LINUX JOURNAL™
Since 1994: The Original Magazine of the Linux Community
JANUARY 2016 | ISSUE 261 | www.linuxjournal.com

AUTOMATE Full Disk Encryption

IMPROVE File Transfer Security

+

Enhance Client-Side Performance for Users

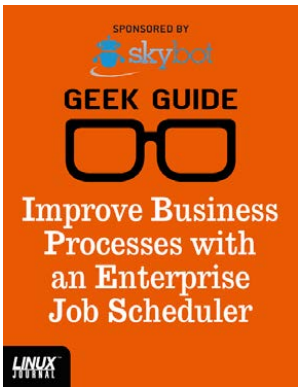
Making Sense of Profiles and RC Scripts

ABINIT for Computational Chemistry Research

Leveraging Ad Blocking

Audit Serial Console Access

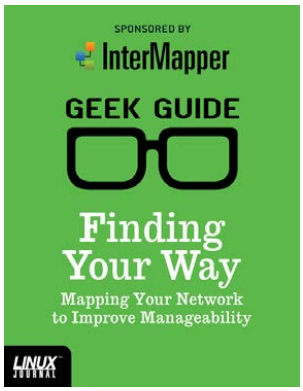
WATCH: ISSUE OVERVIEW



Improve Business Processes with an Enterprise Job Scheduler

Author:
Mike Diehl

Sponsor:
Skybot



Finding Your Way: Mapping Your Network to Improve Manageability

Author:
Bill Childers

Sponsor:
InterMapper

DIY Commerce Site

Author:
Reuven M. Lerner

Sponsor: GeoTrust

Combating Infrastructure Sprawl

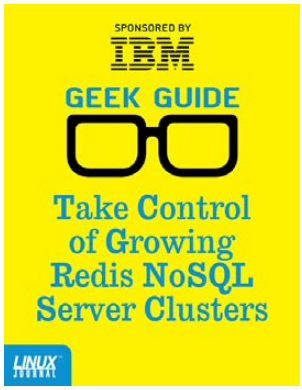
Author:
Bill Childers

Sponsor:
Puppet Labs

Get in the Fast Lane with NVMe

Author:
Mike Diehl

Sponsor:
Silicon Mechanics & Intel



Take Control of Growing Redis NoSQL Server Clusters

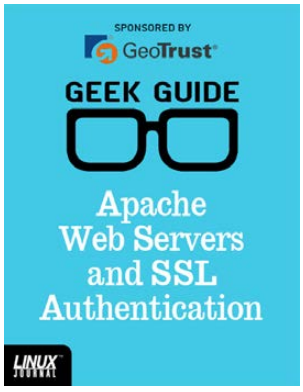
Author:
Reuven M. Lerner

Sponsor: IBM

Linux in the Time of Malware

Author:
Federico Kereki

Sponsor:
Bit9 + Carbon Black



Apache Web Servers and SSL Encryption

Author:
Reuven M. Lerner

Sponsor: GeoTrust

CONTENTS

JANUARY 2016
ISSUE 261

FEATURES

50 Secure File Transfer

Use RFC 1867, thttpd and Stunnel to improve security.

Charles Fisher

72 Transferring Conserver Logs to Elasticsearch

Auditing serial console access in real time.

