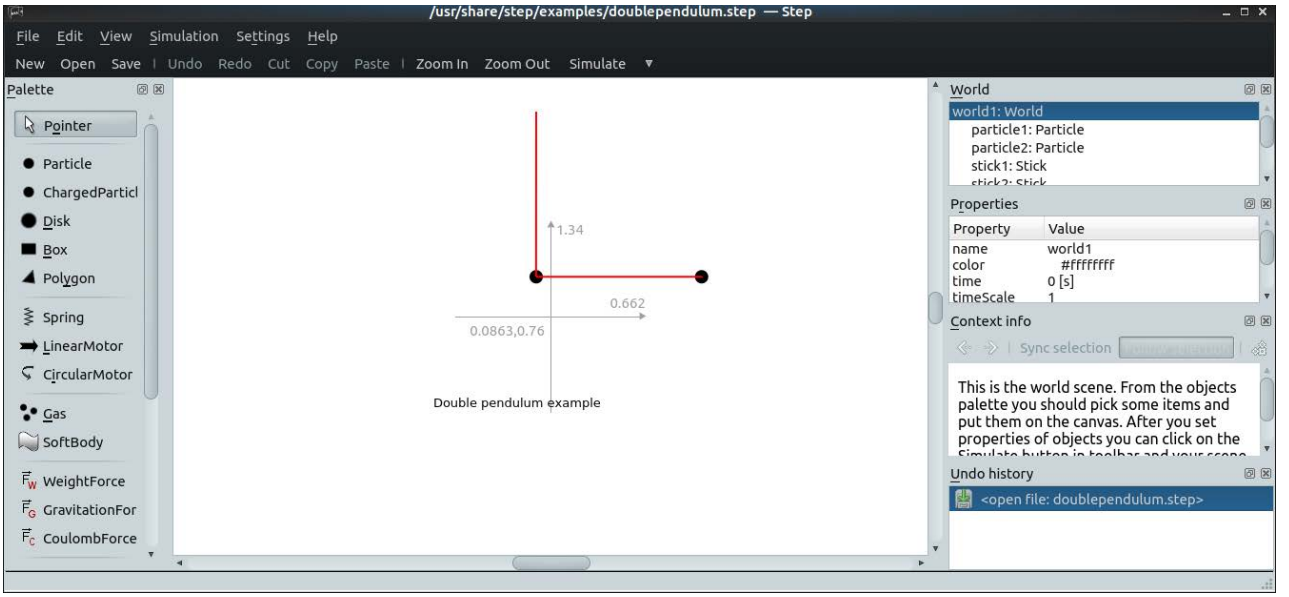


Figure 3. Step comes with a number of example projects, such as the double pendulum.

All I've described to this point is setting up the model of the physical system. I've yet to cover any simulating of the physical processes. To start the simulation, click on the Simulate button at the top of the window (just below the menu bar). You should notice a button next to that (with a down arrow on it) that lets you set the speed of the simulation. This lets you see your system in motion, which can be very illuminating in trying to understand how your system will behave.

Sometimes, however, this isn't good enough. You may need to have numbers tracking what various elements are doing within the simulation. Luckily, Step provides three measurement objects: meters, graphs and tracers. As an example, let's add a meter to monitor the speed of the first particle in the double pendulum example.

Start by clicking the meter option in the object palette. Step then asks you to select where this meter will be displayed. Once it's created, you still need to configure it so that it's monitoring a certain property of some object. You either can right-click on the meter and select Configure meter, or you can select the meter object from the top pane on the right-hand side.

In the example given in the screenshot, I chose the velocity of

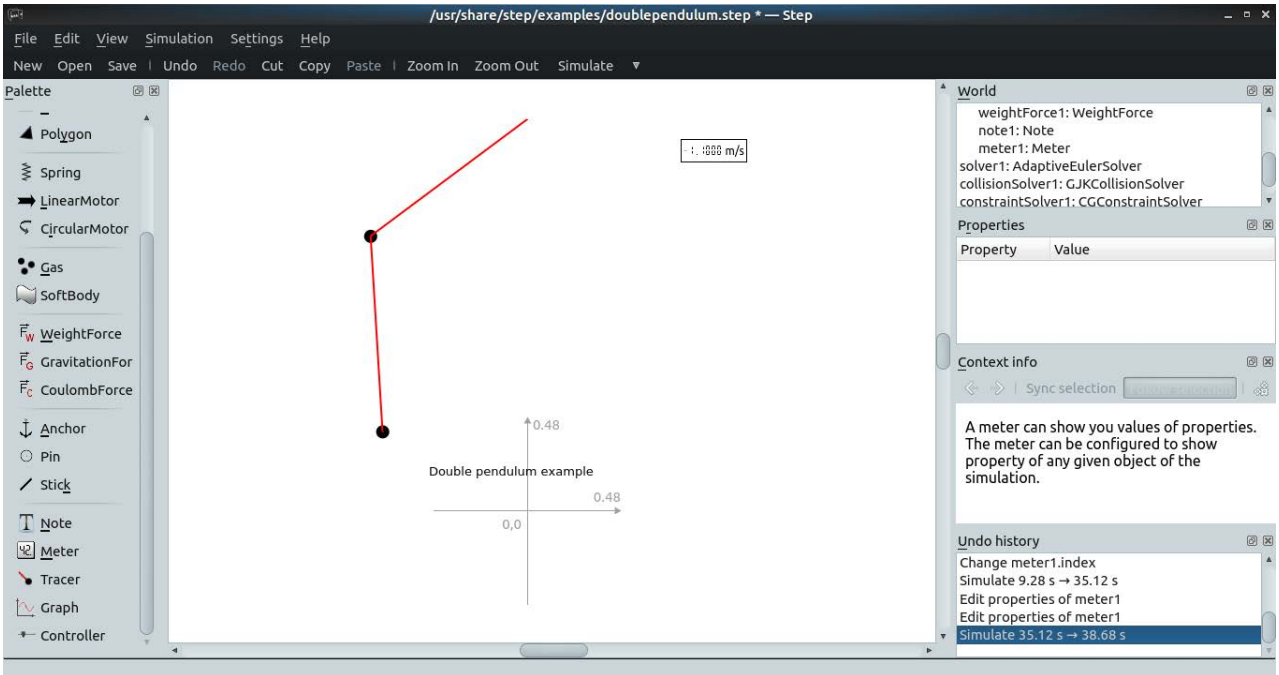


Figure 4. You can add elements, like meters, to track the behavior of elements in your simulation.

the particle1 object. When you first do this, you may notice that the meter isn't actually displaying anything. Don't forget that you also need to select the index of the property. In this case, you would need to change it from the default of 0 to 1.

Once you start creating your own systems, you may decide that the default solver isn't appropriate. Each solver is better or worse, depending on the details and constraints for your system. The details of the numerical analysis involved are too much to cover for such a short article, but you should know that you do have some control over this. You can select the solver1 object from the top pane on the right-hand side. The middle properties pane lets you select the solver type from among the 16 available solvers. You should be able to select one, along with its other properties, that is appropriate for your simulation.

I've covered only the most basic elements available within Step here, but hopefully you've seen enough to spark your interest in playing with it further—especially if you need a good tool to teach these types of physical processes to your students. They'll get a much more intuitive feel for them when they can play with the relevant parameters and see what effects they have.—Joey Bernard

THEY SAID IT

Love all, trust a few. Do wrong to none.
—William Shakespeare

I'd rather look ridiculous when everybody else does than plain and sensible all by myself.
—L. M. Montgomery

We don't receive wisdom; we must discover it for ourselves after a journey that no one can take for us or spare us.
—Marcel Proust

The only people who can change the world are people who want to. And not everybody does.
—Hugh Macleod

Never let the fear of failure be an excuse for not trying. Society tells us that to fail is the most terrible thing in the world, but I know it isn't. Failure is part of what makes us human.
—Amber Deckers

PREVIOUS
UpFront

NEXT
Reuven M. Lerner's
At the Forge

Non-Linux FOSS: Scripts in Your Menu Bar!

There are hundreds of applications for OS X that place information in the menu bar. Usually, I can find one that almost does what I want, but not quite. Thankfully I found BitBar, which is an open-source project that allows you to write scripts and have their output refreshed and put on the menu bar.

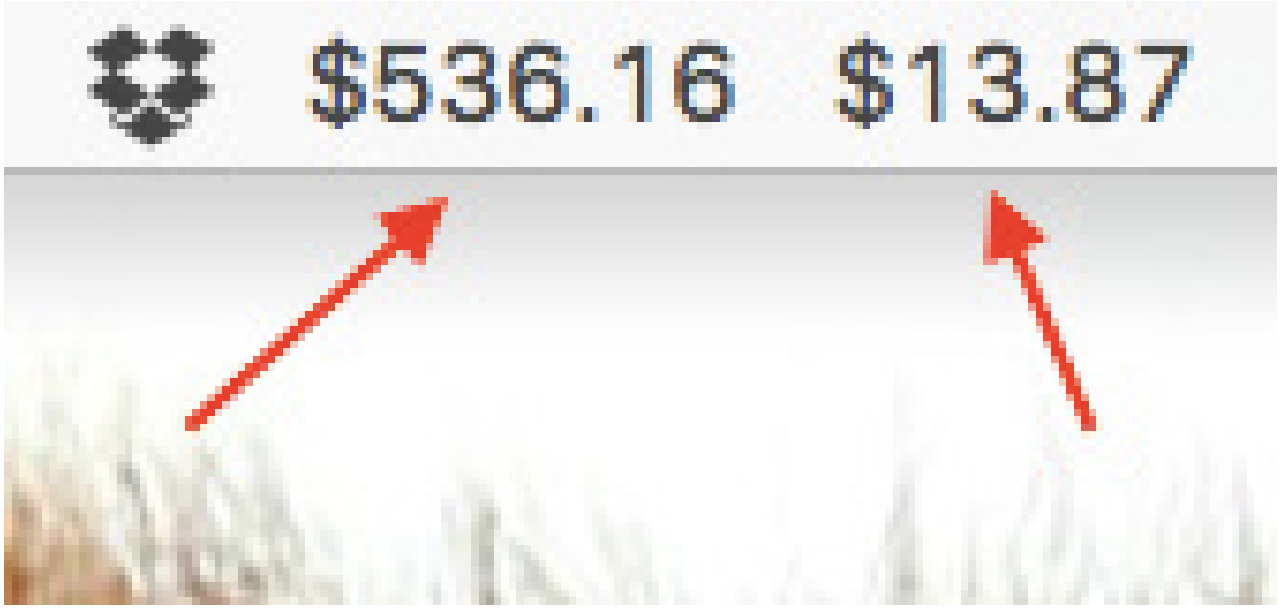


Figure 1. Bitcoin Price-Fetching Script

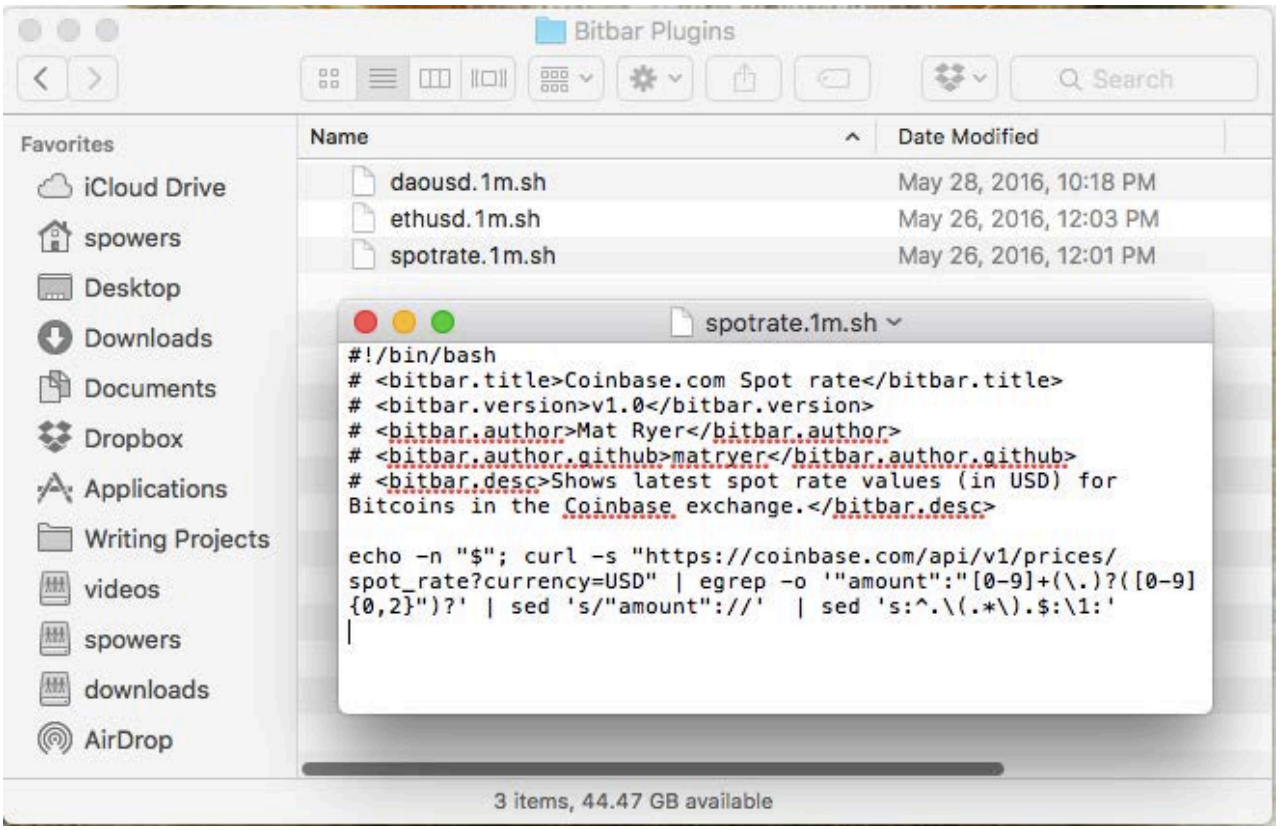


Figure 2. File Format for Additional Scripts

I personally use it to fetch Bitcoin and Ethereum prices, but because you're limited only by what you can get Bash to output, it's extremely flexible. Even the method by which you change update frequency is elegant. By adding a refresh rate to the name of your script, the program refreshes only as often as you desire. You can see an example of my Bitcoin price-fetching script in the screenshot (Figure 1). Also notice the file format for my additional scripts (Figure 2).

If you head over to http://getbitbar.com, you can download the binary or the source code. There is also a huge library of user-contributed scripts so you don't have to start from scratch. My Bitcoin script is actually from this repository, and I based my other scripts off that. Whether you want to pull text from an API like me or possibly `grep` the temperature from a weather page, BitBar is simple and elegant at the same time. In fact, BitBar is so useful and well designed, we're giving it the Editors' Choice award for this month.—Shawn Powers

RETURN TO CONTENTS

nginx and WordPress

How to make your WordPress installation highly scalable using nginx.

REUVEN M. LERNER

Reuven M. Lerner offers training in Python, Git and PostgreSQL to companies around the world. He blogs at http://blog.lerner.co.il, tweets at @reuvenmlerner and curates http://DailyTechVideo.com. Reuven lives in Modi'in, Israel, with his wife and three children.

PREVIOUS
Editors' Choice

NEXT
Dave Taylor's
Work the Shell

IN MY LAST ARTICLE, I took an initial look at nginx, the high-performance open-source HTTP that uses a single process and a single thread to service a large number of requests. nginx was designed for speed and scalability, as opposed to Apache, which was designed to maximize flexibility and configuration. But through the years, nginx has become increasingly flexible as well, with a growing number of plugins and modules that can be used to customize its configuration. Between the performance, increasingly good documentation and convenience, it's no wonder nginx has been increasingly popular.

It's also no surprise that WordPress, the open-source blogging and CMS platform, has become hugely popular. I've heard people say that 10% of websites are now run using WordPress. Even if that's not precisely true, there's no doubt that a huge number

of sites are powered by WordPress. I'm a mostly satisfied WordPress user, having converted my main site and my two ebook sites to it in the past year after years of using it to power my blog.

So, I thought it would be interesting to demonstrate how easy it is to set up WordPress with nginx, given the popularity of each of these systems alone as well as together. In my last article, I described how you can set up a plain-vanilla PHP system with nginx; WordPress is a bit more complex, but less than you might think. Starting with a bare-bones Linux installation, let's walk through the configuration needed to get WordPress up and running.

The Basics

In order to install WordPress and nginx together, you're going to need three basic software systems installed: WordPress, nginx and MySQL. The first two are pretty obvious, given this article's goal; the third is a byproduct of using WordPress, which works exclusively with MySQL.

So, on my Ubuntu Linux machine, I would run the following:

```
$ sudo apt-get install mysql-server mysql-client nginx-core
 ➥php5-cli php5-fpm php5-mysql
```

This installs a very large number of packages, but it will give you the core of what you need to get your system up and running. Notice that you're not installing WordPress here, so that you can install it manually, using the source code. Indeed, installing WordPress via `apt-get` also means installing Apache; although it's certainly possible to undo this choice, the benefits of installing WordPress on your own outweigh those of doing it via a package manager.

You will, as part of this installation, need to choose a password for your MySQL root user. This is an important part of security on your system, so do try to use a strong password.

Once the package manager completes the installation of the above packages, you'll have a working nginx installation. Try it; you can point your browser at your server's port 80, and you should get the default nginx page indicating that it installed correctly.

Installing WordPress

Installing WordPress is quite straightforward; the complex part will be

hooking together nginx with FPM, the PHP version of FastCGI. As you saw if you read my last article (in the June 2016 issue), FPM is the method through which nginx can run PHP in a separate process, without bloating the entire nginx process or reducing performance by very much.

The default location for HTML files in my nginx configuration is /usr/share/nginx/html. Within that directory, there's an index.html file, whose contents provide the default "welcome" page to nginx that you saw earlier.

The thing is, it's probably easiest just to install WordPress in a separate directory. So, I download WordPress and open it up under /usr/share/nginx/wordpress, which is a directory that'll be created anyway, when I open the tarfile. Here's what I did:

```
$ cd /usr/share/nginx
$ wget https://wordpress.org/latest.tar.gz
$ tar -zvxf latest.tar.gz
```

Now that WordPress has been installed, you'll want to run it. But you can't do that until you have created a MySQL database, since part of the WordPress installation requires that your database be working and ready.

So, let's create a new MySQL database! There are several ways to do it. I typically prefer to use the `mysqladmin` program, which takes similar options to the MySQL client, including `-u` to indicate which user you want to use and `-p` to indicate that you want to enter a password. Both will be necessary:

```
$ mysqladmin create wordpress -u root -p
```

Note that when I say you want to use the "root" user here, I'm not referring to the UNIX-level root user. Rather, I'm talking about the MySQL "root" user, which has ultimate privileges on the database. When you installed MySQL earlier, you needed to choose a root password. It's this password that you must enter when prompted, thanks to the `-p` option above.

You can check that your database was created by entering MySQL as root (once again, with `-p` and after entering the root password):

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

Then, issue the command `SHOW DATABASES` at the `mysql>` prompt. On my completely new system, I got the following response:

```
mysql> SHOW DATABASES;

+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| wordpress          |
+--------------------+
4 rows in set (0.01 sec)
```

Notice that there are three databases in the system in addition to the "wordpress" database I created earlier. These are used internally by MySQL. Indeed, you'll now connect from the UNIX shell to the "mysql" database, which is used to run your database:

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

If you prefer, you also can switch to the "mysql" database from within the MySQL client:

```
mysql> \u mysql
```

Either way, you should now be connected to the "mysql" database as root. Next, you'll create a "wordpress" user and then allow that user to connect to your MySQL "wordpress" database.

I should note that when I work with consulting clients, it's not unusual for them to use the "root" user for all of their database connections. After all, it's more convenient, right? However, this is almost always a bad idea; you really want to have and use a separate user name for security reasons.

Once connected, you'll create a user, assign it a password and indicate

that this new user has full privileges on the "wordpress" database:

```
mysql> CREATE USER wordpress@localhost;
mysql> SET PASSWORD FOR wordpress@localhost = PASSWORD('my-wp-pw');
mysql> GRANT ALL PRIVILEGES ON wordpress.* TO  wordpress@localhost;
mysql> FLUSH PRIVILEGES;
```

Note that SQL commands are case-insensitive, so you don't need to use CAPITAL LETTERS when entering them. However, I've done so for years, following the advice of Joe Celko's *SQL For Smarties* books, and I've found that it helps to distinguish between other parts of my programs.

Also note that in the above scenario, you've created a "wordpress" database and a "wordpress" user. Actually, your user isn't named "wordpress" so much as "wordpress@localhost"; when connecting to MySQL, the hostname is taken into account.

Finally, the `FLUSH PRIVILEGES` command is necessary to tell MySQL that it should take the new privileges into account even without doing a restart of the database server.

Once this is in place, you'll want to test it to make sure you can connect to the "wordpress" database as the "wordpress" user. On the UNIX shell, type:

```
$ mysql wordpress -u wordpress -p
```

When prompted for the password, enter the password you used (which is hopefully not "my-wp-pw" from above). You should see the "welcome" message and a `mysql>` prompt. If that doesn't work, then double-check the user name and password that you created, and make sure you flushed the privileges.

Now that you know your configuration works, you'll set up your WordPress configuration in a file called wp-config.php. This file is in the directory /usr/share/nginx/wordpress, thanks to opening the WordPress tarfile earlier.

A new WordPress installation doesn't have a configuration file; you must copy it from the wp-config-sample.php that comes with

the system:

```
$ cp wp-config-sample.php wp-config.php
```

Once that's done, open the file, and look for three lines that define `DB_NAME`, `DB_USER` and `DB_PASSWORD`. Change all three values to reflect the database, user name and password you have created here; this is how WordPress will connect to your database.

Configuring nginx

Next, you'll need to configure the UNIX-level permissions. Every process runs as a certain user, and nginx is no exception. On Ubuntu machines, both nginx and Apache run as the "www-data" user and group. Using a specific user ID to run such programs allows you to ensure that the correct permissions are in place. However, it also means you need to be sure that the WordPress directory and files are owned by that user.

So, you can say:

```
$ cd /usr/share/nginx
$ sudo chown -Rv www-data:www-data wordpress
```

The `-R` option tells `chown` to work recursively. The `-v` option turns on "verbose" mode, meaning that you get additional feedback from the program as it works. I generally prefer to run programs with `-v` to give me more feedback.

Now you have to configure your nginx server. When you installed it, the main configuration file was placed in /etc/nginx/nginx.conf. However, modern nginx configurations also include one file for each server it is running in /etc/nginx/sites-enabled/, with the file /etc/nginx/sites-enabled/default describing the default site.

For the purposes of simplicity, I'm going to assume here that you have a single site, which means you'll be able to modify just the individual "default" file, rather than the overall config file.

As is usual in nginx, the configuration is broken into individual blocks. `server` blocks describe how the HTTP requests coming in should be handled; in this case, you want anything that arrives on port 80 for any

hostname to be passed to PHP. The following server block does the trick:

```
server {
        listen 80 default_server;

        root /usr/share/nginx/wordpress;
        index index.php index.html index.htm;

        server_name localhost;

        location / {
                try_files $uri $uri/ /index.php?q=$uri&$args;
        }

        error_page 404 /404.html;

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
                root /usr/share/nginx/html;
        }

        location ~ \.php$ {
                fastcgi_split_path_info ^(.+\.php)(/.+)$;
                fastcgi_pass unix:/var/run/php5-fpm.sock;
                fastcgi_index index.php;
                include fastcgi_params;
        }
}
```

Let's go through the above, so you can understand what's happening.

First, you indicate that this server will listen on port 80. Unless you have a good reason to choose a different port, this is probably a good idea. Note that if and when you want to use SSL, that goes on port 443 and also requires a separate `server` block. For purposes of simplicity, let's ignore that here.

You also use `default_server` to indicate that if someone provides a hostname that does go to your IP address, but that is unhandled by any

other hostname, then this server should be used. If your system handles only one HTTP server, this directive should be set in your configuration. If your system has multiple virtual HTTP servers, you'll need to decide which should be the default.

The `root` directive describes the directory containing the files you'll want to serve. After opening the tarfile into /usr/share/nginx/wordpress, you tell nginx that the directory contains the PHP programs you want to execute. Actually, it's just one PHP program in index.php that does all the heavy lifting.

The `index` command indicates what files should be read, and in what order, if you don't provide a filename. Note: indicate that index.php should be tried first to give WordPress a chance to run things before static alternatives are attempted.

The `server_name` directive tells nginx which name(s) should be recognized by this server. If you're using `default_server` and have only one virtual host, this doesn't matter all that much. However, if you have multiple servers, giving a name is a good idea.

You then indicate, using a `location` block, what you want to do when you receive a request to the "/" URL—meaning, a directory name. This directive tells nginx that it first should try the URL that you received, but if that doesn't work, then you should invoke index.php, passing it the URL and any arguments that you received with it. In this way, index.php becomes the gatekeeper for any and all requests you recieve.

You then indicate what to do in case of error, separating 404 ("file not found") from more serious server-side errors (50x errors). nginx comes with static files for these errors; you can modify those files if you want something more informative or whimsical.

Finally, you connect nginx to FPM, the PHP back-end system that I discussed in my last column. FPM runs PHP in a separate process, but keeps it going continuously, so you don't have to start up a new process each time. If you find that `php5-fpm` isn't running, you might need to start it with:

```
$ sudo php5-fpm restart
```

Once the above is in place, you can restart nginx:

```
$ sudo nginx restart
```

Point your web browser to your WordPress system's IP address or hostname, and you should see a request to choose a language as part of the WordPress installation. If you do, then you've made it; your server is up and running. Move on to the next page to choose a site title, admin user name, password and email address, and you're all set!

Conclusion

As you can see, it's surprisingly easy to set up WordPress with nginx. Assuming that PHP is installed, and that PHP's FPM system is installed and running, you actually can get an nginx-powered WordPress blog up and running in just a few minutes. And although you could install WordPress via `apt-get` or a similar package manager, doing so means that your updates are at the mercy of the Linux distribution you're using, which inevitably will lag behind the WordPress distribution itself, not to mention plugins, which are perhaps one of the most important parts of the WordPress ecosystem.■

RESOURCES

nginx is a popular server, and as such, there are lots of sources for information about it. One of the best such sources is http://nginx.com, the official site of nginx run by the company that has been founded to develop and support it. From that site, you can read a great deal of high-quality documentation, including a Wiki (https://www.nginx.com/resources/wiki/start) with many user-submitted suggestions.

WordPress, of course, is a hugely popular open-source product. You can read more (*lots* more) at http://wordpress.org. And although there are enormous numbers of blogs, books and references for WordPress, I've found that for simple installation and usage, very little documentation is necessary. You can download, install and use it with a minimum of fuss. That said, if and when you do encounter problems, a search at http://wordpress.org and/or at Stack Overflow generally will solve the problem.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

"It looks like the drone industry has chosen their go-to event!" —Robert Rodriquez, President of the Society of Aerial Cinematography

InterDrone

The International Drone Conference and Exposition

September 7-9, 2016

Paris Hotel, Las Vegas

www.InterDrone.com

The Largest Commercial Drone Show in the World!

InterDrone is Three Awesome Conferences:

Drone TECHCON

For Drone Builders

Content will focus on advanced flying dynamics, chips and boards, airframe considerations, hardware/software integration, sensors, power issues, and software development.

Drone ENTERPRISE

For Flyers, Buyers and Drone Service Businesses

Classes focus on enterprise applications such as precision agriculture, surveying, mapping, infrastructure inspection, law enforcement, package delivery, and search and rescue.

Drone CINEMA

For Flyers Engaged in Aerial Photography and Videography

Class content includes drone use for real estate and resort marketing, action sports and movie filming, newsgathering — any professional activity where the quality of the image is paramount.

135+ Exhibitors • 120+ Classes and Panels • InterDrone Film Festival • Women In Drones Luncheon

Demos • Keynotes • 100+ Industry Expert Speakers

Register Today!

A BZ Media Event

Spinning and Text Processing

Dave delves into complex string processing to write a tool spammers use.

DAVE TAYLOR

Dave Taylor has been hacking shell scripts since the dawn of the computer era. Well, not really, but still, 30 years is a long time! He's the author of the popular *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours* (new edition just released!). He can be found on Twitter as @DaveTaylor and at his tech site: http://www.AskDaveTaylor.com.

PREVIOUS
Reuven M. Lerner's
At the Forge

NEXT
Susan Sons'
Under the Sink

I HAVE A DIRTY SECRET TO SHARE, and I hope you won't think less of me once you learn it. I used to be in the internet marketing world and pitched my coaching programs and DVD sets from stages around the United States. Yes, for $999, I'd teach you how to make money online, and if you were one of the first three to sign up, I'd even throw in my friend's dynamite ebook absolutely free!

Truth is, I didn't last long in that space because I'm much more of a do-er than a salesperson, and it would bug me to no end when people would buy my coaching package—*at 20% off, but only if you* ***sign up right now!***—and then never actually open it and

use it to at least try their hand at creating an online business.

That's all in the past, fortunately, but I've retained an interest in those business opportunity pitches and what they're actually selling. Just like the cliché envelope-stuffing job (you know: "Send me $200 in an envelope, and I'll show you how to ask people to send you money!"), it turns out that a lot of online businesses still are predicated on gaming search engines to gain traffic to pages selling daft and usually worthless things.

And, one way that these entrepreneurs game Google and other search engines is by "spinning" to produce lots and lots of content from a single article that they've paid someone a few bucks to write in the first place.

It's all rather uninspiring, except the spinning idea itself is rather interesting, and I've been toying with writing a shell script to allow easy article spinning for quite a long time. There are more prosaic, less questionable uses for this technology too, like in programs or even games that have text messages useful to vary.

```
The {idea|concept|inspiration} is that each time you'd use a {word|phrase}
you instead list a set of {similar words|synonyms|alternative words} and
the software automatically picks one {randomly|at random}.
```

So the previous sentence would come out of the spinner as "The idea is that each time you'd use a phrase you instead list a set of alternative words and the software automatically picks one at random." Got it? Easy enough.

A more advanced spinner might actually tap a thesaurus, and each time it sees a word, push out a set of synonyms automatically, which the other script then randomly simplifies each time it's invoked.

In fact, go read spam blog comments or spam email, and you'll see the output of these sort of contextless sentence manipulations. They can be... weird, like this:

```
she's got arriving in can easily dresses, still Beth may be 36 yr old
men's city servant, outdoors of waking time 'en femme'. she's single,
symmetrical in addition thinks to achieve marital, "Eventually..."
```

But hey, just because there are bad uses, doesn't mean it's not an interesting project to try to code, right? I trust you to exercise good judgment of your own when you explore this script, okay?

Spinning Out the Spinner

The basic tasks of the script are straightforward: parse the input, isolate each word-choice block, pick one at random, then reassemble everything and display it.

To make things a bit easier, I'm going to start by using `fmt` to make each paragraph one really long line. That way, I then can break the input into lines that don't have a word-choice block and those that do:

```
fmt -w$bigwidth "$1" | tr '{' '\n' | tr '}' '\n'
```

An input line like {this|demo} would then transform.

```
An input line like
this|demo
would then transform.
```

See how that works? I'm going to use `fmt` again at the end of the process to clean up the output.

One facet of shell script programming that most people don't realize is that every loop structure acts as its own subshell, so rather than waste space and time with a temporary file, I'll pipe the output of the `fmt|tr` sequence directly into a while loop:

```
fmt -w$bigwidth "$1" | tr '{' '\n' | tr '}' '\n' | \
while read line
do
  if [ $( echo "$line" | grep -c '|' ) -gt 0 ] ; then
    echo "SPIN THIS: $line"
  else
    echo "$line"
  fi
  lines=$(( $lines + 1 ))
done
```

See how the `fmt` line ends with `| \`, and that feeds directly into the while loop? Very handy structure!

Now I'm going to run this code snippet with the sample input file to see what happens:

```
$ sh spinner.sh spinme.txt
The
SPIN THIS: idea|concept|inspiration
is that each time you'd use a
SPIN THIS: word|phrase
you instead list a set of
SPIN THIS: similar words|synonyms|alternative words
and the software automatically picks one
SPIN THIS: randomly|at random
.
```

That pesky period on its own line is a glitch that'll need to be fixed later, but the basic structure of the script is sound: you can parse and break down the input file data and identify which new lines are selector lines.

The Spinning Function

Instead of just prepending `SPIN THIS:` before a line that has choices, that's a perfect place to put in a function call to a separate block of code that does the actual work.

One of the most interesting parts of the function is how it figures out how many options there are in the given string. It's a specific instance of the general question "how many occurrences of X are in string Y?", and it exploits the little known `-o` flag to `grep`:

```
grep -o '|' <<< "$*" | wc -l
```

Take a deep breath; I can talk you through this one! The <<< notation is a variation on the here document (<<) you've hopefully already seen in scripts. The difference is that the result is fed as a single string on stdin.

The `"$*"` produces the entire argument as given to the function in the main block of the script; the `|` is the character being counted, and of course, `wc -l` produces the number of matching lines (in this case, the

number of delimiters in the line).

All that, and it's not quite what I want, because a line like `word|phrase` has one delimiter, but two choices. Here's how I solve that in this first, skeletal version of the function:

```
function spinline()
{
  source="$*"
  choices=$(grep -o '|' <<< "$*" | wc -l)
  choices=$(( $choices + 1 ))
  echo $choices options, spinning --- $source
}
```

In use:

```
$ sh spinner.sh spinme.txt
The
3 options, spinning --- idea|concept|inspiration
is that each time you'd use a
2 options, spinning --- word|phrase
you instead list a set of
3 options, spinning --- similar words|synonyms|alternative
words
and the software automatically picks one
2 options, spinning --- randomly|at random
.
```

That's it for this month. Next month, I'll finish up the function, including implementing a way to pick one entry randomly from a set of *n* choices, then output the cleaned up copy, ready to use in whatever program or utility you'd like.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

JOIN US IN THE LONE STAR STATE

Austin Convention Center
July 8 - 9

2016.texaslinuxfest.org

Security Exercises

Read on for a crash course on how to plan effective security exercises.

PREVIOUS
Dave Taylor's
Work the Shell

NEXT
New Products

SUSAN SONS

Susan Sons serves as a Senior Systems Analyst at Indiana University's Center for Applied Cybersecurity Research (http://cacr.iu.edu), where she divides her time between helping NSF-funded science and infrastructure projects improve their security, helping secure a DHS-funded static analysis project, and various attempts to save the world from poor information security practices in general. Susan also volunteers as Director of the Internet Civil Engineering Institute (http://icei.org), a nonprofit dedicated to supporting and securing the common software infrastructure on which we all depend. In her free time, she raises an amazing mini-hacker, writes, codes, researches, practices martial arts, lifts heavy things and volunteers as a search-and-rescue and disaster relief worker.

REGULAR SECURITY EXERCISES are, bar none, the most powerful, cost-effective tool for maturing a project's information security operations—when done well. Unfortunately, courses and certifications on InfoSec tend to focus on how to implement specific controls or how to select some baseline best practices when starting from scratch. Little to no attention tends to be paid on how to test what you have and iterate on it. Prepare for a crash course.

Security Exercise? What's That?

A security exercise is a drill designed to propel a team or teams through the steps they would take in the case of a real or suspected information security problem in their organization or project. For example:

- Tell your ops team that the server hosting your

internal bug tracker has experienced data loss due to a critical RAID controller failure. Have them rebuild the server from backups on spare hardware to show that the backups are viable, spare hardware available and the process known and workable.

- Start running an otherwise innocuous, but memory-intensive, piece of unauthorized software on a development server. See how long it takes for someone to notice and what he or she does about it.

Isn't This Dangerous?

Security exercises are not the first step in running an InfoSec program for a project of any size. The first step is coming up with a plan or set of policies appropriate to the size and complexity of the project. For a very small, all-volunteer open-source project, this may be as simple as "Our project manager, $name, accepts risk on behalf of the project and our information security officer, $name, is in charge in the case of a suspected security incident; the integrity of our code base will be prioritized first, confidentiality of yet-undisclosed vulnerability information second and availability of services third." For a larger, more complex organization with paid staff, this normally will start with a Master Information Security Policy and Procedures document supported by a number of other policy documents.

In either case, step one is establishing roles and responsibilities; step two is establishing operational expectations, and step three is testing that your policies, procedures and expectations *work*. If you aren't testing, you don't really know that it works.

Scheduling exercises at a predictable time and reminding others when it will happen prevents confusion among staff. It is wise to begin with low-impact exercises (more on this below) that don't leverage production systems, and move on to higher-potential-impact exercises only when the organization's infrastructure and personnel have had most of the bugs shaken out. If something as small as a runaway process on a single server can seriously impact your business, it's better to find out at a planned time with all hands on deck than at 4am on a holiday when no one who knows what to do can be reached. The whole point of security exercises is to increase resilience: raise the threshold of what is normal for your team to deal with, what your systems can shrug off.

Why Are Security Exercises Important?

When I respond to a security incident that's gone disproportionately bad—that is, far worse than the incident should have gone given the resources and security needs of the organization—it tends to be true that more than one thing has gone wrong. However, the root cause of how those things were all allowed to go wrong at once is almost one or both of two things: lack of interest in and support for information security from organization leadership, or the failure mode I call "death by supposition".

"Death by supposition" is when we make decisions based on "facts" that are *supposed* to be true, but have not been *tested* by us. For example, suppose that hardware or software will behave the way the vendor said it would. Suppose that anybody in the company remembers the incident response plan that was written, approved and stuck in a drawer two years ago. Suppose that the "best practices" written for companies in your sector don't overlook some way in which the sector has changed, or your company is unique. Suppose that your backups work the way they were designed to, and nothing has gone awry in the 25 updates since the system was put in place 18 months ago. Suppose that your VPN hardware fails closed the way the vendor's sales staff insisted that it would.

Supposition kills, and it's an insidious killer because, unlike bad leadership, it's easy to miss. We often aren't aware of our assumptions until something goes horribly wrong—better for that something to be a simulated attack than a real one, leading only to simulated consequences.

Security exercises, done right, will do the following:

- Reveal whether systems and technical controls (still) work as expected.
- Ensure that security, ops, leadership and other team members are on the same page.
- Reveal holes in procedures and policies.
- Provide your team with vital practice at operations that may someday need to be done quickly and/or under stress, especially disaster recovery and incident response procedures.

- Provide your team with *stress inoculation*. This is something that SWAT teams, martial artists, search-and-rescue teams, firefighters, military and so on already know is an essential part of their live drills: getting used to something so it doesn't register as such a large stressor any more.

- Provide non-security personnel and security personnel alike with valuable hands-on security training.

- Improve the relationships needed to make security improvements and incident response go more smoothly.

Most important, well-executed security exercises take your organization from the land of supposition to actually knowing where your weaknesses are, where your resources should be going, and what you are doing right. Don't guess. Know.

What Makes a Good Security Exercise?

Asking what makes a good security exercise a lot like asking what makes a good martial arts or search-and-rescue exercise. If you exercise only once or do only one kind of exercise, you won't get the results you are after. The right question is "What makes a good security exercise *program*?"

The answer is:

- Regularity.

- Purpose and focus.

- Attention.

- Follow through.

Good Security Exercises Happen Regularly

In a small organization without much in the way of infrastructure, run an exercise once per quarter. It doesn't have to be over-the-top in scope. Just make sure you are doing *something* regularly.

In a medium-sized organization with some dedicated IT resources and some in-house infrastructure to look after, run an exercise once per month. This gives you enough time to design the exercise well, do a thorough postmortem and integrate what you've learned into your security operations.

In a large organization with complex IT infrastructure, security exercises should be a near-constant affair, carried out within various units and across units with support from your security team. Consider building out some infrastructure to make exercises easy to run frequently.

These are rough guidelines only; use your brain and a little trial and error to find the right interval for your organization. Don't be afraid to run exercises when key people are missing. Often, real incidents happen at the least convenient time possible: when the security officer is on a long flight, when a needed systems administrator is out sick and so on. Get used to the unexpected.

Purpose and Focus Matters

If I listed all the security exercises I could think of, and your organization drew and ran a random one each month, you'd probably be better off than if you ran no exercises at all. However, exercises tailored to your organization and infrastructure are far more effective. Much of an exercise's quality comes from its planning.

What Are You Exercising?

Each security exercise may be exercising people, systems, policy and procedures, or some combination of the above. Note that I said "exercising" rather than "testing". Security exercises are most effective when they are used as a diagnostic and training utility rather than as a performance evaluation. Using security exercises as a performance metric for personnel tends to decrease the quality of collaboration and initiative during both real and simulated incidents. In the organizations with the most effective security programs, exercises pit team members against the exercise, rather than against one another or against an evaluation mechanism.

In organizations where security exercises are new, they often are broad—for example, "Can we restore this system from backup?" or "What do we do if our password database is leaked?" In organizations

with more practice, exercises often are a mix of the broad, end-to-end kind with more targeted exercises that test specific capabilities, such as detection, ability to bring specific systems up or down smoothly, response to specific attacks and so on.

I keep a list of things I'd like to test via security exercises for each project/organization I'm responsible for. The list contains:

- Any issue for which we've argued whether the control we have is the "right" control.
- Any system or component we haven't tested recently (or at all).
- Any known vulnerability we *think* we've closed.
- Any known vulnerability we haven't effectively closed and to which I'd like to draw leadership or team member attention.
- Anything that looks like a single point of failure—including people.
- Any behavior we assume our team members will do, but haven't tested recently.
- Procedures for "black swan" events—potentially devastating security events that also are rare/unlikely enough that we have practice dealing with them only if we create that practice.
- Procedures that involve roles for which we've had personnel turnover.
- Procedures where I'm not sure it's clear who will be doing what task or job.

Prep

Once you've chosen your exercise focus, prepare for it. Unless your organization is *very* mature from a security standpoint, this will begin with setting a schedule and notifying everyone in the organization, then reminding them again just before the exercise starts.

The simplest security exercises are what we call *tabletop exercises*.

In a tabletop exercise, all the relevant parties sit down together and, in real time, walk through a hypothetical scenario noting everything that would be done in response to a problem. Tabletop exercises are the least informative type of security exercise because they lack realism, but they're also the most lightweight exercise to run.

A tabletop security exercise requires:

- An exercise scenario, written up in as much detail as possible and well understood by the person running the exercise.
- A way for everyone on the team to meet in real time: conference room, conference call, IRC, video conference—whatever works best for your team.
- All of the principals relevant to the organization's potential response to this scenario.
- Anyone else who would benefit from participation in the exercise.
- Excellent note-taking.

That's it. So, you now have no excuse not to at least run tabletop security exercises within your group.

Live exercises are a bit more involved, but they provide a wealth of information and experience to your team that can't be gotten in any other way apart from having something actually go wrong. There are, of course, degrees of "live-ness". It's acceptable—and often easiest on your team—if you start at the less-ambitious end, where you present a hypothetical then step through the resolution live, then progress to more involved exercises where problems are introduced on actual systems.

A live exercise will involve at least a Blue Team and a Red or White Team (possibly both). The Blue Team is the defenders. That team will be doing its job throughout the exercise much as it would during a real incident. The Red Team is the attacker. That team causes whatever problem sets off the exercise, and in advanced adversarial exercises, may continue to act throughout the duration of the exercise. The White Team is the referee. It may provide clarification throughout the exercise, observe and so on, but

should not be actively participating (usually).

The most important part of preparation for a live exercise is setting and communicating the boundaries of the exercise. Consider the following:

- *Who should be informed that the exercise is going on?* Until your organization's information security is fairly mature, this should be everyone within the organization. Once live exercises are part of your regular routine, it can become fun to schedule some exercises in secret, telling only key members of the Red and White teams when it will happen, in order to get more realistic responses. Surprise exercises tend to end badly in organizations that aren't very practiced at running exercises, however, because if not planned carefully, they can backfire. Also, consider whether to inform anyone outside the organization. For example, you might want to warn your data center's staff (if you colocate) before running an exercise to prevent them from initiating a security incident upon observing suspicious traffic to/from your systems.