Fabien Wernli

COLUMNS

26 Reuven M. Lerner's At the Forge
Client-Side Performance

32 Dave Taylor's Work the Shell
Planetary Age

36 Kyle Rankin's Hack and /
Full Disk Encryption

40 Shawn Powers' The Open-Source Classroom
Profiles and RC Files

86 Doc Searls' EOF
What We Can Do with Ad Blocking's Leverage

IN EVERY ISSUE

8 Current_Issue.tar.gz

10 Letters

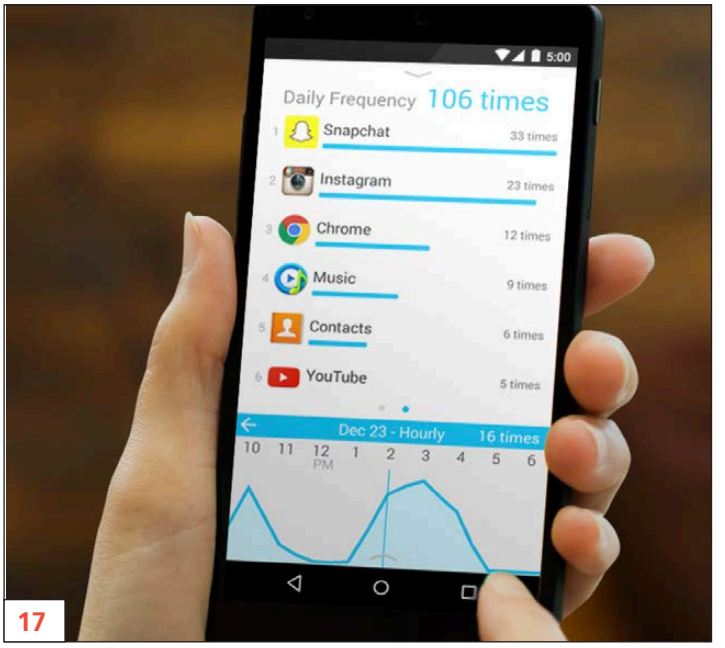
14 UPFRONT

24 Editors' Choice

46 New Products

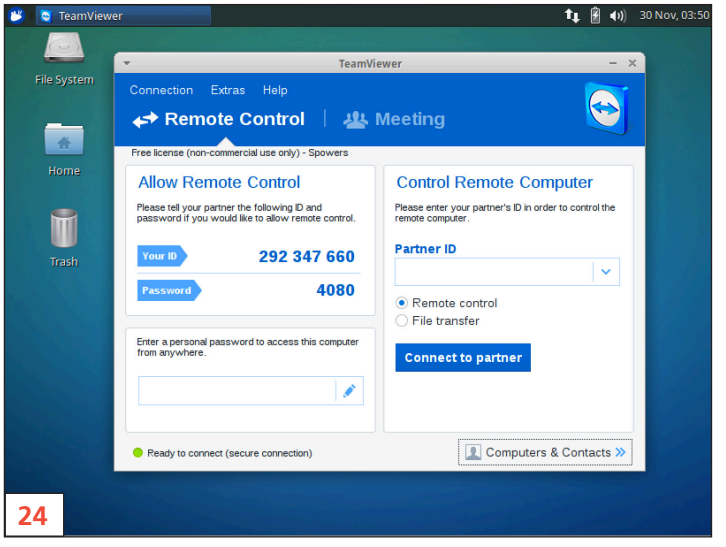
ON THE COVER

- **Improve File Transfer Security, p. 50**
- **Audit Serial Console Access, p. 72**
- **Automate Full Disk Encryption, p. 36**
- **Enhance Client-Side Performance for Users, p. 26**
- **Making Sense of Profiles and RC Scripts, p. 40**
- **ABINIT for Computational Chemistry Research, p. 20**
- **Leveraging Ad Blocking, p. 86**



17

18



24

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

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

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

SHAWN POWERS

2016: a Long Year

I know you're expecting a sarcastic comment about an election year in the US making it seem longer than normal, but no, 2016 is literally a longer year than most. (Although that bit about it seeming even longer has some merit.) What better way to start this bonus-sized year than with an issue of *Linux Journal*? I'm not a fan of resolutions, but I do have a challenge for you: learn something new this year. Personally, I plan to learn more about development. I dabbled in 2015, and it's given me the urge to learn more. Reuven M. Lerner is the perfect author to join on a journey like that, and this month, he teaches how to help improve client-side performance on your Web applications. Sure, we could buy everyone faster computers, but Reuven shows that there are better (and cheaper) ways to accomplish client-side improvements.

Dave Taylor does some really cool calculations this issue and explains how to determine your age on other planets programmatically. There's more to it than that, but whether you plan to stay on Earth or migrate to Mars, learning to calculate with the `date` command will be a useful skill no matter where you live. Speaking of time, Kyle Rankin gives a lesson in how he spent many hours saving a few minutes. More specifically, he teaches how to use the Debian preseed procedure to automate disk encryption and partition creation. It sounds like something that wouldn't be too complicated to automate, but Kyle found it was a messy rabbit hole. His column should at least provide a flashlight if you decide to delve into a similar hole.