- *How much degradation of live services is acceptable due to an exercise, and how will you ensure that this limit is not exceeded?* Live exercises can be unpredictable. This is good, because real incidents are unpredictable. However, it is important to scope exercises so that they don't exceed the organization's tolerance for interruptions to normal service. Once you have a great deal of practice running exercises, you'll be able to play it closer to the wire, but while your organization is new at this, consider limiting security exercises to operating on non-production systems, and/or systems that are easily re-imaged after the exercise.

- *How will you clean up the mess after the exercise and ensure that it was cleaned up thoroughly?*

- *How will you handle conflicts between the security exercise and other duties?* Ideally, the answer here is "the same way we would handle conflicts between a real incident of this magnitude and other duties". However, institutionalizing that will take work in most organizations. Beginning with lower-grade simulated incidents (for which diverting effort from other projects might not be

acceptable in real life) and working your way up may help. After a few successful responses, plan to simulate a more critical incident (preferably while nobody is in the middle of putting out any real-life fires) and see how your teams do. If adequate effort isn't shifted to the exercise, it's important to point this out as a metric of how a real exercise will be treated. It's been my observation that managers who refuse to reallocate effort during an exercise almost always refuse to reallocate effort (or reallocate more than inadequate, token effort) during a real incident.

Red Team

When forming the Red Team, do your best to pull members of your staff who have not been on the Red Team in the past, or at least not recently. Using non-security-team personnel on the Red Team and rotating those personnel regularly can provide an incredible morale boost to your organization because:

- It's fun and different from being the defender.
- It's a good learning experience.
- It keeps people from feeling like being on the Blue Team is a test.
- It builds investment in the success of security exercises.

Not everyone on the Red Team needs to be technical. Plenty of exercises can have a social-engineering aspect to them, and those are carried out just as well by non-technical staff from time to time.

Give the red team plenty of preparation time, but urge them to keep the nature of the planned exercise a secret. Most white-hats who don't do security full time (and many who do!) don't have much experience carrying out mischief, so they'll need time to familiarize themselves with tools and test techniques. This becomes faster and more lightweight the longer your organization does security exercises, because once you are practiced, there likely will be one experienced Red Team member participating in each exercise to help the less-experienced ones.

White Team

Not all exercises need a White Team. However, if any part of the exercise is hypothetical rather than happening live, a White Team is necessary to answer any questions not explained in the hypothetical. The White Team may have to improvise, if it's asked something the exercise designer did not expect!

White Team members often play bit parts in the exercise as well, representing entities outside the project, such as frustrated users or curious reporters.

Blue Team

The Blue Team is everyone not explicitly placed on the White or Red Team and not explicitly excluded from the exercise. The Blue Team is generally responsible for reacting to the simulated security incident as it would to a real one. The main differences will be that, unless you have a partner organization that participates in your security exercises, any outside communication that would happen in a real incident is directed instead at the White Team during exercises.

Follow Through

It is of paramount importance that members of every team record their actions and ideas throughout the exercise. The most important part of any exercise is what is learned from it, and if the knowledge isn't captured, the team as a whole won't learn.

Debriefing

Debriefing an exercise ideally is done within a few hours of the exercise's conclusion. However, with longer, more complex exercises, this may not be possible. I cannot stress enough the importance of good record-keeping to ensure that nothing significant is forgotten before the debrief.

Typically, the incident response leader (Blue Team lead) is responsible for writing a report on the exercise. However, it's been my preference to ask that person to withhold the report until *after* everyone involved in the exercise has had a meeting to debrief the exercise so as not to taint anyone else's recollections.

The debriefing meeting should walk through the exercise from start to

finish, giving everyone who participated the chance to voice thoughts, opinions and observations. Anyone interested should have the opportunity to ask questions, and good notes should be taken so that the Blue Team incident response team lead can integrate anything new and interesting into the final report.

The Report

Report-writing may sound boring, but it's an essential part of the process. You've just invested quite a bit of time and effort in a learning exercise. Losing what you've learned would negate that investment. It is important to get the details down so you can refer back to them later when you want to compare a similar incident (real or simulated) or remember how some piece of software or equipment performed. It's also important to have enough information to back up the conclusions and recommendations at the end of your report.

Reports don't have to be fancy or formal if that's not your organization's usual mode of communication. What they should have is a narrative describing the exercise—who was there, what happened, what the timeline was—a summary of what was learned and any suggestions as to how security could be improved through technical controls, policy, training, resource allocation or other methods.

Don't Put It in a Drawer

Finishing the report is not the end of the exercise: your organization either needs to implement the recommendations made in the exercise report, or the person who accepts risk on behalf of your organization needs to document which recommendations will not be implemented and why.

Lather, Rinse, Repeat

These exercises are not an effort to train until you succeed but to train until you can't fail. Although no security program is perfect, if you've trained to the point of near perfection against advanced persistent threat drills, runaway script kiddies become child's play.

In the event of a true failure, the exercise should be rerun with a slight variation within six months. This verifies that new training and controls

have remedied the problem, provides needed practice and gives the team an opportunity to overcome the loss, increasing morale.

Tips and Tricks

Here are some pseudo-random thoughts about planning, running and using the information from security exercises:

- Keep track of what is learned in exercises, and keep copies of exercise reports. Ideally, these are great fodder for demonstrating the success of your efforts in improving information security for your project. In the worst case, when recommendations go unheeded, referring decision-makers back to this after a real incident often can bring them around to taking security issues more seriously in the future.

- Have fun! Be willing to see exercises as a game. Encourage creativity and limits-testing. Drop funny Easter eggs into the exercise. This is how you'll get the best bang for your buck in terms of learning and morale.

- Be willing to adapt. The planned exercise doesn't have to be the exercise if something goes wrong. Pivot, and keep everyone on their toes.

- Consider how you'd like your team to respond during real incidents, and be sure that this is the behavior you encourage during exercises.

- Treat every exercise like a success, even when the results are embarrassing. If your incident response usually goes perfectly smoothly, your exercises aren't hard enough. Expect some things to need tweaking after most exercises. It is very important that your team members not see security exercises as an opportunity for them to be graded. If someone performs badly, your response should be that more training is required.

- Start small, and build the difficulty and complexity of exercises over time. Just like weightlifters can't lift 400 pounds on the first day or progress if they don't add weight over time, a team won't get better if it's not challenged. If you are in fact learning, what was challenging last month won't be challenging next year.

- Notes and debrief discussion from Red, White and Blue Team members will identify additional scenario opportunities. Keep track of these ideas as they come up so you have them at the ready when you need to come up with a scenario.

- If you experience resistance to security exercises from The Powers That Be, figure out what influential people you can invite to the Red Team for an exercise. Don't make them token members; make sure they are active and having fun. This tends to turn people around on the practice.

- Don't try too hard for absolute realism in all exercises. Realism is where you begin, but if you are willing to venture into the unreal occasionally, you will learn more. The best Red/Blue exercise I ever participated in was part of an ICS-CERT training out at their facility in Idaho. They built out a surprisingly realistic playground for us to attack and defend, then set us loose with a ridiculous constraint: under no circumstances can you take this infrastructure down to fix its obviously life-threatening problems. No sane person issues that edict in real life. However, not being able to take down the network that the White Team so helpfully built with security akin to Swiss cheese after a mouse convention and shore it up before attackers struck made the Blue Team—of which I was a part—try things we'd never do in real life. I found myself breaking into my own systems to reclaim them from the Red Team, using ARP-spoofing tricks I'd thought died out in the 1990s to reclaim IP addresses on my internal network, and all sorts of other shenanigans. It made me think fast about how the Red Team was operating, and it led me to teach the other Blue Team members details of OSI layer 2 manipulation that many had not been exposed to.

- If the White Team is experienced in exercise design and experienced in running live exercises, do not be afraid to break my non-interference rule. In the aforementioned ICS-CERT training, the White Team kept us on our toes in part by messing with whomever was in the lead and helping whichever team was struggling, in subtle ways. If done badly, White Team interference can ruin an exercise. If done well, it can ensure that everyone is pushed to the limits, even when the Red and Blue Teams have a significant disparity in skill, resources or team cohesion.■

Example Security Exercises

1) It's Gone

Pick a system, any system. Think of a reason why it's completely hosed—failure of the entire RAID array, fire in the data center, evil script kiddies, sysadmin mistake—and see how your team copes. Some questions to ask when all is done:

- If you don't have another of these systems to fail over to, where are your users going while the system is down? What stopped working and for how long?
- If you have a failover system, how long did it take to fail over? What did your users experience in the meantime?
- How hard was it to replace the system? Were backups adequate? Did the available personnel know what to do and have the authority to do it?
- What data was lost? Are backups being made often enough?
- Were any other systems impacted by this system's death? For example, if your LDAP server died suddenly, did administrators still have access to other systems? Did anything fail open?

2) Naughty Ned

Choose a team member with elevated privileges (any member of your security or systems administration/ops team is usually a good choice, so might be a leadership team member or a developer). Pretend he or she has been fired, and revoke all of his or her privileges. Now he or she gets to cause whatever chaos he or she can with any privileges that remain. This is a great way to test your off-boarding checklist.

3) Wolf in Sheep's Clothing

Most of the Red Team plays the part of ordinary users here. One plays a malicious user. Can the Blue Team terminate the malicious user's activity without negatively impacting any of the nice users?

4) Committer Should Be Committed

This is a great one for software development teams. A developer, working while sleep-deprived (thank you Red Team), has committed something to the master branch of the repo that he or she shouldn't have. It might have been login credentials for an internal system or naked pictures of the boss' dog—the content doesn't matter. The important thing is that it has to go.

See how your team removes the offending data both from the working tree *and* the repository history, without breaking everyone's workflow beyond recognition.

5) Operation!

If you run a DevOps environment, this one's for you. It's far too easy for deployment workflows to end up with very low bus factors (the number of people who must be hit by a bus before the project is doomed or at least in serious trouble). Watch a deployment or two and figure out who the 1–3 most critical people are in that path, then declare them unreachable for the purpose of the exercise.

Now, suppose that a critical security vulnerability has been found in your deployed product. Challenge your team to make a trivial code change (for example, add a comment saying "We did it!" to the code at a specified point), then run your entire test suite and deploy the code with those critical people gone.

6) Finger in the Dam

Find a (hopefully fairly harmless) proof-of-concept for the most recent security vulnerability for which you applied patches. Run it against everything and find out whether the hole really was plugged.

7) Negative Nancy

Have a Red Team member contact your primary customer support avenue, playing the part of a user who is absolutely certain that his or her private information entrusted to your service has been compromised. Bonus points if the character is a "difficult" personality. See how the team handles it.

8) Fell Off a Truck

Your primary authentication database has fallen off a truck (your choice whether this is your database of external user accounts or something for internal personnel only). Demonstrate how you would notify those affected and force password resets. Bonus points if you can detect and flag attempts to use compromised credentials.

9) Ewe Did It

Start an (otherwise innocuous) process on one of your systems that occupies as much RAM as it can get its hands on. See how long it takes for anyone to notice, and how they respond.

10) Stowaway

Connect an unauthorized network device into your network and let it talk to something. See how your team tracks it down and removes it.

11) Exfiltration

One of your employees has decided he or she would like your big, valuable, internal database. The Red Team tries to exfiltrate the target (any way it likes) without being detected.

12) Nosy Nelly

One of your systems starts nmapping the network. Does anyone notice?

13) Blame the Mailman

A system that never should send mail starts sending (or trying to send) spam. What happens next?

14) Delivery

In a disguise, try to make your way into some limited-access area of the building, such as your data center. It helps to appear pregnant, talk on the phone, tailgate someone, carry something heavy or insist you are making a delivery or have an appointment. See if anyone stops you.

15) Pick-Up Stix

Drop some USB sticks around the building—in the parking lot, the restroom, a conference room, a lobby. Place an autorun executable on the sticks that notifies you when they are inserted in a machine that autoruns USB devices, and place an interesting-looking file on there that also tries to call home when opened.

16) Phishing Expedition

Send a convincing phishing e-mail (with at least one flaw that a reasonable person would pick up on) to your staff, directing them to a fake login page and see who gives up their credentials. Note: this one is likely to rankle some people who feel duped when you come out and tell them what happened, but it's really good at driving home the importance of phishing awareness if you can afford the political fallout.

17) Compromising Positions

Suppose that a rootkit has been discovered on a critical piece of infrastructure on your internal network (for example, your satellite server or your LDAP server). Challenge your team to prove that none of your *other* systems have been compromised (not *assume*, *prove*).

18) Failure Is Always an Option

Single points of failure frequently are discussed in meetings. Pay attention and document these, then break one. Does the scope of the outage match expectation? Does the recovery time/process match expectations?

19) Free for All

This is a big, high-investment exercise to run, but it's also the best. Set up a dedicated environment for your exercise to run in that is not connected to your other internal networks or to the public internet. Provide a set of services that needs to be kept running and consider adding some data meant to be kept confidential. Don't set up that environment in the most secure way possible.

Set targets for the Red and Blue Teams with various point values—for example, 10 points to each team for each system it controls at the end of the exercise, 20 points for the Blue Team for every half hour that a particular service continues without interruption, 50 points for the Red Team if it finds and decrypts such-and-such a file. Then set both teams loose with nothing but a time limit and see what happens.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

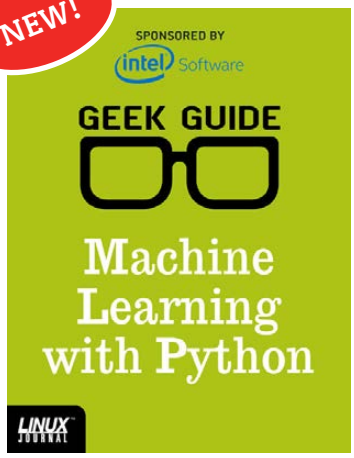
Linux Journal eBook Series

GEEK GUIDES

Practical books for the most technical people on the planet.

FREE Download NOW!

NEW!

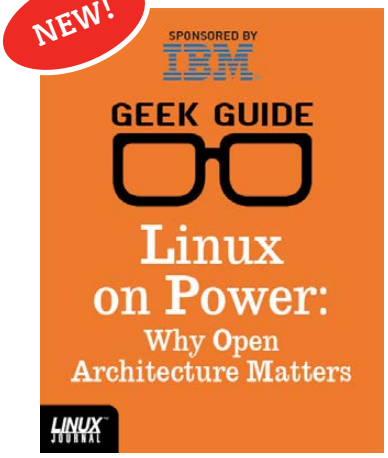


Machine Learning with Python

Author:
Reuven M. Lerner

Sponsor:
Intel

NEW!

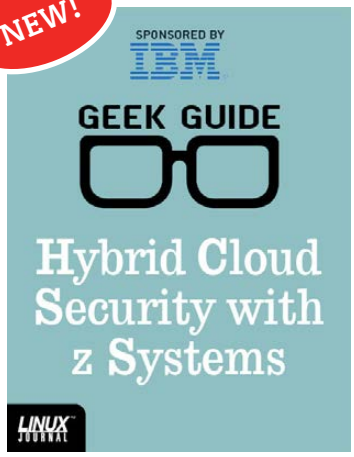


Linux on Power: Why Open Architecture Matters

Author:
Ted Schmidt

Sponsor:
IBM

NEW!



Hybrid Cloud Security with z Systems

Author:
Petros Koutoupis

Sponsor:
IBM

NEW!

LinuxONE: the Ubuntu Monster

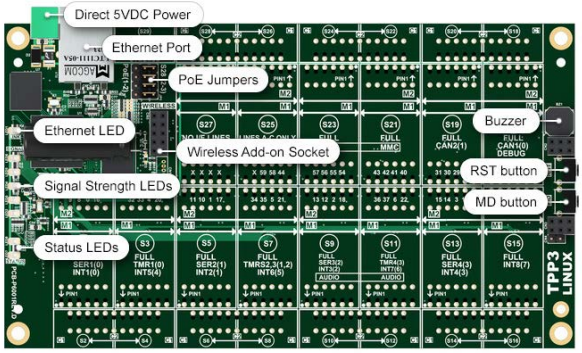
Author:
John S. Tonello

Sponsor:
IBM

Go to **http://geekguide.linuxjournal.com**

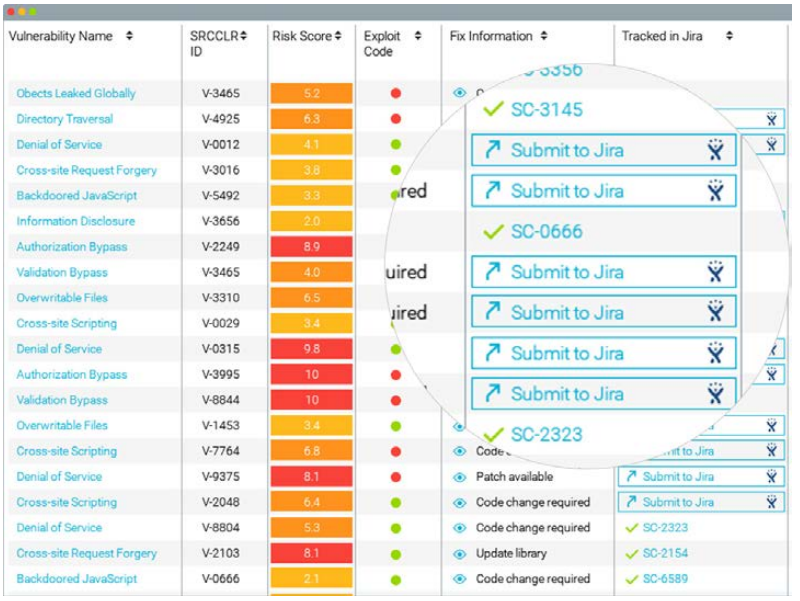
PREVIOUS
Susan Sons' Under the Sink

NEXT
Feature: Android Browser Security



Tibbo Technology's Tibbo Project System

The motto for the Tibbo Project System (TBS) is "Take what you need. Leave out what you don't." TPS, winner of a red dot Design Award, is Tibbo Technology's highly configurable and affordable automation platform featuring miniature blocks that implement specific I/O features. Need a certain I/O function? Install the respective Tibbit. No use for something? Skip it. Tibbo Technology argues that this module-based approach saves users money by allowing precise definition of the features in automation controllers. Tibbo's latest complement to TBS is a Linux-based Tibbo Project PCB (LTPP). The new LTPP3 board runs Tibbo's own, highly polished and updated distribution of Linux and is based on the powerful Texas Instruments 1GHz Cortex-A8 Sitara CPU. What sets the LTPP3 apart from plain vanilla products, such as Raspberry Pi and BeagleBone, is its mechanical and electrical compatibility with Tibbo's Tibbit blocks and size-3 Tibbo Project Box enclosures. Uses for the LTPP3 include running Embedded AggreGate, Node.js and TiOS applications, not to mention use as a generic Linux board.
http://tibbo.com

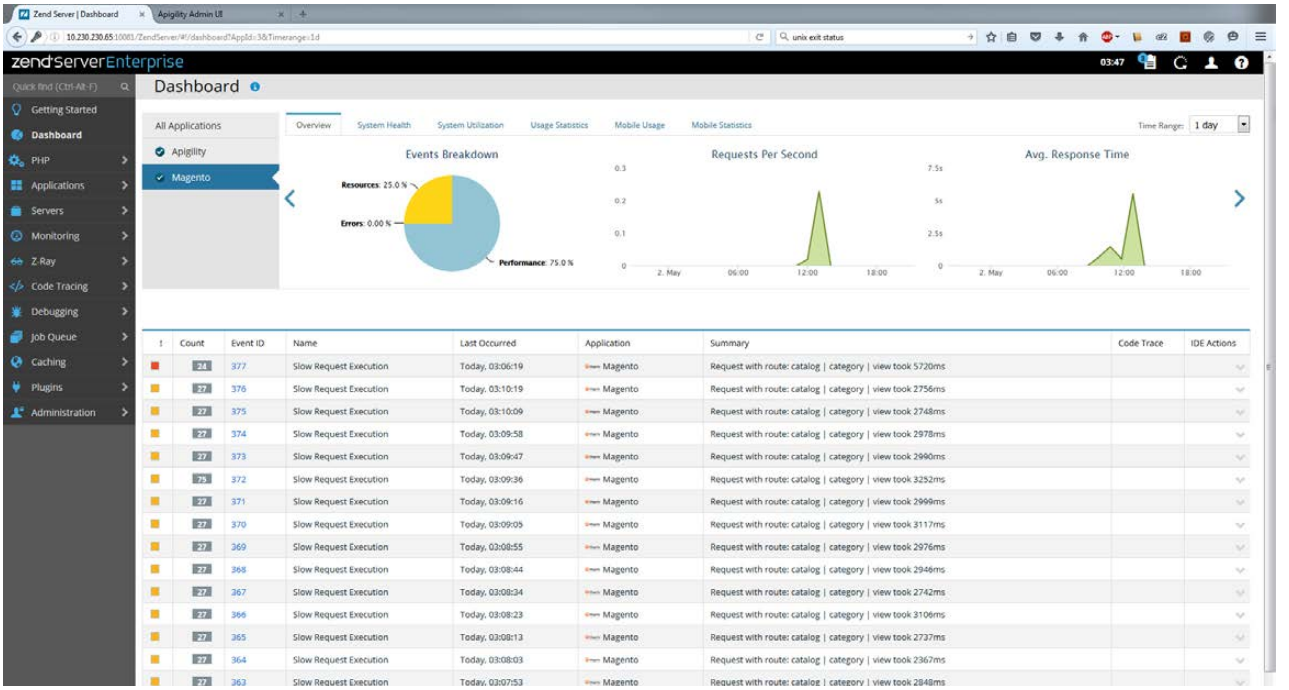


SourceClear Open

Open source and DevOps have been a boon to software development. Nevertheless, sneaky hackers understand—and exploit—the fact that reusable code also means reusable vulnerabilities to distribute throughout the global software supply chain. To aid developers in navigating this new threat landscape, SourceClear announced a new product, SourceClear Open, a cloud service that tracks thousands of threat sources and analyzes millions of open-source library releases. In explaining the need for SourceClear Open, the company notes that developers are held increasingly accountable for security, which creates demand for tools that help them with this responsibility. Unfortunately, traditional security products are insufficient, and public and government-backed software vulnerability databases have limitations. The SourceClear Open tool vaults beyond these databases, enabling developers to identify what open-source libraries they are using, what vulnerabilities exist, which vulnerabilities actually matter, and what needs to be done to fix them. And, perhaps most important, SourceClear Open integrates with the tools (GitHub and Jenkins) and supports the languages (Java, Ruby, Python and JavaScript) upon which development teams rely.
http://srcclr.com

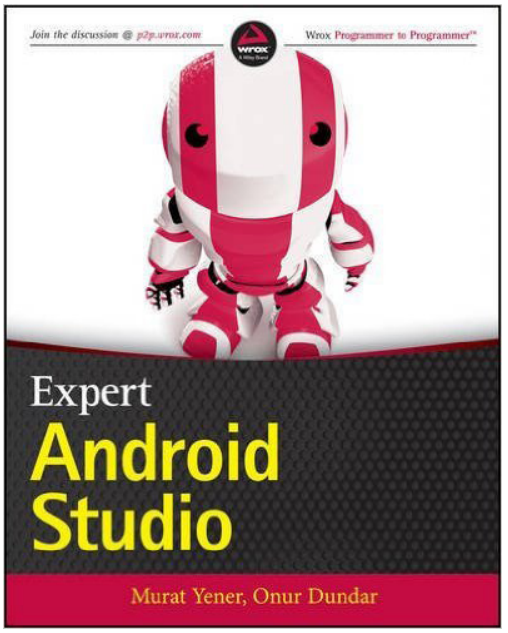
LiveCode Ltd.'s LiveCode

"Everyone Can Code" is the vision that its maker has for LiveCode, a highly productive coding environment for Linux, Android, iOS, Mac, Windows and Server platforms. LiveCode Ltd. spent three years and weathered a $750,000 Kickstarter to enable a LiveCode rewrite from the ground up. The result is LiveCode 8, which adds a new plug-in architecture that allows users to drag/drop widgets containing sophisticated functionality. Widgets can be built within the LiveCode platform and used to wrap functionality in external libraries or each of the operating systems LiveCode supports. The new features in LiveCode 8 are intended to empower a new audience of app makers. Some of these include nine pre-made widgets, 46 new extensions, the all new LiveCode Builder language, a 3.5x performance boost, Script Only stacks for better version control and working in teams, LiveCode for HTML5 and a new Feature Exchange for community funding of new features, among others.
http://livecode.com



Rogue Wave Software's Zend Server

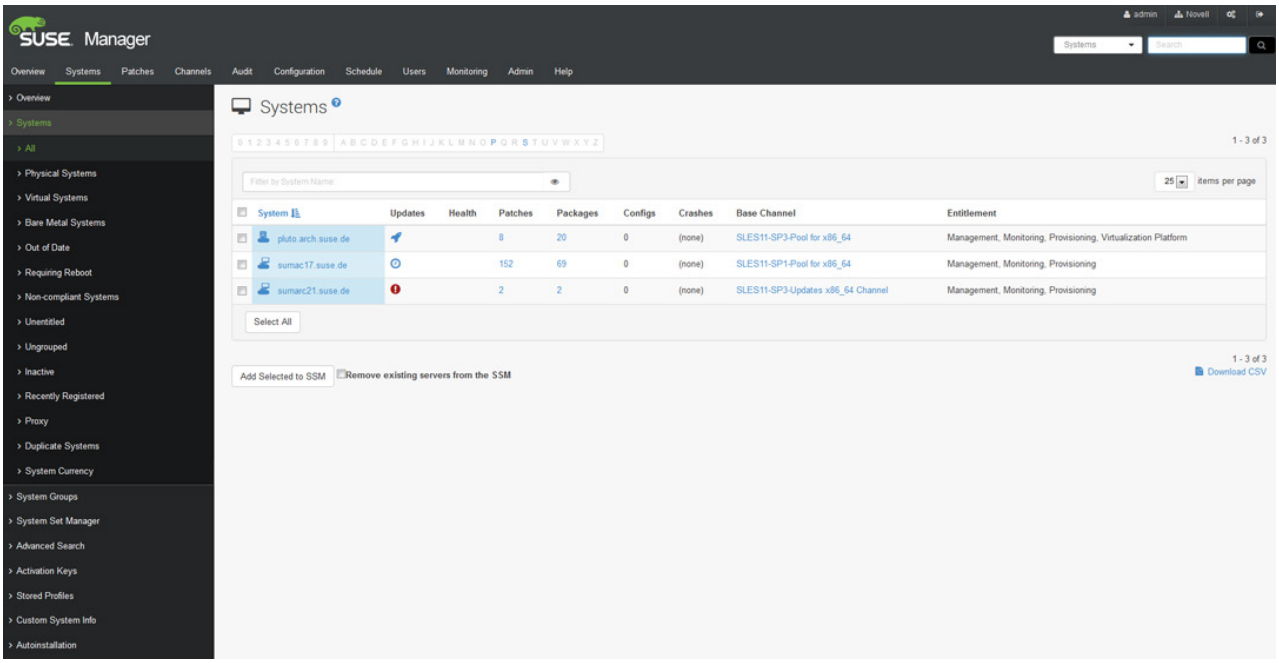
The CTO at Rogue Wave Software, Zeev Suraski, says he's never seen anything like PHP 7 in the software space—namely the halving of hardware needs after a mostly painless software upgrade. Organizations salivating to leverage this massive performance gain would be wise to investigate Zend Server 9, an application server that builds on the benefits of PHP 7, both on-premises and in the cloud. This new version of Zend Server also offers code tracing and black box recording, making it an effortless process to perform root-cause analysis. Another key feature is Z-Ray, a developer toolkit that accelerates developer productivity by displaying all of the under-the-hood details of a page request across all the PHP scripts involved in building a page. Finally, with Zend Server 9, Rogue Wave introduces Zend Server Professional Plus and Zend Server Enterprise Plus editions that offer customers expert support for the open-source stack underlying the Zend Server-hosted PHP application.
http://roguewave.com



Murat Yener and Onur Dundar's *Expert Android Studio* (Wrox)

Expert Android Studio is a new book for expert and experienced developers who want to take their Android programming skills to the next level by unleashing the potential of Android Studio. Authors Murat Yener and Onur Dundar, developers at Intel and self-described open-source code geeks, have packed their new Wrox-published book with best practices and advanced tips and techniques on Android tools, the development cycle, continuous integration, release management, testing and performance. After reading the book, developers will push the boundaries of the Android platform with the developer tools. Besides discovering the basics of working in Android Studio and Gradle, readers will explore the application architecture of the latest Android platform; understand the Native Development Kit; complete the development lifecycle with automated tests, dependency management, continuous integration and release management; write their own Gradle plugins to customize the build cycle and write plugins to support development tasks.

http://wrox.com



SUSE LLC's SUSE Manager

SUSE Manager is a open-source IT management solution with a centralized console for managing multiple Linux distributions, hardware platforms (x86, IBM Power Systems and z Systems), as well as physical, virtual and cloud environments. SUSE says that the solution helps customers reduce the complexities of managing their IT infrastructures, a key advantage as customers look to cut costs and increase the responsiveness required to adopt DevOps and hybrid cloud solutions. The updated SUSE Manager 3 is now available, featuring Salt automation software, improved configuration management, easier subscription management and enhanced monitoring capabilities. Confronting the fact that manually updating, patching and configuring servers can be difficult and time-consuming, SUSE Manager reduces costs with automated, centralized management of Linux systems.
http://suse.com

Clarus Glassboards LLC's Balance Luxury Ping Pong Table

Got VC cash to burn on your startup's cool factor? Clarus Glassboards LLC has just the image-maker you need in Balance, the "World's First Luxury Ping Pong Table". A limited edition item—just 25 will be made—Balance will get your team's creative juices back in the game. Balance is a horizontal version of Clarus Glassboards' primary product, esthetically pleasing dry-erase glass systems of various forms that allow teams to brainstorm and share ideas or add "chic" accents to their offices. With Balance, work and play are creatively fused like never before. The table is set atop a refined wood base, and the mesh net is removable, allowing it to be used for conferences and dining.
http://www.clarus.com

The Firebird Project's Firebird Relational Database

Firebird distills its identity into the phrase "True universal open-source database" and boasts not only of being "free like free beer" but also, fittingly, of being "free like a bird". The latter permits anyone to build a custom version of the Firebird, as long as the modifications are made available for others to use and build upon. Technically speaking, Firebird is a relational database offering many ANSI SQL standard features and runs on Linux, Windows and various UNIX platforms. Firebird 3.0 is the latest major release that succeeds most notably in unifying the server architecture and improving support for SMP and multiple-core hardware platforms. Parallel gains in 3.0 include an improvement in the threading of engine processes and the options for sharing page cache across thread and connection boundaries, among other improvements.
http://firebirdsql.org

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.

RETURN TO CONTENTS

ANDROID BROWSER SECURITY—What You Haven't Been Told

Google Android is undergoing legal attacks in several jurisdictions due to perceived abuses of its market position. Android security is beginning to play a larger role in these conflicts, and significant penalties for Google, network carriers and OEMs due to security shortfalls can no longer be discounted.

CHARLES FISHER

PREVIOUS
New Products

NEXT
Feature:
Radio Free Linux

This article focuses on flaws in Android's stock web libraries, while acknowledging related exploits. Some modern Android browsers have critically weak encryption and other dangerous flaws that cannot be patched or otherwise corrected. This weakness extends to multiple browsers and applications and is determined by the linkage to the system webcore on older OS versions. HTML applications that do not provide their own rendering engine should be avoided for all versions of Android less than 5.0.

Recent statistics indicate that 19% of the population accessing internet information systems have been the victims of an exploit or data breach, causing 45% to reduce or eliminate reliance upon electronic commerce and communication (http://www.theregister.co.uk/2016/05/13/americans_cutting_back_on_online_activity_over_security_and_privacy_fears). Computer security flaws are resulting in a direct impact on the economics of online business, and this must be corrected.

Weakened WebKit

Most mobile platforms (including Android) owe a great debt to the KHTML rendering engine from the KDE Konqueror web browser. Mobile HTML is essentially a monoculture from the perspective of an OS default browser—they all emerge from KHTML, which won this position by providing a high-quality codebase under a reasonable license at the right time.

Although one would hesitate to call Apple a consistently good steward of KHTML due to past friction with the Konqueror project (https://blogs.kde.org/2005/04/29/bitter-failure-named-safari-and-khtml), the Safari browser introduced a compelling rework of KHTML known as WebKit. Apple has provided both new features and regular security fixes for WebKit (more than 100 security-related patches in 2015), which eventually were brought back to Konqueror (http://arstechnica.com/information-technology/2007/07/the-unforking-of-kdes-khtml-and-webkit). As Safari moved from desktop MacOS to mobile

The opinions expressed in this article are those of the author and do not reflect the opinions of *Linux Journal* or its Editors.

iOS, WebKit assumed the mantle of mobile browser leadership, and that role never has been seriously challenged.

Google also adopted WebKit into both Android and the Chrome web browser. Chrome has become the dominant browser by market share, and it receives regular updates from Google on all platforms where it is supported.

However, Google also added WebKit as a library to Android. Any application can link the system WebKit to render HTML as part of the User Interface (UI) by calling WebView, which links in /system/lib/libwebcore.so.

The problem with Android's bundled WebKit is that for older versions, it is never updated, which is not well known. Android 5.0 Lollipop is the first release where the bundled WebKit can be patched.

For Android 4.4 KitKat, the bundled WebKit 537.36 (http://jimbergman.net/webkit-version-in-android-version) and its TLS implementation does not conform to best-practice encryption as defined in RFC 7525 (https://www.rfc-editor.org/rfc/rfc7525.txt). As reported by the Qualys SSL Scanner (https://www.ssllabs.com):

- SSL version 3 is enabled—a *must not* from the RFC. This can be exploited via the POODLE downgrade attack to decrypt traffic.
- Weak "export-grade" ciphers are allowed—also a *must not* from the RFC. This enables hostile decryption by third parties via the FREAK attack.
- RC4 ciphers are allowed—also a *must not* from the RFC.
- Weak Diffie-Hellman primes are allowed, which can be exploited via the Logjam attack.

These software flaws preclude the use of the KitKat system WebKit for sensitive transactions of any kind. Android JellyBean, which spans numerical releases of 4.1 through 4.3, has WebKit version 534.30, which is even worse, as it wasn't actually updated since Android version 4.0.1 Ice Cream Sandwich:

- JellyBean disables TLS 1.1 and 1.2 by default, in addition to allowing SSL v3.

- JellyBean also is vulnerable to the "Same Origin Policy" bug CVE-2014-6041 (https://community.rapid7.com/community/metasploit/blog/2014/09/15/major-android-bug-is-a-privacy-disaster-cve-2014-6041), which allows hostile websites to spy on browser activity.

Google cannot and will not patch these or other bugs (http://www.zdnet.com/article/google-why-we-wont-patch-pre-kitkat-android-webview), because the Android patch process is both technically and politically (too) difficult—in Google's own words, patches are "no longer practical to do safely". Google regularly abandons large segments of the Android base, and the above design flaws now infect more than 50% of Android devices. As of April 4, 2016, KitKat is 33.4% of the total Android base; Jelly Bean is 21.3%, and earlier versions sum to 4.9% (https://developer.android.com/about/dashboards/index.html). Web browsing of sensitive data on those platforms is unsafe if the system libraries are involved.

This problem is exacerbated by wireless carriers who still stock KitKat, Jelly Bean and earlier versions. Even the largest of carriers are guilty of this activity, and they include no disclosure that these older OS versions have weak, exploitable encryption and a slew of other flaws, which place them at a severe disadvantage for sensitive traffic. Some carriers spend far more effort in locking phones with bootloaders that require kernels bearing digital signatures than they have ever spent on security patches.

Original Equipment Manufacturers (OEMs) cause equal trouble. Although many "stock browsers" in various versions of Android use the system WebKit, some OEMs build separate versions of WebKit for their branded browsers that exhibit the same (lack of) support as shown by Google. OEM/vendor browsers also cannot be trusted with sensitive data.

The US Federal Communications and Trade Commissions (FCC and FTC) have announced a joint investigation into Google and its partners over the lack of security updates for Android (http://www.theregister.co.uk/2016/05/09/fcc_ftc_android_updates), which may result in future architecture changes but is unlikely to secure the older releases. Ideally, the FCC would compel carriers and

OEMs to release signing keys for phones that have gone without security patches for more than six months, giving users of abandoned phones the option of third-party security support.

Had Microsoft taken the final Trident rendering engine from Windows XP's Internet Explorer and locked it to 50% of the Windows user community while actively preventing updates, the condemnation would have been fierce and brutal. The time has come to recognize that what Google has done is far worse—XP and KitKat support ended within a few months of one another, but Microsoft does not allow XP to proliferate through third parties as Google does with its orphaned products.

It was likely with some measure of relief that Apple and the WebKit team greeted the news that Google was forking the code, forming the BLINK engine and leaving the project (http://www.wired.com/2013/04/blink). Google has the worst security record of any large WebKit implementation. Although other Linux distributions also lag on WebKit security (https://blogs.gnome.org/mcatanzaro/2016/02/01/on-webkit-security-updates), none has the ubiquity of Android. Google's departure will only improve WebKit's security standing.

The Plague Spreads

A number of browsers in the Google Play Store are reputedly faster and more feature-rich than Chrome. Some even assert greater security, which I soon will refute here. Chrome is generally seen as a conservative choice by app review sites, and it is rarely listed in first place in Android browser reviews.

These faster browsers often simply wrap new UI controls around the system WebKit, and thus inherit all of the security flaws of the Android version upon which they run.

Browsers that run with degraded security, as tested on Jelly Bean and reported by the Qualys SSL Scanner (https://www.ssllabs.com), include Apus, Apus Turbo, Best Browser, Boat Browser, Brave (Link Bubble), CM Browser, Dolphin, Dolphin Zero, Easy, Flynx, Flyperlink, Ghostery, Javelin, Maxthon, Mercury, Naked Browser, Next Browser, Ninesky, Safe Browser and UC Browser. **These browsers are to be avoided on 4.4 KitKat and lower versions of Android.** See Table 1 for details.

Table 1. Browser Security Comparison

Browser	Version	FREAK	Logjam	POODLE	RC4	TLS 1.2
Apus	1.4.9	Vulnerable	Vulnerable	Vulnerable	Yes	No
Apus Turbo	1.4.7.1003	Vulnerable	Vulnerable	Vulnerable	Yes	No
Best	1.5.1	Vulnerable	Vulnerable	Vulnerable	Yes	No
Boat	8.7.4	Vulnerable	Vulnerable	Vulnerable	Yes	No
Boat Mini	6.4.6	Vulnerable	Vulnerable	Vulnerable	Yes	No
Brave (Link Bubble)	1.9.19	Vulnerable	Vulnerable	Vulnerable	Yes	No
CM Browser	5.20.44	Vulnerable	Vulnerable	Vulnerable	Yes	No
Dolphin	11.5.6	Vulnerable	Vulnerable	Vulnerable	Yes	Yes
Dolphin Zero	1.3	Vulnerable	Vulnerable	Vulnerable	Yes	No
Easy	3.0.2	Vulnerable	Vulnerable	Vulnerable	Yes	No
Firefox	46.0.1	Safe	Safe	Safe	No	Yes
Flynx	2.0.1	Vulnerable	Vulnerable	Vulnerable	Yes	No
Flyperlink	1.36.RC4	Vulnerable	Vulnerable	Vulnerable	Yes	No
Ghostery	1.3.3	Vulnerable	Vulnerable	Vulnerable	Yes	No
Google Chrome	50.0.2661.89	Safe	Safe	Safe	No	Yes
Javelin	4.1.11	Vulnerable	Vulnerable	Vulnerable	Yes	No
Maxthon	4.5.9.3000	Vulnerable	Vulnerable	Vulnerable	Yes	No
Mercury	3.2.3	Vulnerable	Vulnerable	Vulnerable	Yes	No
Naked	1.0 Build 114	Vulnerable	Vulnerable	Vulnerable	Yes	No
Next	2.11	Vulnerable	Vulnerable	Vulnerable	Yes	No
Ninesky	5.2.0	Vulnerable	Vulnerable	Vulnerable	Yes	No
Opera	36.2.2126.102826	Safe	Safe	Safe	No	Yes
Opera Mini	16.0.2168.103662	Vulnerable	Vulnerable	Vulnerable	Yes	No
Power	48.0.2016042602	Safe	Safe	Safe	No	Yes
Puffin	4.7.4.2567	Safe	Safe	Safe	Yes	Yes
Safe Browser	1.17	Vulnerable	Vulnerable	Vulnerable	Yes	No
UC Browser	10.9.8.770	Vulnerable	Vulnerable	Vulnerable	Yes	Yes
Yandex	16.2.2.7988	Safe	Safe	Safe	No	Yes
Yolo	1.0.1.83	Safe	Safe	Safe	No	Yes

Surprisingly, JellyBean does include the latest TLSv1.2 encryption protocol, but it is disabled by default. There is a procedure to enable it that a developer must follow to secure an application with this feature (http://blog.dev-area.net/2015/08/13/android-4-1-enable-tls-1-1-and-tls-1-2). A few of the browsers mentioned above have done so, but many have not, either out of ignorance or sloth. There are extensive options for detailed cipher control (https://developer.android.com/reference/javax/net/ssl/SSLSocket.html) that can be used to pass more of the Qualys SSL Labs tests with the standard WebKit (by disabling SSLv3, RC4, export ciphers and so on), but none of the tested "rebadged-WebKit" browsers listed above have done so (likely as no best-practice details the procedures).

Beginning with Android 2.3 Gingerbread, Google made the surprising decision to alter the preferred cipher order, switching to RC4-MD5 from Android 2.2 Froyo's AES256-SHA1 (https://op-co.de/blog/posts/android_ssl_downgrade). Although it appears that this was done to mirror the Java standards, the impact is described as "a sign of horrible ignorance, security incompetence or a clever disguise for an NSA-influenced manipulation". This flaw remained in place for the initial Android JellyBean 4.1 release, although it appears to have been corrected by release 4.3.

Moving onward to third parties, the CM Browser application specifically advertises that it is "Secure: Malicious & Fraud Protection" with the "#1 antivirus engine...which can protect you from malicious threats." Despite these claims, its use of the WebKit system exposes it to all the problems of the older platforms. The Safe Browser advertises anti-spyware/-virus, but the SSL test results surely negate any dubious benefit from a malware host list. The Ghostery browser appears to be available as a plugin for Firefox—use it in this manner for better encryption support.

Opera Mini deserves special mention. In the default configuration, the Qualys SSL scanner detects the "Presto" rendering engine, not WebKit, and it passes all of the security tests. However, if the "data savings" setting is switched from "extreme" to "high", then WebKit is detected (not Presto), and all of the tests fail. It appears that, while Presto is active, all of Opera Mini's browser traffic is routed through Opera's servers for pre-rendering and compression. This is deceptive, so Opera Mini's failing grade is reported here.

All of the browsers tested were free, but some have "ad-free/pro" versions that must be purchased. Be sure to test via the Qualys SSL scanner before paying for any Android browser to avoid purchasing a failure.

This problem does not end with browsers. A number of applications will render web pages as a small subset of their function, and those rendered pages also are unsafe. For example, the Tinfoil for Facebook application has an option to "Open links inside app". Those are opened with the system WebKit. Banking, securities and finance apps may well do the same. If you run an Android app that handles sensitive data, ask the developers if they use WebKit/WebView. If so, do not use it on Android 4.4 or below.

Safe Harbor

From the web browsers above that pass all standards tests, Google Chrome likely has the largest installed base, as it is bundled on many Android devices when they are sold. Many will be tempted to use it as their secure browser. This is likely the wrong choice, for a number of reasons:

- Although Chrome allows malware blocking as an extension in other operating systems, this feature has been removed from the Android version. Google has not stopped with merely denying this single feature in Chrome, but has gone further and removed third-party malware/adblock applications from the Play store (http://lifehacker.com/5990448/google-has-started-removing-ad-blockers-from-the-play-store), further endangering the Android community. Such a move demonstrates clearly that advertising telemetry is more important than security for the Android architects.

- Google Chrome is (partially) closed source, and users have no idea what it might be harvesting when it is installed and in use, especially on the Android platform where Google likely feels a sense of entitlement.

- Instead of Chrome, it is possible to load the open-source Chromium browser on Android (to which Google adds closed-source components prior to distribution). The getChromium

application on F-Droid can install this precursor browser (https://f-droid.org/repository/browse/?fdid=com.anddevw.getchromium). Still, Chromium lacks malware filters.