I took a note from my own challenge this month and learned the exact way Linux systems deal with profile and RC files. It seems like a trivial thing to learn about, but it turns out that the procedures for loading profiles and such are fairly complicated. I was tired of just copy/pasting information into files without knowing exactly why some information goes into profiles and

VIDEO:
Shawn Powers runs through the latest issue.

some into RC files, so I decided to get to the bottom of how those preference files are loaded. This month, I share the fruit of my labor and hope to demystify the shell-based config files for everyone reading.

Encrypting filesystems and salting hashes are common ways to protect data on a server. Quite honestly, we're beginning to see the value in encrypting local data, and it's becoming common for servers to be secured more than ever before. Unfortunately, most security breaches aren't happening on the local machines; rather, they're happening over the network. It doesn't matter how secure your local filesystem might be, if you're not transmitting and receiving data in a secure way, no amount of local encryption will protect your data. Charles Fisher not only exposes the weaknesses with traditional file transfer methods, but he also explains how to shore up network transfers when sending and receiving data. Whether you consider your data sensitive or not, there's no reason to adopt insecure methods in your environment. Charles shows how to make sure you keep your private data private, even when you send it across the Internet.

Fabien Wernli also discusses security this month, but rather than securing network transfers, he covers how to manage log files for console connections. Keeping track of serial connections to the server console can be challenging when your server number increases, but thanks to syslog-ng, you're able to log that information. Fabien then goes on to describe the process for consolidating log files into searchable archives and even shows how to integrate console logs into a real-time monitoring solution. If you manage a large number of servers via console or serial (even over the LAN), you'll want to read his article.

Doc Searls finishes the issue by discussing the ramifications of ad blocking on the modern Internet. If you browse the Web, chances are pretty good that you use an ad blocker to make your experience more pleasant. Blocking ads means blocking revenue for content creators, and rather than pretending it's not an issue, we need to figure out how to respond in a way that is useful both to consumers and content creators. As usual, Doc has incredible insight, and you'll want to check it out.

This first issue of *Linux Journal* in 2016 may be brand new, but it still has all the tech tips, product reviews and helpful information you've come to expect month after month. Whether the new year means ice and snow or sunshine and roses in your part of the world, we hope this issue helps start it off on a good note. We'll see you again next month, when February grows an extra day and is almost a full-size month!■

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

letters

Server Hardening—ipset:set

Regarding Greg Bledsoe's "Server Hardening" article in the November 2015 issue: I created a modified script for generating ipset blocklists. Namely it creates a set of ipsets, one a hash:net and the other a hash:ip. The script generates a second script called blset.sh, which adds the IP addresses to the ipset hashes. The blset.sh script first adds all the hash:net entries from the various sources, then the hash:ip set is created, but entries are not added if they already exist in the hash:net set.

The new script does not exceed the ipset size limit. The suggested script in Greg's *Linux Journal* article puts all IPs into a hash:ip, which quickly becomes too large. The script is at https://github.com/zuikway/tlj_blocklist.

—Wayne Shumaker

Server Hardening, II

Greg Bledsoe missed one small thing that can increase a server's security: reduce the amount of network traffic a server must process:

```
iptables -t mangle -I PREROUTING -m state --state INVALID -j DROP
```

`INVALID` packets are those that must belong to an established connection, yet netfilter has no connection recorded for it. They are "spurious" packets that cannot be delivered, so they should be dropped as early as possible. It isn't worth spending one extra CPU cycle on these packets. Although it won't eliminate the ill effects of a DDoS attack, it can significantly reduce the time the CPU spends handling `INVALID` packets.

—Neal

Find Words

Dave Taylor's Work the Shell column in the September–November 2015 issues covers a fun toy program near and dear to my heart. I've been using

a very similar de-jumbling algorithm to strengthen my scripting in Perl and Python—although I must admit I haven't been ambitious enough to implement it in bash! It was cool to see Dave use the nearly the same approach I came up with myself. I figured it might be interesting to share my own variation to the same problem.

Considering that modern machines are overkill for most scripts, I started off simply alphabetizing the entire dictionary (first in /usr/share/dict/words, and later a set of professional *Scrabble* word lists I found on-line). In my language du jour, I construct a massive hash keyed on the alphabetized words, with an array of matching original words as the value. For example:

```
$list{'abt'} -> ['bat', 'tab']
```

All in all, this approach takes only a few seconds on a five-year-old laptop, and 21MB of RAM for the data structure.