It's plain that the safest browser on Android should be open source, include malware block capability, receive regular updates and not be based on WebKit in any way to ensure that it does not clandestinely utilize vulnerable Android components. The obvious browser that meets these qualifications is Firefox. This is not to imply that Firefox is a perfect browser. It famously lacks a sandbox, which has not helped its reputation (http://www.extremetech.com/computing/178587-firefox-is-still-the-least-secure-web-browser-falls-to-four-zero-day-exploits-at-pwn2own). However, it is far better than the majority of its peers on Android. It also has a large extensions library that includes several malware/adblock options, which Google has confirmed will never come to Chrome for Android (http://www.omgchrome.com/chrome-android-extensions-not-planned-ama), which might be for the best, as criminals have been soliciting Chrome extension app authors to abuse Chrome users (https://software-gunslinger.tumblr.com/post/143004937538/google-chrome-a-history-of-crime).

To address other Android components that present a danger to safe usage, consider the following:

- /system/lib/libstagefright.so — this library has been compromised in attacks transmitted by web pages and media sent by MMS (https://www.schneier.com/blog/archives/2015/07/stagefright_vul.html). Some have suggested that Firefox is not vulnerable to StageFright exploits (http://android.stackexchange.com/questions/120914/why-is-firefox-not-vulnerable-to-stagefright).

- /system/lib/libc.so — the core standard library for the C programming language was taken largely from OpenBSD, then neglected for years. Recent update efforts by the maintainers expose the sad state to which the code had fallen (http://undeadly.org/cgi?action=article&sid=20140506132000): "I've seen what a mess things were when we diverged (and how many bugs went unfixed in Android despite having been fixed for years upstream)."

YOU DO NOT WANT TO TRUST YOUR BANK ACCOUNT TO AN OPERATING SYSTEM WHERE NO ONE CARES THAT HUNDREDS OF BUGS ARE IGNORED, REGARDLESS OF ENTHUSIASM, HYPE OR ATTRACTIVENESS.

- The Linux kernel itself — more a demonstration of policy than security, Google's source contribution to the kernel for Android was erased by a kernel maintainer who announced the reason why (http://m.theregister.co.uk/2010/02/03/android_driver_code_deleted_from_linux_kernel): "In short, no one cared about the code, so it was removed" (http://www.kroah.com/log/linux/android-kernel-problems.html?seemore=y). For a definitive kernel security weakness that, for many users, will never be patched, the example of Pinkie Pie's Towelroot is the best known (http://geeksided.com/2014/06/16/towelroot-exploit-reveals-security-nightmare-android). This flaw allows any application to gain root privilege on some KitKat devices and many earlier versions.

You do not want to trust your bank account to an operating system where no one cares that hundreds of bugs are ignored, regardless of enthusiasm, hype or attractiveness. Assuming that you must use your Android device to process such sensitive information, it is likely better to use a mobile HTML site in a safe browser rather than a local app. If circumstances force you to use an app, prefer an app developer that recognizes Android libraries for the security minefield that they are and thus avoids using OS features that can compromise your data.

Conclusion

Users of modern, enterprise Linux are accustomed to five-year support cycles on platforms with regular security updates that rely on fine-grained package databases, allowing for easy reversal of an individual patch. All of this is available for free from multiple vendors, although paid support is available through several avenues.

For such individuals, Android releases closely resemble unsupported development snapshots. In the rare events that Android updates do arrive, they come as giant .ZIP files that are slapped over (/system)/bin in an irreversible manner. Updates commonly apply new locks over boot firmware, fencing users out of hardware that they purchased and own.

Should a new enterprise Linux distribution appear exhibiting these behaviors, no one would use it. Should an existing distribution implement these policies, the exodus of the user community would likely not be as fast as the blizzard of lawsuits that tore the vendor to bankruptcy.

A day will come when consistent security policies and updates are required on embedded, mobile, workstation and server platforms. For the good of the computer sciences and the people who use them, let's hope that day comes soon.■

Charles Fisher has an electrical engineering degree from the University of Iowa and works as a systems and database administrator for a Fortune 500 mining and manufacturing corporation. He has previously published both journal articles and technical manuals on Linux for *UnixWorld* and other McGraw-Hill publications.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

peer1 hosting

Where every interaction matters.

break down
your innovation barriers

power your business to its full potential

When you're presented with new opportunities, you want to focus on turning them into successes, not whether your IT solution can support them.

Peer 1 Hosting powers your business with our wholly owned FastFiber Network™, global footprint, and offers professionally managed public and private cloud solutions that are secure, scalable, and customized for your business.

Unsurpassed performance and reliability help build your business foundation to be rock-solid, ready for high growth, and deliver the fast user experience your customers expect.

Want more on cloud?
Call: 844.855.6655 | go.peer1.com/linux | Vew Cloud Webinar:

Public and Private Cloud | Managed Hosting | Dedicated Hosting | Colocation

RADIO FREE LINUX

It's a Windows world at your favorite radio station, but in DC, Linux is coming on strong.

ALAN PETERSON

PREVIOUS
Feature: Android Browser Security

NEXT
Feature: The Tiny Internet Project, Part II

You would have a difficult time today finding a radio station that was all-live and did not have some kind of computerized, automated means of storing and playing audio.

In a bygone era, hands-on media management of records and tape cartridges ("carts") was the way of the world. Through the years, turntables, CD players and "cart decks" gave way to mechanized playback of content by banks of reel-to-reel machines under crude sequential control. The earliest fully computerized "audio on hard drive" systems for radio stations and networks ran generally under DOS, evolving into the elaborate Windows-based systems in use everywhere today.

So What about Linux?

Do a web search for "Linux radio station", and the pickings are slim indeed, with most sites promoting instead ham radio software or streaming audio players, and a handful devoted to setting up a streaming web radio station—including one such optimistic article in *Linux Journal* some 15 years ago (see "Running a Net Radio Station with Open-Source Software", January 2001: http://www.linuxjournal.com/article/4397).

Unfortunately, much of this documented interest took place a decade or more in the past via domains like opensourceradio.com that are no longer with us. A few projects persevere, but a good number of postings are similarly dated. The fact is, there are more Linux-based ways to stream and listen to radio stations than there actually are the means to broadcast and control them.

Fortunately, the choices today are getting better. Transmitter manufacturer Nautel incorporates Linux in its AUI. Broadcast automation and media management systems, such as Airtime and DIY-DJ, were designed around Linux from the start. Many Windows-based commercial automation systems seem happy when networked with Linux servers. But, as for in-studio hands-on control of program execution, there still is a way to go.

Off to Washington

In and around the nation's capital, actual over-the-air radio programming driven by Penguin power is a reality, and you already may be listening to it wherever you are.

Back around 2002, Fred Gleason, then the director of engineering for the Salem Communications radio cluster in Arlington, Virginia, saw the need for an open-source radio automation system: one that was nimble enough to handle the turn-on-a-dime demands of radio station programming, flexible enough to be modified and rewritten as the demands on it grew, and as crash-proof as possible to keep the music flowing and the commercials rolling. Working with partner and automation expert Scott Spillers, the two planted the seeds for what became the Rivendell radio automation suite. Rivendell is now under the banner of Paravel Systems, with Gleason as President and Chief Developer, and co-founder Spillers

In and around the nation's capital, actual over-the-air radio programming driven by Penguin power is a reality, and you already may be listening to it wherever you are.

holding the VP reins. The company offers Rivendell as custom-built turnkey hardware, with a free software version as a downloadable iso file.

Rivendell originally was crafted for SUSE Linux but now runs under CentOS. It is an artistic mix of custom code and popular existing libraries and applications. Gleason used the Qt Toolkit to create the user interface, allowing it to run on anything right down to a Raspberry Pi; MySQL to maintain and manage the audio database; the Apache Web Server, ID3Lib, a couple CD rippers and the X11 Window System, among its many components.

Gleason's handcrafted code includes the GlassSuite collection of audio tools for Icecast and HTTP Live Streaming (HLS), and what he calls "a slew of code" to perform audio format conversions whenever new audio is imported into the station's audio library. Early in the development of Rivendell, SOX handled the format conversion, but Gleason noticed a slowdown due to "limited process control with the multi-stage command pipeline. This was before SOX had its library interface." His code performs

the format conversions natively, without any shell outs.

Today, there are three major broadcast operations based in Washington, DC, that employ Rivendell—Radio America Network, Salem Communications DC and Radio Free Asia—with listeners everywhere in both Eastern and Western hemispheres. This is not to say that Rivendell is available only to major players. Many smaller users have come to depend on Rivendell as well. High school educational radio station WKHS-FM in Worton, Maryland, has been a Rivendell user since 2012, opting for a 32-bit Ubuntu-based appliance distro called RRABUNTU. A recent count showed more than 40 US AM and FM radio stations with a Rivendell installation in operation. A dozen miles south of DC, "Rolling Valley Radio", a low-power license-free community station uses the 64-bit CentOS version to broadcast its short-range signal to a small enclave of four or five suburban blocks.

Behind the Scenes

The Radio America Network in Arlington, Virginia, is a creator, distributor and syndicator of national radio programming, including conservative talkers Dana Loesch and Chad Benson. A stroll through the bullpen reveals a somewhat non-conservative and very creative attitude: a disco mirror ball hangs from the ceiling of a video production suite. Ukuleles are available for strumming while audio mixdowns are rendering. Coffee urns are labeled "Folger's" and "Creosote", and every Friday is Hawaiian Shirt Day.

But on the air, it's all business. The network runs 24/7, delivering programming via satellite to affiliates from Guam to the tip of Maine. With hundreds of thousands of listeners across the US, dependability and stability are critical and downtime is not an option. So with the exception of a dozen or so Windows office machines and a quartet of Mac video editors, the plant is populated by CentOS desktops, in-house-built servers and Rivendell automation computers in five studios. The network also streams its programming over the internet using Glasscoder and logs every minute of broadcast audio around the clock on a Linux computer nicknamed "Delorean".

Not being locked in to a single commercial system allows for a lot of flexibility. For instance, in studios that use more than one computer, actual physical desktop space is at a premium. Synergy software is used across all machines: one keyboard and one mouse move seamlessly across the screens and leapfrog from one computer to the next. And for assistant

Figure 1. A screenshot of the Ardour audio workstation performing a mix on a ten-year-old Toshiba laptop.

engineers and technicians lacking mad command-line skills but who need to tend to the system, Webmin is in use on several computers.

Some staffers experiment on their own. Show producers in the bullpen employ Linux desktop computers running Audacity audio software to collect and clean up sound bites for various talk hosts. The same hosts surf the web and manage their call-screening software during their shows on CentOS machines. The network's production director favors UbuntuStudio on an older laptop, using Ardour to do preliminary mixes on several prerecorded weekend lifestyle shows (Figure 1). And engineers recently have begun exploring LinPhone as a replacement for Skype and ISDN as a two-way studio link with hosts at remote locations.

Radio America adopted Rivendell sometime around 2005 and has served as a proving ground for subsequent upgrades and improvements. It is a classic example of Linux radio automation working, and it's working well.

Operations Director Rich McFadden, a former user of numerous radio automation systems, said "Rivendell—and Linux—gives us the most stable environment to stay on the air without interruption."

Figure 2. The master control console at the Salem radio cluster in Arlington, Virginia: Rivendell radio automation keeps four signals running on three terrestrial stations and satellite radio. (Photo Credit: Chris Roth.)

Across the Potomac River from the nation's capital in Rosslyn, Virginia, the Washington DC cluster of Salem Communications—WAVA-FM, WAVA(AM) and WWRC(AM)—also is driven by Linux and Rivendell. The Salem cluster recently closed up its old facility (on the air since 1996) and moved to all-new studios and offices, with Rivendell tied in closely to its new complement of Axia digital audio consoles and engines. The software commands the consoles to perform specific functions and automatically reroute audio to different destinations, and because of its open-source nature, can be changed and rewritten at almost anytime. Director of programming Chris Roth noted that "It interfaces very well with our Axia/Livewire [digital broadcast consoles], which makes our operation run very smoothly." The master control console shown in Figure 2 handles three stations and the Sirius XM channel programmed by Salem.

Other Possibilities

On-air execution is not the only opportunity for Linux—and open-source software in general—to shine in radio broadcast operations. Both the LibreOffice and Apache OpenOffice suites have replaced or augmented Microsoft Office when cost containment becomes an issue. Creative DJs who spin and mix multiple songs into sonic tapestries are exploring MIXXX, a free music performance tool that rivals commercial DJ applications. An Icecast-compatible encoder with the memorable name B.U.T.T. (Broadcast Using This Tool) has found many a home, including the college webcasting station E-Radio WMCR based at Montgomery College in Maryland.

With most US radio stations owned by large corporations—several hundred each at a time—it is likely the dependence on Windows-type computer systems for radio broadcast will persist, given the need for large and fast equipment deployments, compatibility with other systems and familiarity of operation. Fortunately, the Linux alternative exists, and although it may not enjoy the exposure and popularity that Win-based broadcast products do, it is out there and being embraced by those who are looking for stability, versatility and affordability.

DC today. Tomorrow, who knows?■

Alan Peterson is a DC-based broadcaster, writer, audio engineer and has been advocate of Linux in radio broadcasting since 2007. His presentation on open-source software, "Run Your Entire Station on Two DVDs", has been a staple of the Intercollegiate Broadcasters' conference and convention for six years. Write him at alanpdarsen@gmail.com.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

drupalize.me

Instant Access to Premium Online Drupal Training

- *Instant access to hundreds of hours of Drupal training with new videos added every week!*
- *Learn from industry experts with real world experience building high profile sites*
- *Learn on the go wherever you are with apps for iOS, Android & Roku*
- *We also offer group accounts. Give your whole team access at a discounted rate!*

Learn about our latest video releases and offers first by following us on Facebook and Twitter (@drupalizeme)!

Go to http://drupalize.me and
get Drupalized today!

THE TINY INTERNET PROJECT, Part II

Turn an old PC into a virtual-machine host using Proxmox and KVM.

JOHN S. TONELLO

PREVIOUS
Feature:
Radio Free Linux

NEXT
Doc Searls' EOF

In the May 2016 issue, I introduced the idea of the Tiny Internet Project, a self-contained Linux project that shows how to build the key pieces of the public internet on a single computer using one or two old computers, a router and a bunch of Linux software. In this second part, you'll learn how to build the virtual-machine host—using Proxmox—and deploy your first server. Later, in Part III, you'll build a template to make DNS and email hosts, a website and a Linux distribution mirror.

You'll need two separate computers for this project. This first is your "administration PC". It's any desktop or laptop with network access, a graphical browser (like Firefox or Chrome) and at least one USB port. The second machine will become your virtual machine host.

The central idea is to build a sort of internet-in-a-box with common Linux servers and use the setup to teach young people or newcomers about Linux. Using virtualization software, you'll deploy several servers that will handle domain names, email, web pages and more—all on a single piece of hardware. You don't need anything new or fancy. I built the prototype on a six-year-old Velocity Micro desktop with an Intel i3 processor, 8GB of RAM, two network cards and a couple 1TB drives.

Choosing Your Hardware

First off, you need a computer that supports virtualization—meaning its BIOS, 64-bit-capable CPU and motherboard allow you to share all the system's resources with virtual machines that will run on it. You may have experimented with VirtualBox or even free versions of VMware's ESXi software. The idea here is the same: place a number of virtual servers on a single physical machine and share all of that physical machine's resources, including memory, CPU and drives.

The Linux version is called KVM for Kernel-based Virtual Machine. For this project, you'll use a pre-compiled version called Proxmox, which comes with everything you need.

To see if the computer you have in mind can become a Proxmox server, you'll need to check whether it supports virtualization. Graphical tools are available for Windows, Linux or Mac OS X, and if you've got a machine with no operating system installed, you can boot it from a USB or CD drive using any flavor of Linux to test it. See the Resources section at the

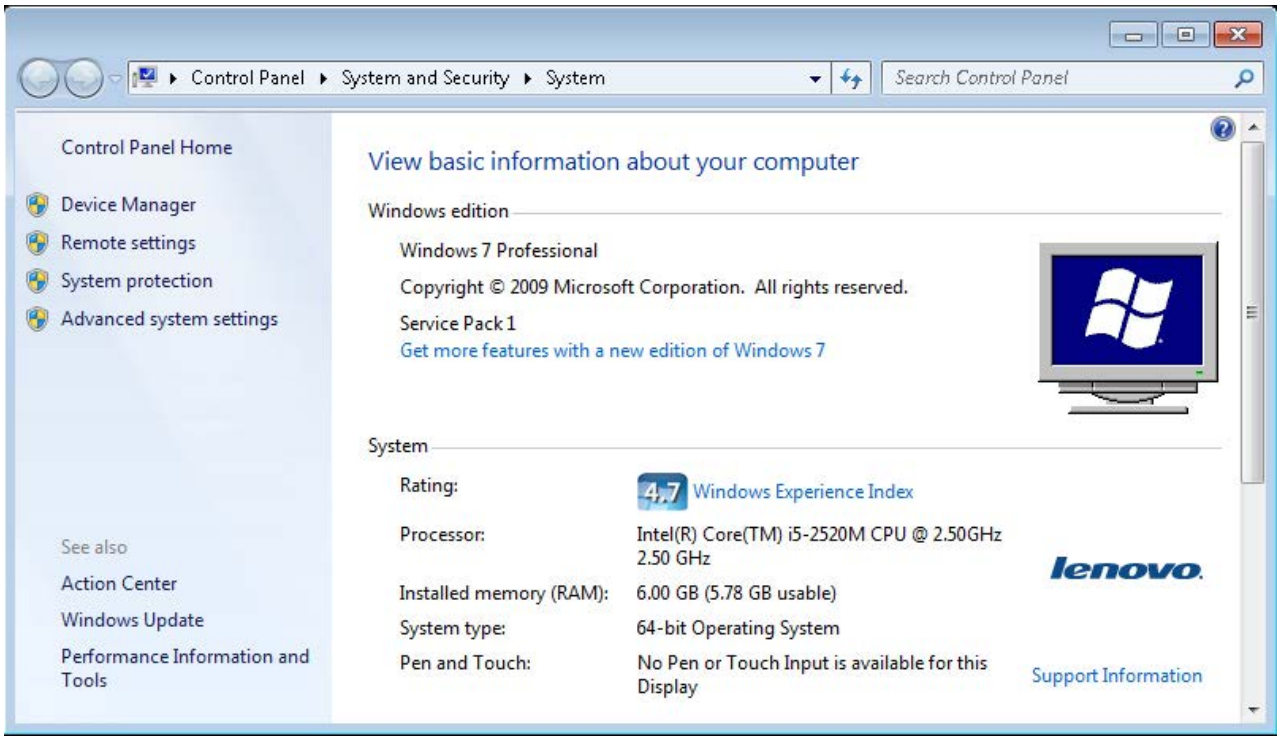


Figure 1. Windows Computer Properties

end of this article to learn how to do that.

On Windows, you can learn a lot from the main Computer properties. Right-click on Computer (on the desktop or Start menu), and look at the lower part of the window, which will look something like Figure 1.

If you see "Quad CPU", or something similar, and "64-bit Operating System", you likely have a machine that (once wiped) will work. To be more certain, you can use Intel or AMD tools to identify further your CPU's ability to support virtualization (again, see the Resources at the end of the article).

If you're planning to use an old Intel-based Apple Mac, there are many that support virtualization. Getting an older Mac to boot from USB can be a little tricky, and I won't cover those steps here. However, I have successfully installed Linux on several Intel Macs, and it runs well.

If you already have Linux installed on a desktop or server, you can use a few simple terminal commands to see if it supports virtualization. Open a terminal and enter this command:

```
$ cat /proc/cpuinfo | grep vmx
```

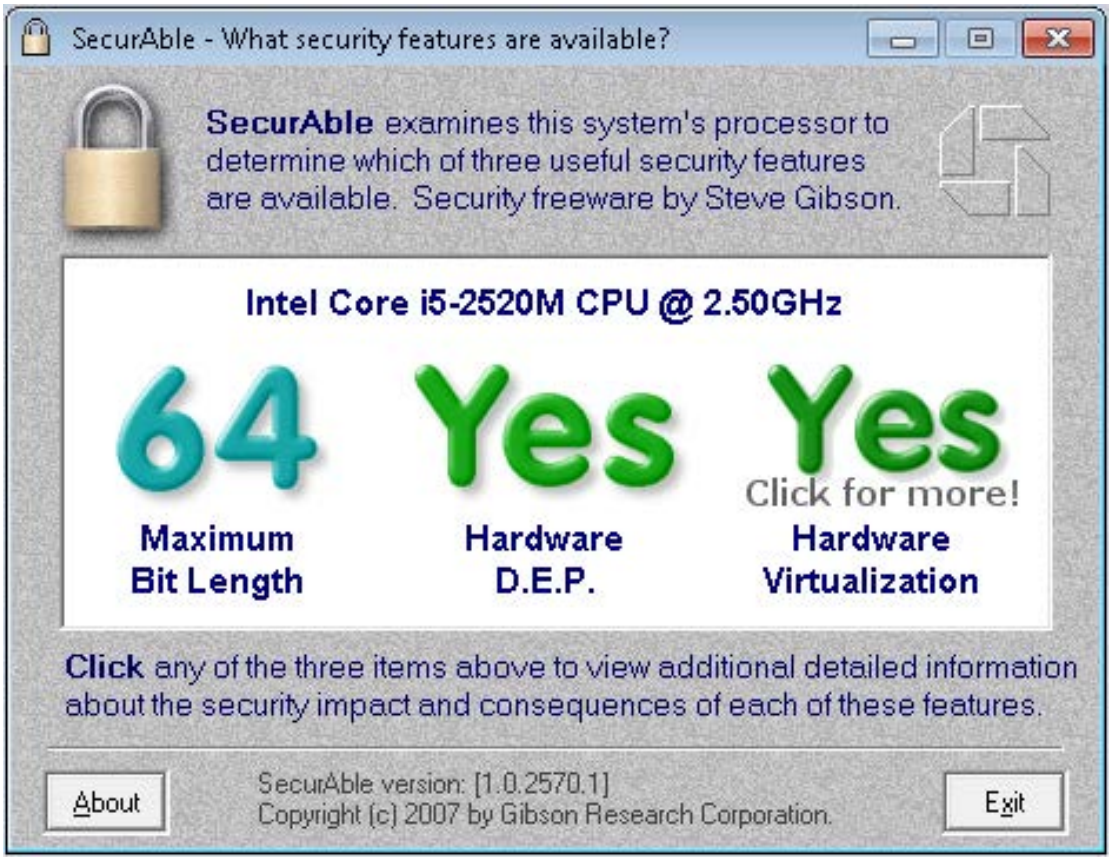


Figure 2. SecurAble helps you learn more about your processor.

If it returns something that looks like the following text (repeated several times for each CPU), you're in business and can proceed to the next step (if you're still uncertain, check the Resources for more options):

```
[flags        : fpu vme de pse tsc msr pae mce cx8 apic sep
 ➥mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
 ➥sse sse2 ss ht tm pbe syscall nx rdtscp lm constant_tsc
 ➥arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc
 ➥aperfmperf eagerfpu pni pclmulqdq dtes64 monitor ds_cpl
 ➥vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid sse4_1 sse4_2
 ➥x2apic popcnt tsc_deadline_timer aes xsave avx f16c rdrand
 ➥lahf_lm ida arat epb xsaveopt pln pts dtherm tpr_shadow
 ➥vnmi flexpriority ept vpid fsgsbase smep erms]
```

If the `flags` output includes `vmx` (possibly highlighted red in the output), you should be set. If not, check your system's BIOS. Virtualization

often is possible on a system, but it's disabled by default. Look for virtualization settings in your system's BIOS, enable it and reboot.

Although the Tiny Internet Project is designed to provide everything you need without having to access the public internet, the fact is you'll need your Linux distribution mirror to connect to servers on the outside. You can do this by setting up a proxy server (which I will cover in Part III) or by installing two Ethernet cards on your Proxmox host. Wired connections are easier to set up than wireless ones, and I don't recommend using a Wi-Fi card or USB dongle as the host's second connection.

With two cards installed, your Proxmox host will be able to talk to both public and private networks, and so will the virtual machines running on top of it.

The Virtualization Software: Proxmox

Since you're obviously most interested in Linux and free software, let's use the custom KVM Proxmox.

You can install KVM during the server set-up process on many Linux distributions, particularly Debian-based flavors. These out-of-the-box KVMs work well, but I found them a little too complicated for the Tiny Internet Project. Installing KVM on an existing machine using `apt` or `yum` works, but it's not something a newcomer can do easily.

By itself, KVM doesn't come with a GUI interface either—a shortcoming that's fine for Linux experts, but not newbies. There will be plenty of command-line work to do later, so I wanted an easy-to-use interface for KVM management, preferably something browser-based. The tools I found, such as WebVirtMgr (http://www.webvertmgr.net), were workable, but again, they were a little too complicated for a newbie to set up.

Proxmox works well because it includes the following:

- A fairly standard, if lightweight, Debian kernel.
- An easy-to-use web interface.
- A ready-made .iso that can be burned onto a USB or DVD.
- Nice tools for managing clusters and storage.

Proxmox supports clustering, which lets you set up multiple VM host machines. It also supports a variety of storage types, including local and network-based drives.

Install Proxmox

1. Download the .iso and make a bootable USB. If you've ever downloaded a Linux .iso and used it to create a bootable USB or DVD, you can breeze through this step and go right to the installation. If you're new to the process, you'll need a couple tools. The Ubuntu website provides good instructions for Windows, Mac and Linux users. (See Resources to learn more about each.)

The DVD approach is doable, but it will take longer and it'll be a little less flexible. I recommend using USB thumbdrives for creating bootable OS installers, but sometimes an older system's BIOS may not support booting from USB. In those cases, rather than banging your head against a wall trying to get a raw system to start from USB, use a DVD. If you're using a Mac, this may be the only way to get started; support for USB

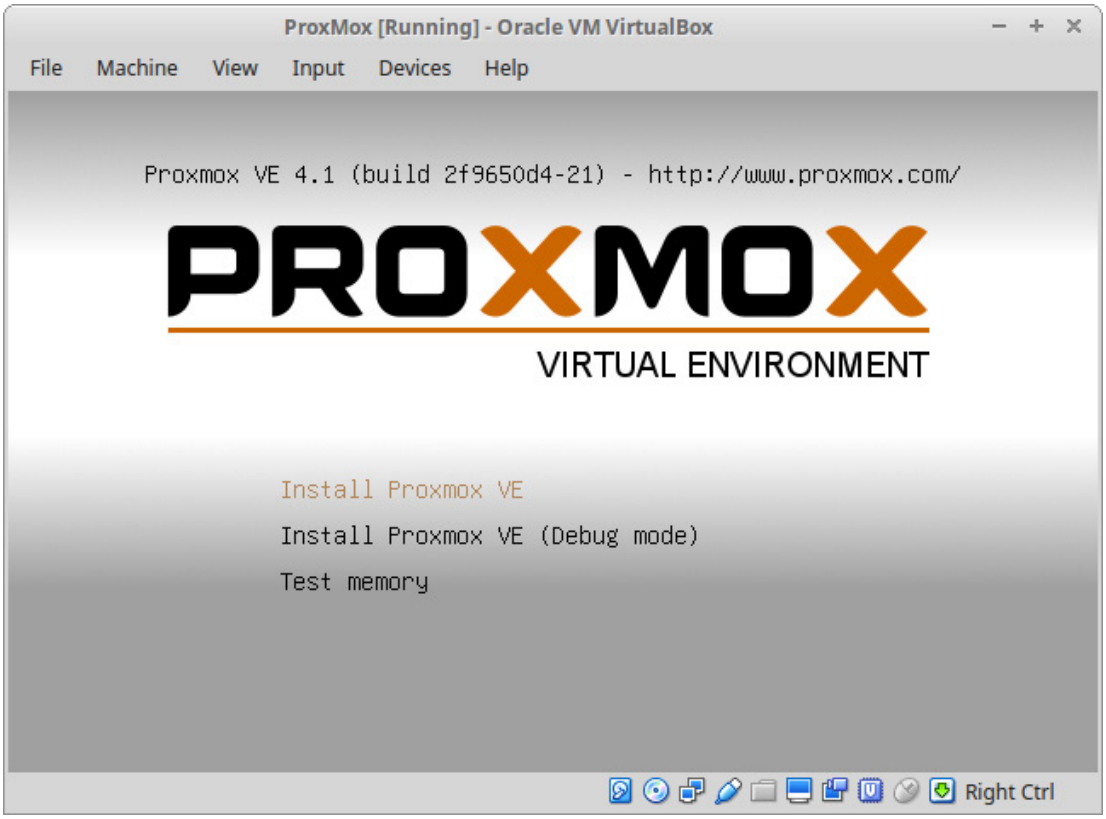


Figure 3. Initial Proxmox Installation Screen

booting on Apple hardware is a tutorial unto itself.

The .iso you want is the Proxmox Virtualization Environment (PVE). Using your administration PC, download the latest version. (It was 4.1 at the time of this writing.) The file is less than 1GB and easily fits on a 2GB thumbdrive. Burn the .iso to a USB.

Use your administration PC and go to http://proxmox.com, and download the installer.

2. Boot the Proxmox PVE installer. Remove the USB from your administration PC and use it to boot your Proxmox machine.

The initial installation screen offers three choices. Select Install Proxmox VE.

Next, choose the drive on which you want to install it. If your host machine has more than one drive, you'll get choices here. Otherwise, it will default to something like /dev/sda.

As with any OS install, this will wipe out everything you have on the

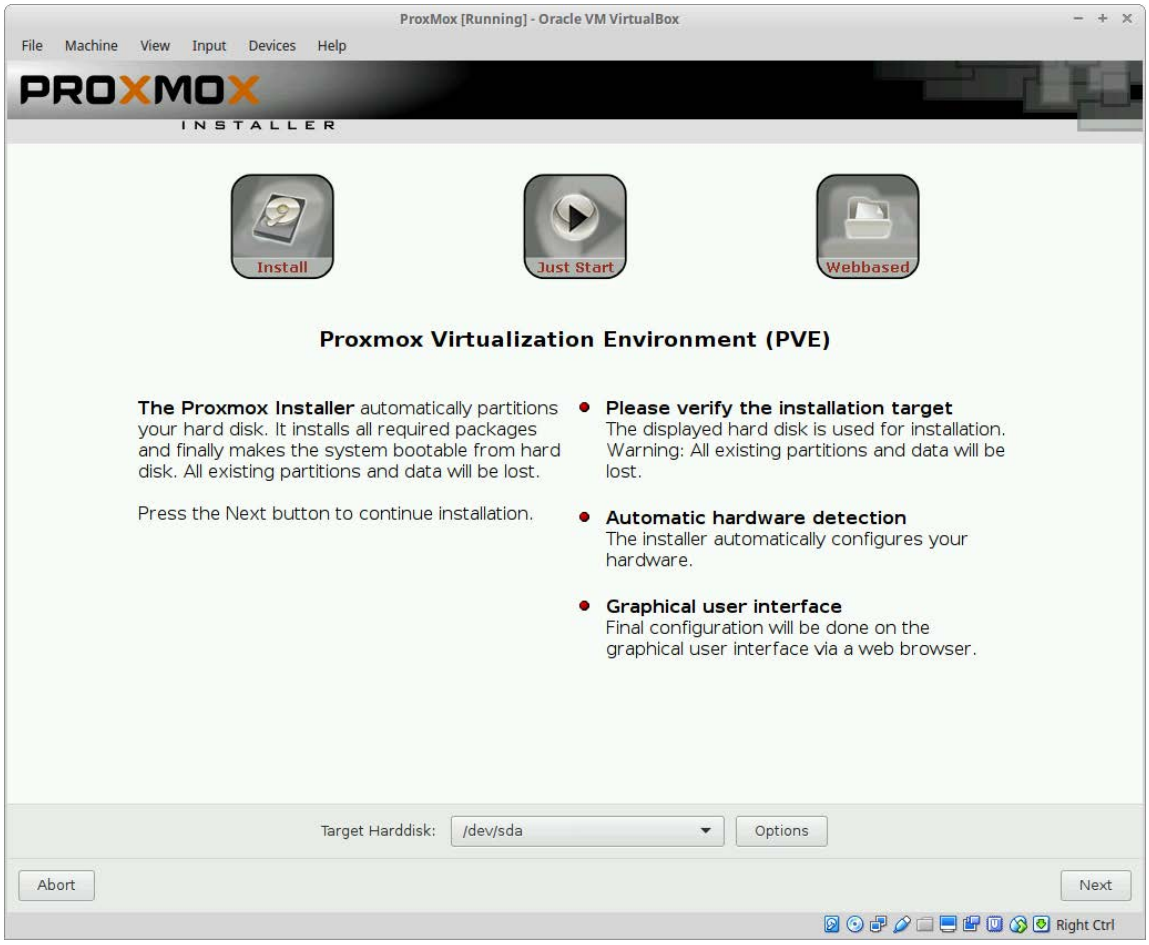


Figure 4. Choose the Installation Drive

drive. Take your time and make sure to select the correct drive.

Set your location in the next screen before moving on to set up the network, which includes the hostname (the name of the machine as it will appear on your network and in DNS), the IP address, netmask, gateway and DNS server. These won't be random; you'll need to give some thought to your future network, your VMs and the address you're going to give your DNS server.

For your private network, you'll be deploying between five and seven machines that will need their own addresses and a domain name. I used "tiny.lab" to avoid using a .com, .net, .org or any other public domain extension that could cause problems. So, with this simple plan, you'll be creating the following:

- The Proxmox host.
- Two DNS servers.
- One mail server.
- One mirror.
- Two or more web servers.

Given this schema, give the Proxmox host machine (pve in my example) the first non-gateway address, and address the others, like so:

- pve — 10.128.1.2
- dns01 — 10.128.1.3
- dns02 — 10.128.1.4
- mail — 10.128.1.5
- mirror — 10.128.1.6
- web01 — 10.128.1.7

Therefore, for the Proxmox host, set the Network Configuration settings to the following:

- hostname — pve.tiny.lab
- IP Address — 10.128.1.2
- Netmask — 255.255.255.0
- Gateway — 10.128.1.1
- DNS Server — 10.128.1.3

If you're planning to deploy multiple Proxmox hosts and form a cluster,

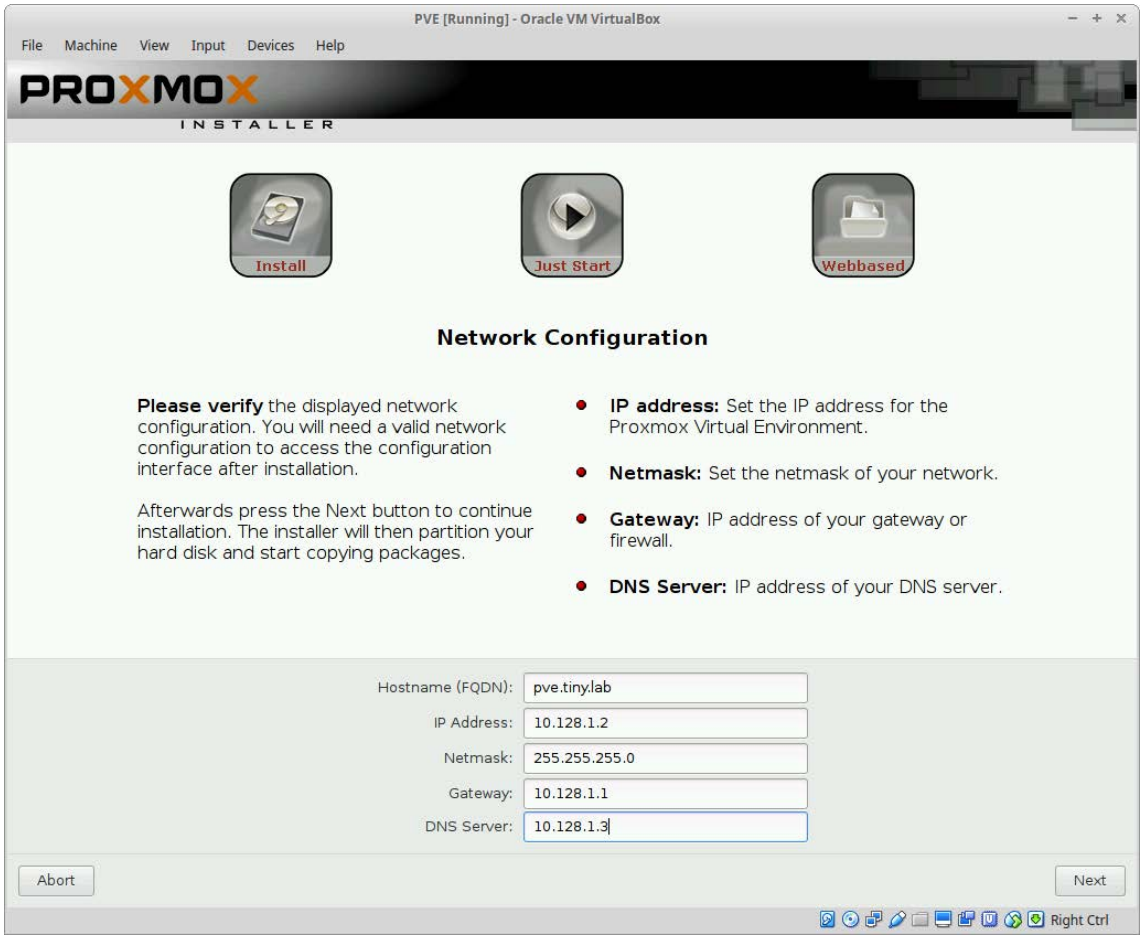


Figure 5. Proxmox Network Configuration

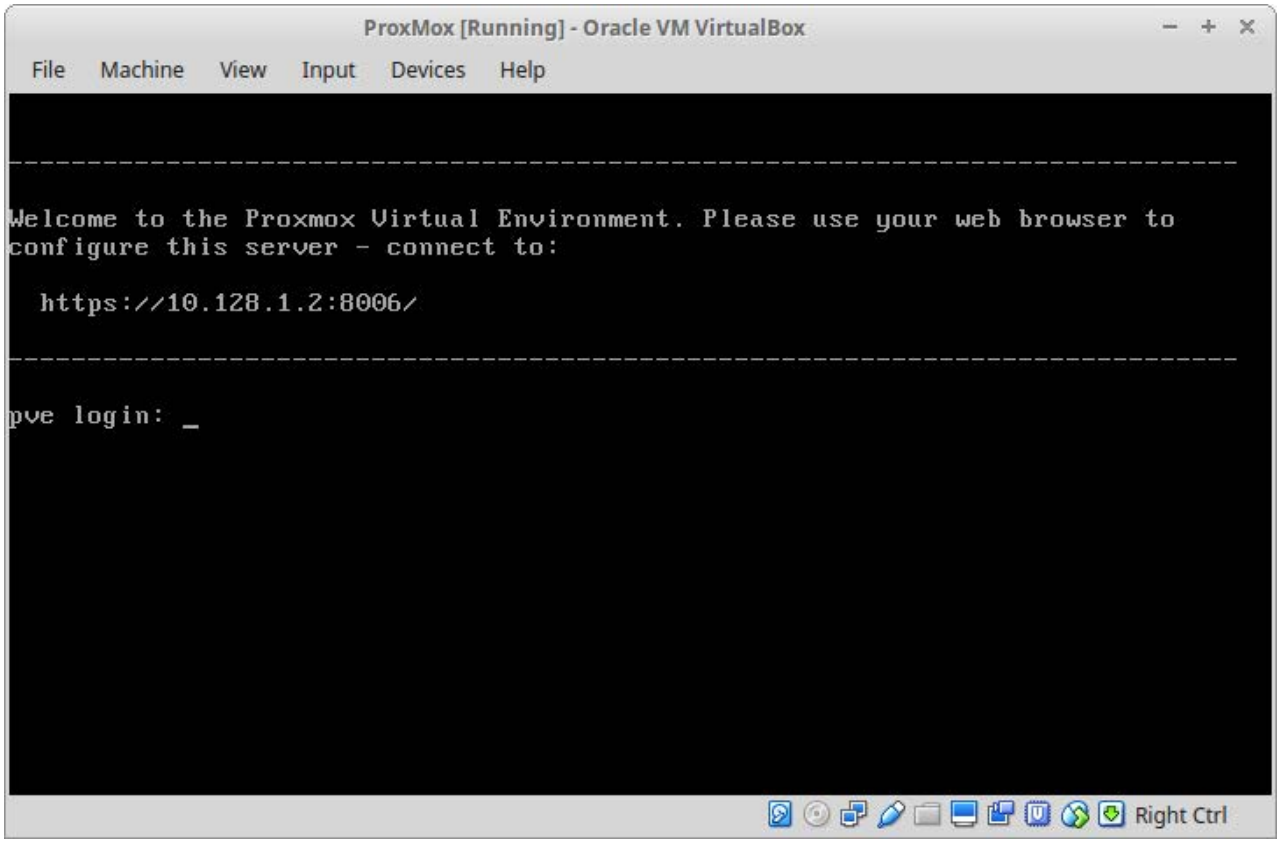


Figure 6. Proxmox is ready.

I recommend reserving the lower range of your chosen subnet—which is 10.128.1.1 to 10.128.1.255—so you can keep things logical. For example, you could give pve01 10.128.1.2 and pve02 10.128.1.3, and then start your DNS servers at 10.128.1.4.

Once you enter a password on the next screen, the installation will begin. After about five minutes, you'll be prompted to reboot. The initial boot screen looks like any Grub menu, and if all goes well, you'll end up with a login screen and a welcome telling you where to point your web browser: https://10.128.1.2:8006.

If you used a different IP address, that IP will appear instead. Later, after you've set up your domain, you'll be able to access the server at https://pve.tiny.lab:8006.

For now though, the IP is the only way in.

3. Confirm network settings on your Proxmox host. Unlike a typical Linux network setup, the Proxmox host uses bridged ports. Where you'd typically see eth0 and eth1, on Proxmox you'll see vmbr0 and vmbr1.

To get the two Proxmox host NICs to work properly, you'll need to edit the network interfaces file. From the pve login screen, log in using the user name "root" and the password you set during installation. Make a copy of the interfaces file (for safekeeping), then edit the original:

```
# cd /etc/network/
# cp interfaces interfaces.bak
# vi interfaces
```

When you first open the file, it'll look something like this:

```
[
auto lo
iface lo inet loopback

iface eth0 inet manual

iface eth1 inet manual
]
```

This isn't going to work for your purposes. You need to set up a static bridged address to eth0 and a static bridged address to eth1. First, set the interface that will communicate with your private network (tiny.lab):

```
[
auto lo
iface lo inet loopback

#iface eth0 inet manual (comment out or delete)

#iface eth1 inet manual (comment out or delete)

auto vmbr0
iface vmbr0 inet static
        address  10.128.1.2
        netmask  255.255.255.0
```

```
        dns-nameservers 10.128.1.3 10.128.1.4
        dns-search tiny.lab
        bridge_ports eth0
        bridge_stp off
        bridge_fd 0
]
```

Note that the interface bridges to eth0, but eth0 itself is not configured here. That's on purpose. Now, configure the second NIC to communicate with your public network (the network in your house or classroom that connects to the internet):

```
[
auto vmbr1
iface vmbr1 inet static
        address  192.168.1.75
        netmask  255.255.255.0
        gateway  192.168.1.1
        dns-nameservers 8.8.8.8 8.8.4.4
        bridge_ports eth1
        bridge_stp off
        bridge_fd 0
]
```

A couple things to note here. The address—192.168.1.75—is any free address on your public network. Don't pick this at random; make sure the address is available.

Also note that there is no gateway address on the first interface. That's because you can have just one gateway on a machine connected to multiple networks. Also, the dns-nameservers are set to 8.8.8.8 and 8.8.4.4, Google's public nameservers. You can use these or the nameservers provided from your ISP or school. Finally, note that this vmbr1 interface bridges to eth1.

Save the file and reboot.

4. Set up your private network devices. From this point forward, you'll do most of your work from your administration PC, not the Proxmox host.

In order to connect other machines to your Proxmox host, you'll need

to place your Proxmox server and your administration PC on the same network. That requires a network switch or router.

If you're using a router, preferably one with wireless capabilities, you can set it up with a base LAN address of 10.128.1.1. That will become its gateway address (even though you won't use it for that). If your router includes a DHCP server (most do), the device will hand out IP addresses to all the computers you attach to your tiny internet network automatically.

In your router, leave the WAN settings empty or disabled. Just set the following for the LAN:

- IP address (10.128.1.1)
- Netmask (255.255.255.0)
- Turn on DHCP
- Configure wireless security

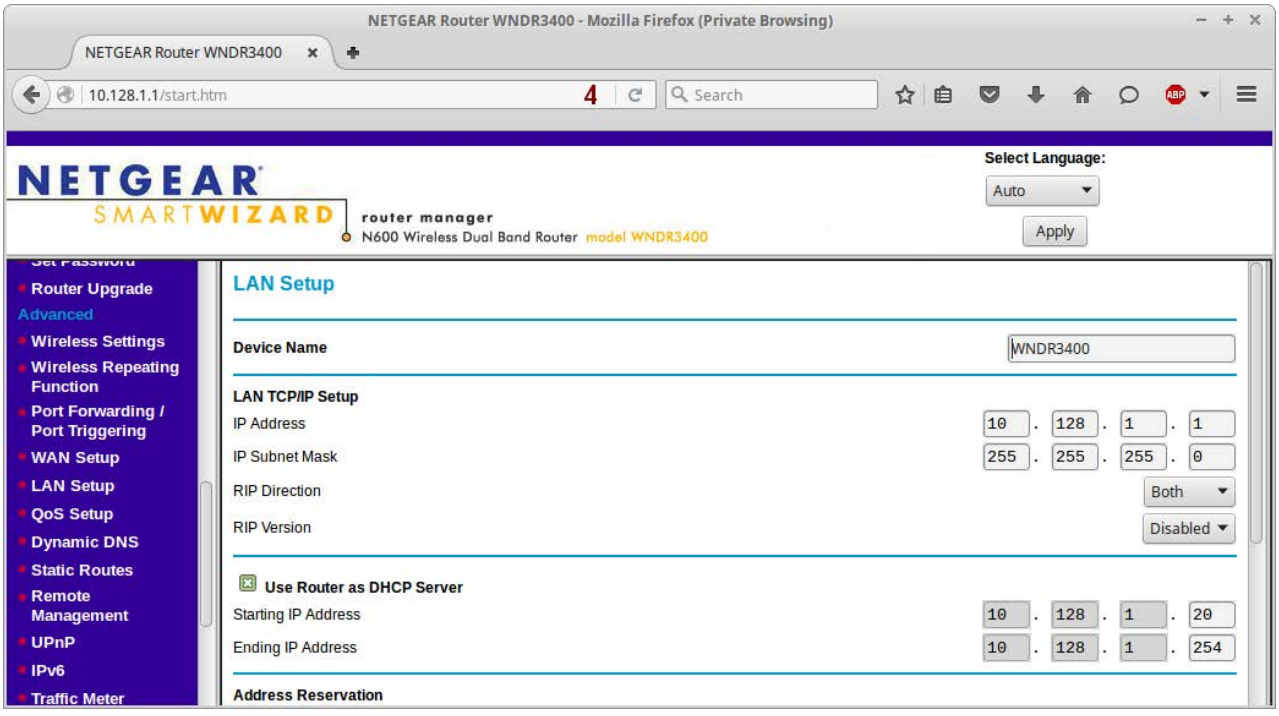


Figure 7. Router Setup

If you're using a "dumb" switch (one that is unmanaged), you'll need to set up a static IP address on your administration PC before it can join your private network. The switch won't hand out DHCP addresses on the private network, so you have to set one manually. Later, you can deploy a DHCP server, but for now, static is easy.

Be sure to give your administration PC a static address beyond the range of your server addresses, perhaps starting at 10.128.1.25 or even 10.128.1.50. The basic static-IP configuration should look something like this:

- IP address — 10.128.1.25
- Netmask — 255.255.255.0
- DNS servers — 10.128.1.3, 10.128.1.4

Again, gateway is purposely left out. In this case, your administration PC's gateway already is set to your home or school network (something like 192.168.1.1). That gateway address enables you to get to destinations on the internet. Your private-network machines won't need a gateway to talk to each other.

If your administration PC has an Ethernet port and wireless, use the wired port to connect to the public network (192.168.1.1, in this example). Use the wireless to connect to your tiny internet (10.128.1.x). That way, you'll have simultaneous access to both the public internet and your private tiny internet.

It's important to note that connecting a single computer to two separate networks can be very quirky. Fortunately, Windows 7, Mac OS X and most modern flavors of desktop Linux auto-negotiate network connections. In Linux, Network Manager (network-manager) can handle dual networks, but it can give routing priority to the wired connection. That means you should connect your internet-capable network to the wired port and use wireless to connect to your private network. In Linux Mint, I found that the opposite configuration will make web browsing slow because the system is trying to reach the internet via the wired private network first. If you must connect this way, set metrics in /etc/network/interfaces. (See Resources.)

Be aware that you may have to reboot your administration PC to get the network settings to take hold. Do that before proceeding.

5. Log in to the Proxmox GUI. Now that the PVE host is up and running, and both it and your administration PC can communicate with each other via your router (or switch), you're ready to get down to business. On your administration PC, point a browser to the address the Proxmox host offered you: https://10.128.1.2:8006. Enter the root user name and your password.

Once you click OK to dismiss the "No valid subscription warning", you'll see the main view, split into a server-manager column on the left and the main information panel on the right. The panel at the bottom gives real-time updates on actions you take, such as starting or stopping a server.

If you named your Proxmox host "pve", the server listed when you expand the Datacenter folder will be "pve". Below it is listed the local storage, which is named "local".

Explore the various tabs and become familiar with the interface. Much of it is self-explanatory.

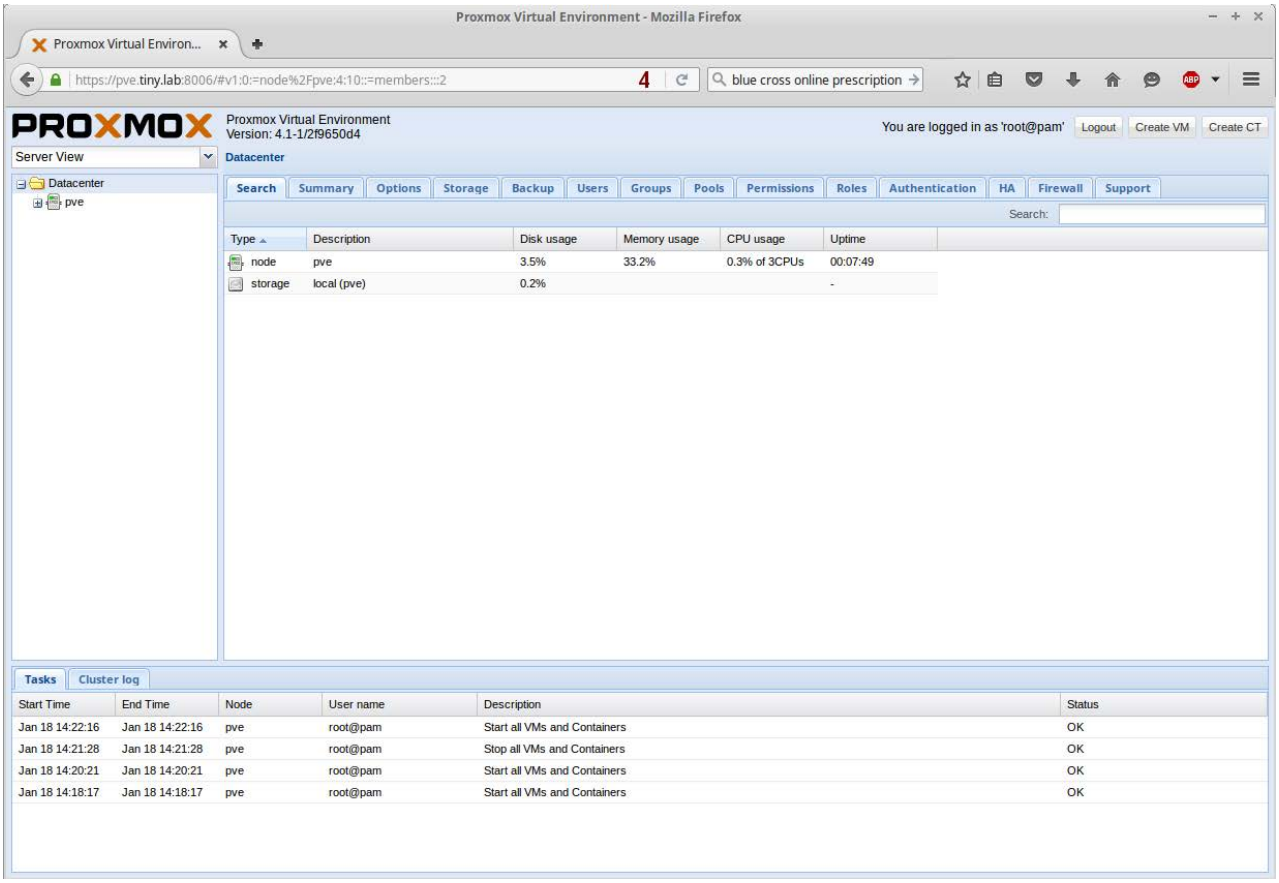


Figure 8. Proxmox Main View

6. Deploy your first server. You're now ready to deploy your first virtual machine. Before you do, decide whether you want to deploy a cluster by adding one or more Proxmox servers to your setup. If you do, you must add the second PVE host now before adding any VMs. Otherwise, if you later decide to add more PVE hosts, you'll have to delete all the virtual machines you created and basically start over.

Proxmox has some freely available templates for everything from CentOS 7, Ubuntu 15.04, Debian 7, a LAMP stack and WordPress. If your Proxmox server has access to the internet—either directly or via a proxy server—you can download and install these by clicking on "local" (your storage drive), and then choosing the Content tab and Templates.

For now, however, you're going to deploy your own virtual machine from scratch by downloading the latest .iso of Ubuntu (or whatever flavor Linux you like) and building a VM from it.

Using your administration PC, download the Ubuntu 14.04.3-server-amd64.iso.

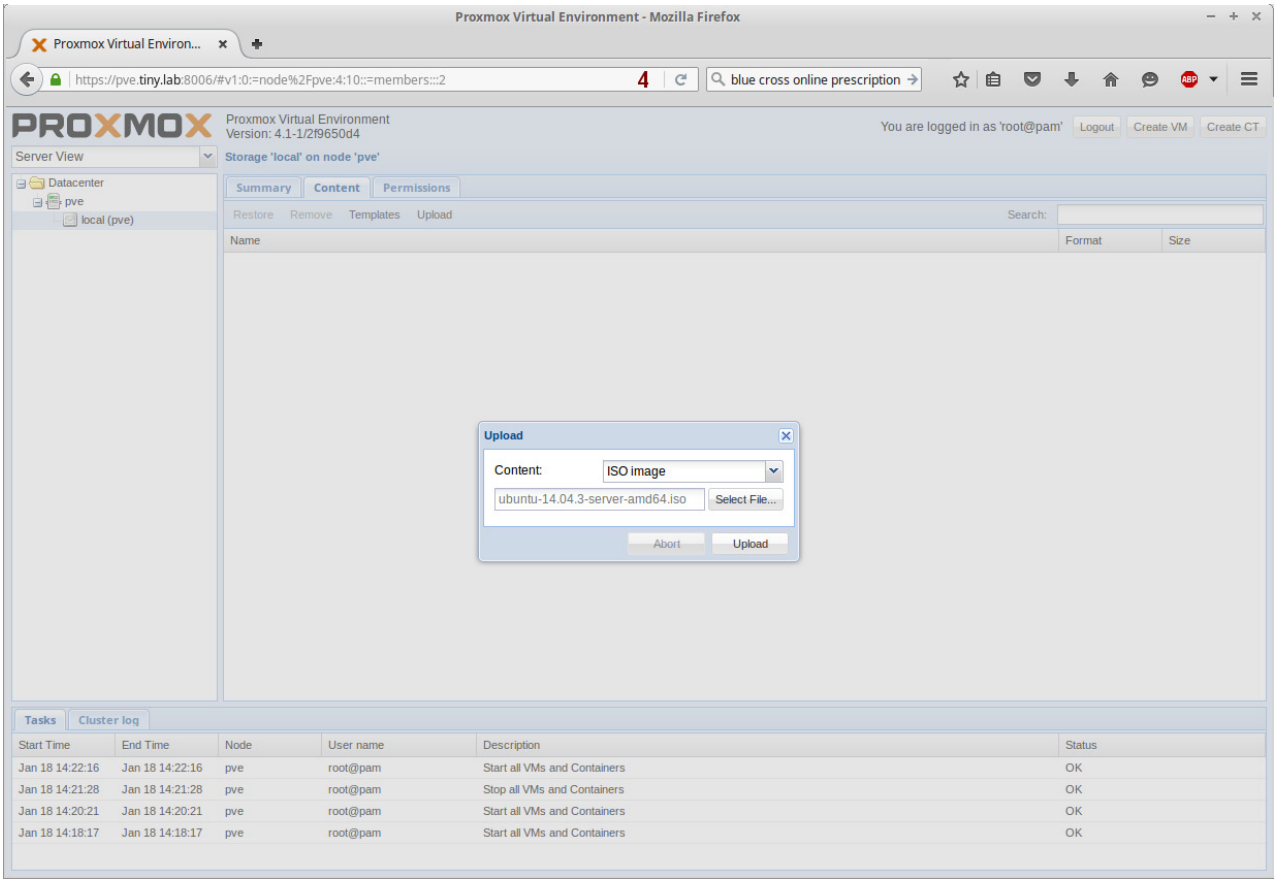


Figure 9. Uploading the .iso

This is the long-term release of the server version of Ubuntu, which you'll run without a GUI desktop. The .iso file is just more than 600MB.

To build a VM from this .iso, you'll need to upload it to your Proxmox server.

In the main PVE browser view, open Datacenter and your machine (pve) and click on the "local" storage icon. In the right-hand pane, click the Content tab and the Upload button. In the pop-up window, browse for the .iso file you just downloaded. Click OK to begin the upload.

When the upload is complete, you should see the file listed under the ISO Image list on the Content tab page. You're now ready to deploy it.

In the top right of the main Proxmox browser view, click the Create VM button. In the modal window that opens, Node will be filled in with the name of your Proxmox host ("pve"), and the VM ID will be set to 100. Each future VM will auto-increment from there. Enter a Name, such as "ubuntu", and click the Next button or the next tab.

Select the OS type—Linux 4.x/3.x/2.x Kernel (I26). Click Next and

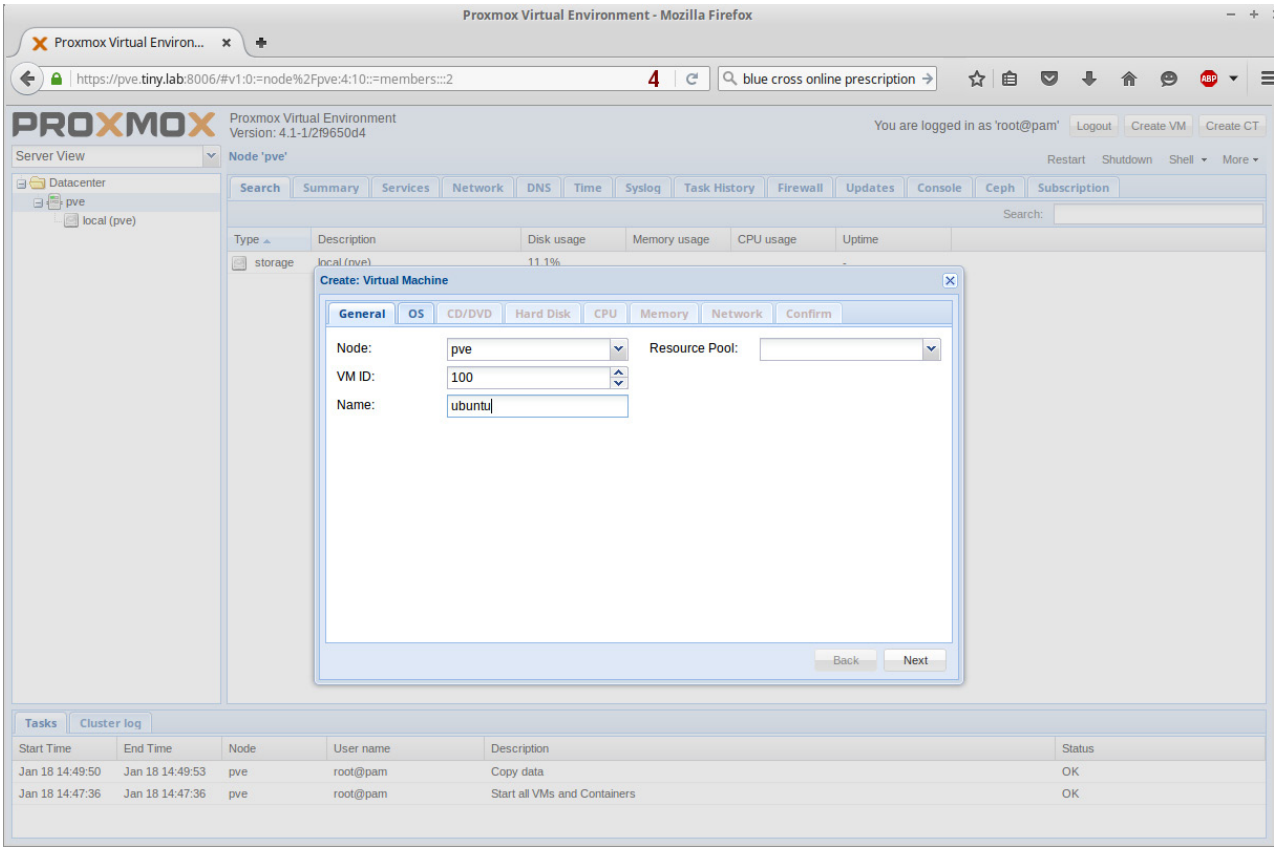


Figure 10. Creating a Proxmox VM, Step 1

make sure the "Use CD/DVD disc image file (iso)" radio button is chosen, select the storage drive ("local"), and use the "ISO Image" drop-down to choose the .iso you uploaded. Click Next.

The hard disk settings are fine as is. Note that you'll be creating a 32GB drive on the "local" drive. That will be plenty for all the machines you create, except the mirror server, which will be more like 200GB. Click Next.

Depending on how many CPUs and cores your machine has, you'll be able to add more than one "Socket" and more than one "Cores". Your Ubuntu servers will run fine with a single CPU, so leave the defaults (1 socket, 1 core, Default kvm64 Type), and click Next.

With memory, like CPU, your settings can vary based on how much memory your system has. I assume you don't have much, and I've tested various configurations and found that setting a range works best. Click the radio button next to "Automatically allocate memory within this range", and set the Maximum value to 1024 and the

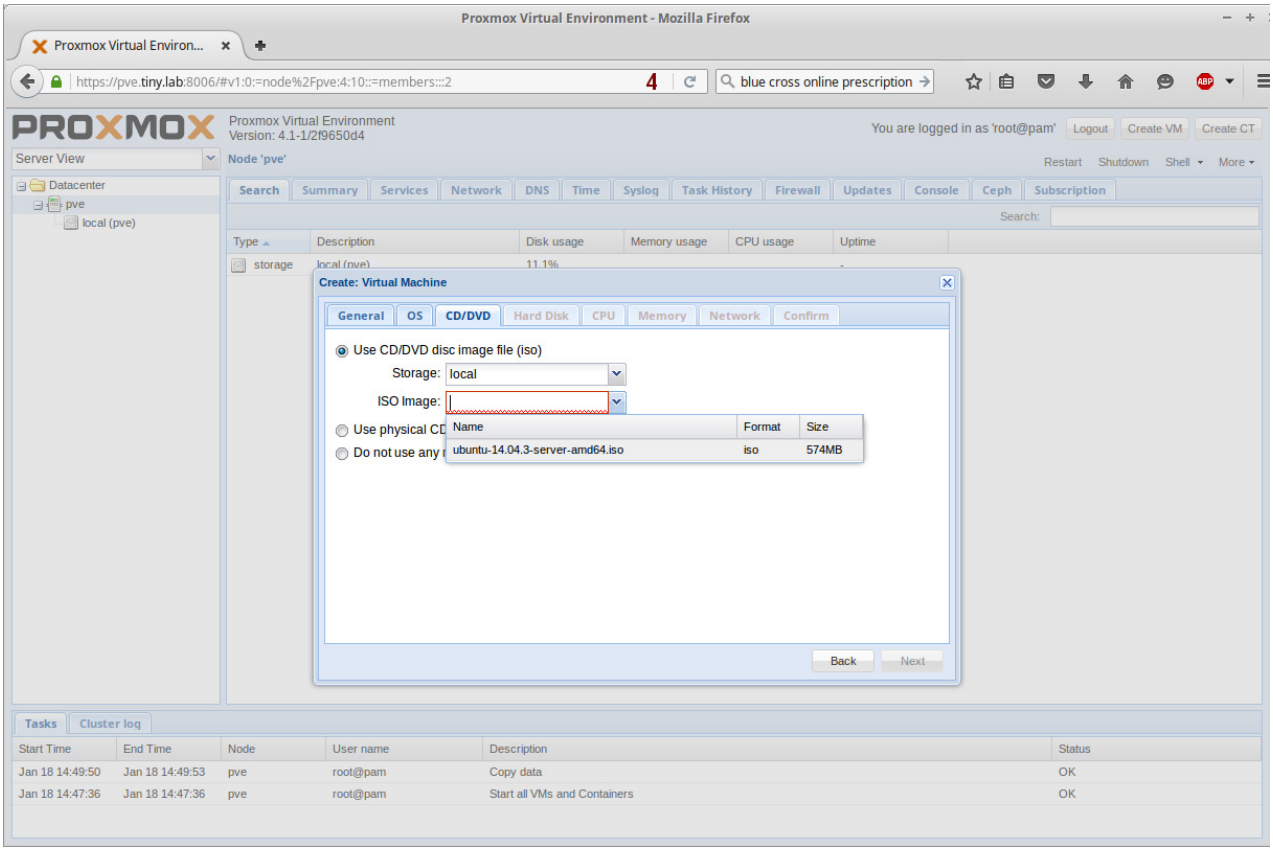


Figure 11. Creating a Proxmox VM, Step 2

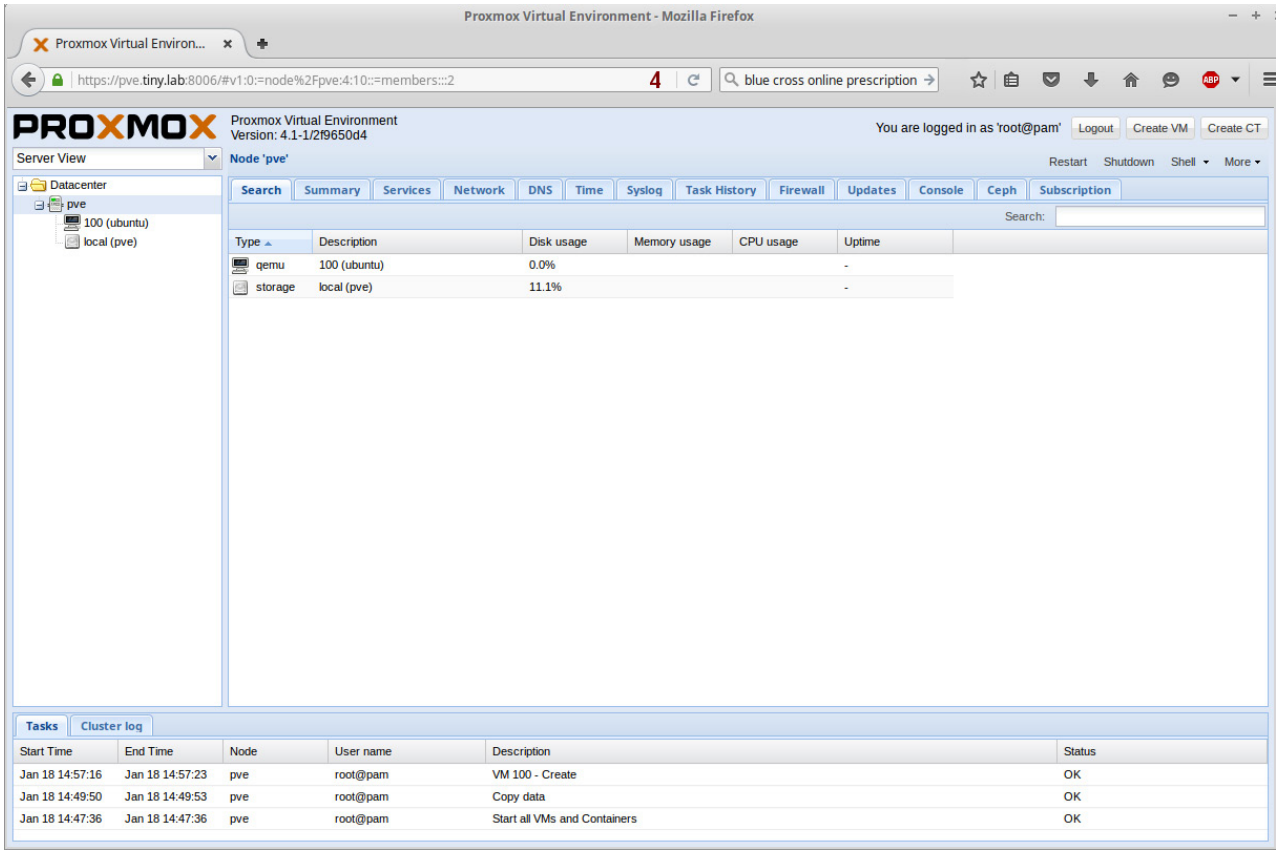


Figure 12. Creating a Proxmox VM, Step 3

Minimum value to 512. The virtual machine will use only the memory it needs, which typically will be much less than 512MB, but it automatically can use as much as 1GB if necessary. Click Next.

For networking, select Bridged mode and Bridge vmbr0. Click the box next to Firewall and leave all the other settings as defaults. Click Next to review a summary of your choices. When you click Confirm, Proxmox will create the virtual machine. When it's done, you'll see a new icon in the left-hand pane of the main view.

To start the new VM, either right-click on it and choose "Start" or left-click it once and choose Start from the menu located above the tabs in the right pane. As it starts up, you'll see the content of the Summary tab change. Right-click the machine and choose "Console" from this list (or from the menu above the tabs). A new browser window will open (check to make sure you're not blocking pop-ups on the site), and you'll see the Ubuntu start-up screen. You're ready to start deploying VMs!

Figure 13. You're ready to start!

Stay tuned for the third and final installment in an upcoming issue soon!■

John Tonello is the Director of IT for NYSERNet Inc., New York state's regional optical networking company. He's been a Linux user and enthusiast since building his first Slackware system from diskette 20 years ago. Since then, he's developed web and IT solutions for major universities, Fortune 500 companies and small start-ups. A former Cornell University IT trainer and writer, John served six years as the mayor of an Upstate New York city, where he championed the use of technology to help solve problems facing municipalities.

Resources

Create a bootable Linux (Ubuntu) USB:
https://help.ubuntu.com/community/Installation/FromUSBStick.

Check a Linux system for virtualization capabilities:
http://virt-tools.org/learning/check-hardware-virt.

Check a Windows system for virtualization capabilities, for Intel-based systems:
http://intel.ly/217A6MK.

SecurAble is a tool from Gibson Research Corp. that helps you learn more about your processor:
https://www.grc.com/securable.htm.

Check a Macintosh system for virtualization capabilities—Apple provides some tools at
https://support.apple.com/en-us/HT203296.

You can download the bootable Proxmox .iso files from
http://proxmox.com/en/downloads/category/iso-images-pve.

Setting metrics on Ubuntu Linux interfaces: http://bit.ly/1mRibHa.

Information on router metrics from WikiPedia:
https://en.wikipedia.org/wiki/Metrics_%28networking%29.

Download the Ubuntu server 14.04.3-server-amd64.iso from
http://www.ubuntu.com/download/server.

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

The Forrester Wave™: Digital Experience Platforms, Q4 2015

The demand to be at every touchpoint in the customer lifecycle is no longer an option—it's a requirement. To manage and deliver experiences consistently across all touchpoints, organizations are looking to digital experience platforms as the foundation of their digital presence.

Get Forrester's evaluation of the best vendors, including:

- The ten providers that matter most.
- How each vendor stacks up to Forrester's criteria.
- Six needs a digital experience platform architecture must meet.

> **http://geekguide.linuxjournal.com/content/forrester-wave-digital-experience-platforms-q4-2015**

ACQUIA™ The Ultimate Guide to Drupal 8 by Acquia

With 200+ new features and improvements, Drupal 8 is the most advanced version of Drupal yet. Drupal 8 simplifies the development process, enabling you to do more, in less time, with proven technologies that make it easier to be a first time Drupal user. Read this eBook, written by Angie Byron (you may know her as "webchick"), to get up to speed on the new changes in Drupal 8. Drupal 8's improvements include:

- API-driven content approach.
- Rest-first native web services.
- Seamless integration with existing technologies.
- Multilingual features and capabilities.
- Responsive by nature and mobile-first.

> **http://geekguide.linuxjournal.com/content/ultimate-guide-drupal-8**

ACQUIA™ How to Choose a Great CMS by Acquia

Web Content Management Systems serve as the foundation of your digital experience strategy. Yet many organizations struggle with legacy proprietary products that can't keep pace with the new realities of digital marketing. To determine if you are in need of a new CMS, use our guide, which includes:

- An evaluation to see if your current CMS supports your digital business strategy.
- The top considerations when selecting a new CMS.
- A requirements checklist for your next CMS.
- Ten questions to ask CMS vendors.

> **http://geekguide.linuxjournal.com/content/how-choose-great-cms**

Fast/Flexible Linux OS Recovery

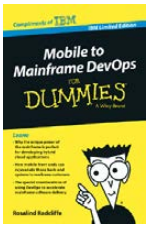
How long does it take to restore a system, whether virtual or physical, back to the exact state it was prior to a failure? Re-installing the operating system, re-applying patches, re-updating security settings takes too damn long! If this is your DR Strategy, we hope you've documented every change that's been made, on every system?!

Most companies incorporate backup procedures for critical data, which can be restored quickly if a loss occurs. However, that works only if you have an OS to restore onto and the OS supports the backup.

In this live one-hour webinar, learn how to enhance your existing backup strategies for complete disaster recovery preparedness using Storix System Backup Administrator (SBAdmin), a highly flexible full-system recovery solution for UNIX and Linux systems.

Webinar: April 26, 2016 at 1:00 PM Eastern

> **http://www.linuxjournal.com/storix-recovery**



Mobile to Mainframe DevOps for Dummies

In today's era of digital disruption empowered by cloud, mobile, and analytics, it's imperative for enterprise organizations to drive faster innovation while ensuring the stability of core business systems. While innovative systems of engagement demand speed, agility and experimentation, existing systems of record require similar attributes with additional and uncompromising requirements for governance and predictability. In this new book by Rosalind Radcliffe, IBM Distinguished Engineer, you will learn about:

- Responding to the challenges of variable speed IT.
- Why the mainframe is a unique and ideal platform for developing hybrid cloud applications.
- How mobile front ends can rejuvenate back-end systems to reach new customers.
- And, special considerations for using a DevOps approach to accelerate mainframe software delivery.

> **http://devops.linuxjournal.com/devops/mobile-mainframe-devops-dummies**

BRAND-NEW EDITION!

DevOps For Dummies – New Edition with SAFe®

In this NEW 2nd edition, learn why DevOps is essential for any business aspiring to be lean, agile, and capable of responding rapidly to changing customers and marketplace.

Download the E-book to learn about:

- The business need and value of DevOps.
- DevOps capabilities and adoption paths.
- How cloud accelerates DevOps.
- The Ten DevOps myths.
- And more.

> **http://devops.linuxjournal.com/devops/devops-dummies-new-edition-safe**

Doing for User Space What We Did for Kernel Space

A way to give individuals root for themselves in a world of administrative silos.

DOC SEARLS

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.

PREVIOUS
Feature: The Tiny Internet Project, Part II

I believe the best and worst thing about Linux is its hard distinction between *kernel space* and *user space*.

Without that distinction, Linux never would have become the most leveraged operating system in the world. Today, Linux has the largest range of uses for the largest number of users—most of whom have no idea they are using Linux when they search for something on Google or poke at their Android phones. Even Apple stuff wouldn't be what it is (for

example, using BSD in its computers) were it not for Linux's success.

Not caring about user space is a feature of Linux kernel development, not a bug. As Linus put it on our 2003 Geek Cruise (http://www.linuxjournal.com/article/6427), "I only do kernel stuff...I don't know what happens outside the kernel, and I don't much care. What happens inside the kernel I care about." After Andrew Morton gave me additional schooling on the topic a couple years later on another Geek Cruise (http://www.linuxjournal.com/article/8664), I wrote:

> Kernel space is where the Linux species lives. User space is where Linux gets put to use, along with a lot of other natural building materials. The division between kernel space and user space is similar to the division between natural materials and stuff humans make out of those materials.

A natural outcome of this distinction, however, is for Linux folks to stay relatively small as a community while the world outside depends more on Linux every second. So, in hope that we can enlarge our number a bit, I want to point us toward two new things. One is already hot, and the other could be.

The first is blockchain (https://en.wikipedia.org/wiki/Block_chain_%28database%29), made famous as the distributed ledger

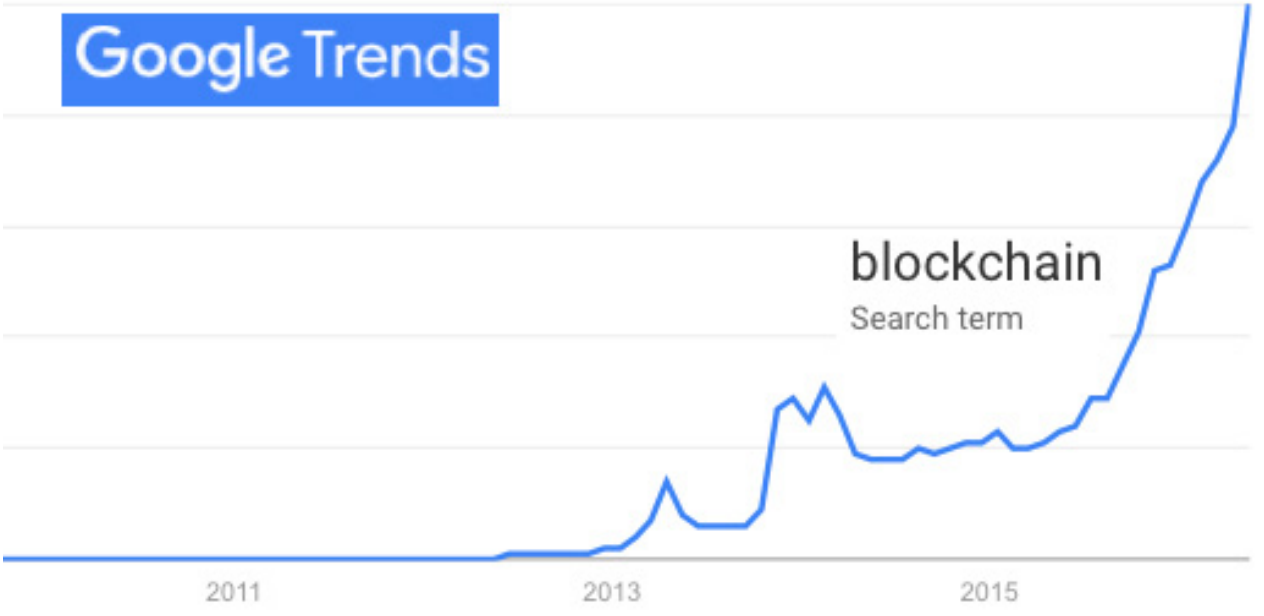


Figure 1. Google Trends for Blockchain

used by Bitcoin, but useful for countless other purposes as well. At the time of this writing, interest in blockchain is trending toward the vertical (https://www.google.com/trends/explore#q=blockchain).

The second is **self-sovereign identity**. To explain that, let me ask who and what you are.

If your answers come from your employer, your doctor, the Department of Motor Vehicles, Facebook, Twitter or Google, they are each **administrative identifiers**: entries in namespaces each of those organizations control, entirely for their own convenience. As Timothy Ruff of Evernym (http://evernym.com) explains, "You don't exist for them. Only your identifier does." It's the dependent variable. The independent variable—the one controlling the identifier—is the organization.

If your answer comes from your self, we have a wide-open area for a new development category—one where, finally, we can be set fully free in the connected world.

The first person to explain this, as far as I know, was Devon Loffreto (https://twitter.com/nzn). He wrote "What is 'Sovereign Source Authority'?" in February 2012, on his blog, The Moxy Tongue (http://www.moxytongue.com/2012/02/what-is-sovereign-source-authority.html). In "Self-Sovereign Identity" (http://www.moxytongue.com/2016/02/self-sovereign-identity.html), published in February 2016, he writes:

> Self-Sovereign Identity must emit directly from an individual human life, and not from within an administrative mechanism... self-Sovereign Identity references every individual human identity as the origin of source authority. A self-Sovereign identity produces an administrative trail of data relations that begin and resolve to individual humans. Every individual human may possess a self-Sovereign identity, and no person or abstraction of any type created may alter this innate human Right. A self-Sovereign identity is the root of all participation as a valued social being within human societies of any type.

To put this in Linux terms, *only the individual has root for his or her own source identity*. In the physical world, this is a casual thing. For

example, my own portfolio of identifiers includes:

- David Allen Searls, which my parents named me.
- David Searls, the name I tend to use when I suspect official records are involved.
- Dave, which is what most of my relatives and old friends call me.
- Doc, which is what most people call me.

As the sovereign source authority over the use of those, I can jump from one to another in different contexts and get along pretty well. But, that's in the physical world. In the virtual one, it gets much more complicated. In addition to all the above, I am @dsearls (my Twitter handle) and dsearls (my handle in many other net-based services). I am also burdened by having my ability to relate contained within hundreds of different silos, each with their own logins and passwords.

You can get a sense of how bad this is by checking the list of logins and passwords on your browser. On Firefox alone, I have hundreds of them. Many are defunct (since my collection dates back to Netscape days), but I would guess that I still have working logins to hundreds of companies I need to deal with from time to time. For all of them, I'm the dependent variable. It's not the other way around. Even the term "user" testifies to the subordinate dependency that has become a primary fact of life in the connected world.

Today, the only easy way to bridge namespaces is via the compromised convenience of "Log in with Facebook" or "Log in with Twitter". In both of those cases, each of us is even less ourselves or in any kind of personal control over how we are known (if we wish to be knowable at all) to other entities in the connected world.

What we have needed from the start are personal systems for instantiating our sovereign selves and choosing how to reveal and protect ourselves when dealing with others in the connected world. For lack of that ability, we are deep in a metastasized mess that Shoshana Zuboff calls "surveillance

capitalism" (http://www.faz.net/aktuell/feuilleton/debatten/the-digital-debate/shoshana-zuboff-secrets-of-surveillance-capitalism-14103616.html?printPagedArticle=true#pageIndex_2), which she says is:

> ...unimaginable outside the inscrutable high velocity circuits of Google's digital universe, whose signature feature is the Internet and its successors. While the world is riveted by the showdown between Apple and the FBI, the real truth is that the surveillance capabilities being developed by surveillance capitalists are the envy of every state security agency.

Then she asks, "How can we protect ourselves from its invasive power?"

I suggest self-sovereign identity. I believe it is only there that we have both safety from unwelcome surveillance and an Archimedean place to stand in the world. From that place, we can assert full agency in our dealings with others in society, politics and business.

I came to this provisional conclusion during ID2020 (http://www.id2020.org), a gathering at the UN on May. It was gratifying to see Devon Loffreto there, since he's the guy who got the sovereign ball rolling in 2013. Here's what I wrote about it at the time (http://blogs.harvard.edu/doc/2013/10/14/iiw-challenge-1-sovereign-identity-in-the-great-silo-forest), with pointers to Devon's earlier posts (such as one sourced above).

Here are three for the field's canon:

- "Self-Sovereign Identity" by Devon Loffreto: http://www.moxytongue.com/2016/02/self-sovereign-identity.html.
- "System or Human First" by Devon Loffreto: http://www.moxytongue.com/2016/05/system-or-human.html.
- "The Path to Self-Sovereign Identity" by Christopher Allen: http://www.lifewithalacrity.com/2016/04/the-path-to-self-soverereign-identity.html.

A one-pager from Evernym (http://evernym.com), digi.me (https://get.digi.me), iRespond (http://irespond.com) and Respect Network (https://www.respectnetwork.com) also was circulated there, contrasting administrative identity (which it calls the "current model") with the

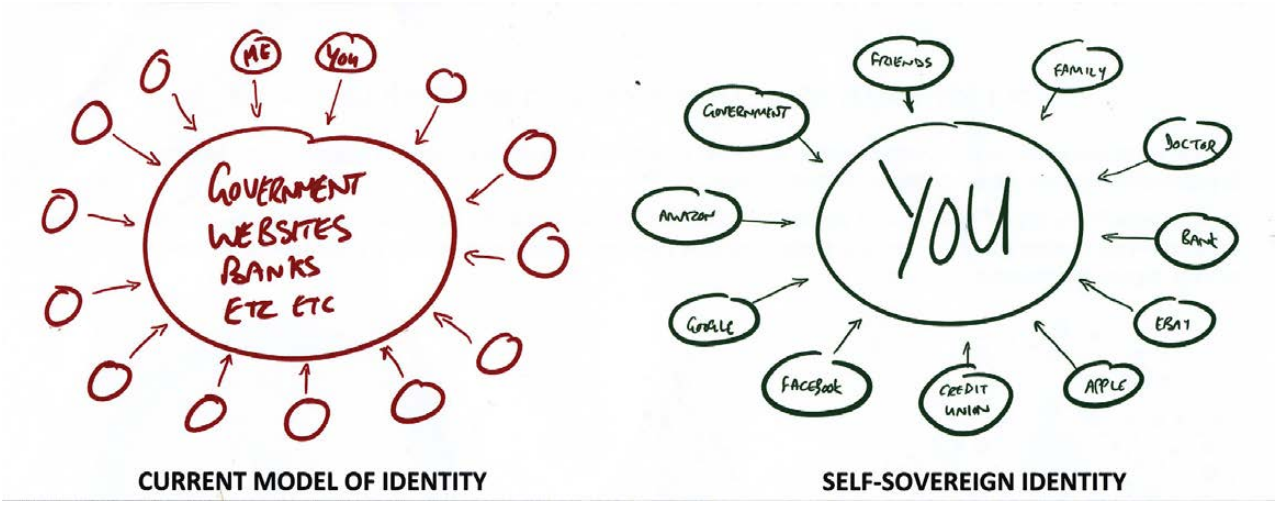


Figure 2. Current Model of Identity vs. Self-Sovereign Identity

self-sovereign one. In it is the graphic shown in Figure 2.

The platform for this is Sovrin, explained as a "Fully open-source, attribute-based, sovereign identity graph platform on an advanced, dedicated, permissioned, distributed ledger" (http://evernym.com/technology). There's a white paper too: http://evernym.com/assets/doc/Identity-System-Essentials.pdf?v=167284fd65. The code is called plenum (https://github.com/evernym/plenum), and it's at GitHub.

Here—and places like it—we can do for user space what we've done for the last quarter century for kernel space.■

Send comments or feedback via http://www.linuxjournal.com/contact or to ljeditor@linuxjournal.com.

RETURN TO CONTENTS

ADVERTISER INDEX

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

ADVERTISER	URL	PAGE #
AnDevCon	http://www.AnDevCon.com	7
Drupalize.me	http://drupalize.me	87
InterDrone	http://www.InterDrone.com	35
LinuxCon North America	http://go.linuxfoundation.org/lcna16-linuxjournal	23
Peer 1 Hosting	http://go.peer1.com/linux	79
Texas Linux Fest	http://2016.texaslinuxfest.org/	41