The next fun part was digging into my computer science background and using a recursive algorithm to

deconstruct the input letter sets by calling the same function minus a different letter each time and looking up the result in the hash. Putting the input function into a loop (checking for EOF or "q" for termination) allows you to perform multiple searches against the hash you've spent several busy CPU-seconds constructing.

Keep on hacking!
—Chandler Wilkerson

Dave Taylor replies: *Great to hear from you, Chandler, and glad my column brought you some enjoyment as you realized we'd taken the same algorithmic approach to the word jumble algorithm!*

AWS EC2 VPC CLI

Thanks for an excellent journal. I really enjoy it and love the digital version on my Kindle.

The reason I'm writing is just a general hint to Kyle Rankin's great article on the EC2 CLI in the October 2015 issue. I have myself gone through an identical process for exactly the same reasons in changing to the Python CLI. The only thing I chose to do differently in the end was processing the output. I, on occasion, had issues in processing the text output of the Java CLI in that it sometimes changed slightly between versions, forcing me to stick to a certain version or adapt my awk|perl|grep processing of the text output. Text output for the Python CLI was bigger and a bit trickier to parse well—enter JSON output. As Kyle writes, the Python CLI offers the option of different outputs, including JSON. It's a slightly steeper learning curve, but using the JSON output together with the jq JSON command-line parser makes processing anything from the CLI straightforward and keeps me safe from EC2 CLI adding fields or new lines, etc., that may break by text processing! One can always script things prettier, but being a one-liner fan, one can, for example, get all the volume IDs for one's servers:

```
aws ec2 describe-instances | jq -r
➥'.Reservations[].Instances[].BlockDeviceMappings[].Ebs.VolumeId'
```

Taking it a little further, snapshot every EBS volume, but only if it does not belong to a certain tag (or do it the other way around and snapshot only a given tag) and snapshot only those that are mounted on a given device name:

```
aws ec2 describe-instances | jq -r '.Reservations[].Instances[] |
➥select(contains({Tags: [{Key: "SomeKey",Value:
➥"SomeValue"} ]}) | not) | .BlockDeviceMappings[] |
➥select(contains({DeviceName: "/dev/sda"})) | .Ebs.VolumeId'
➥| parallel aws ec2 create-snapshot
➥--description "backup_`date +\%Y\%m\%d`" --volume-id
```

`parallel` is a great trick to call the command on every volume ID. I would often use xargs and give multiple IDs in one call, but with the Python CLI, I could give each call only one volume ID. I add the date to the description for a better overview of snapshots and a simple way to monitor and delete given snapshots.

Then, I would also have a similar simple one-liner to clean up old snapshots and monitor that all snapshots are successful.

Keep up the good work!

—Elfar

Photo of the Month

Mateo from Argentina, already supporting Linux the first day of his life.

—Gaston

PHOTO OF THE MONTH

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

WRITE *LJ* A LETTER

We love hearing from our readers. Please send us your comments and feedback via http://www.linuxjournal.com/contact.

LINUX JOURNAL

At Your Service

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

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

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

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

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

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

diff -u
What's New in Kernel Development

There's an ongoing impulse among a diversity of developers to be able to compile some or all of the Linux kernel as a library, so that a piece of software could use kernel services and APIs while running under a different kernel entirely, or a different operating system.

This time, the impulse came from **Octavian Purdila**, creator of the **Linux Kernel Library** (LKL), essentially an entire kernel compiled as a static library. He distinguished LKL from projects like **User Mode Linux** (UML), saying that LKL was more lightweight, having no infrastructure requirements or needing any particular sort of runtime environment.

A bunch of folks expressed interest, especially in terms of interacting with similar projects like **libOS** and **libguestFS**. And, **Richard Weinberger** remarked that LKL seemed to solve UML's biggest pain points: the need to use **ptrace()** to handle system calls and to do virtual memory management using **SIGSEGV**.

In a device-centric world with heavy, inefficient battery technology, there's a big incentive to figure out ways to save power. One possibility is to turn off portions of hardware when they're currently not in use, like a phone's touchscreen when the phone is in your pocket.

The difficulty lies in knowing exactly which piece of hardware to turn off, and when. If there's a clear user action, like flipping closed a flip-phone, the problem is simplified. **Irina Tirdea** recently tried to recognize such actions and come up with mechanisms to respond to them properly. She posted some patches to do this.

Octavian Purdila, also working on the project with Irina, described a target scenario as being when a touchscreen has been blanked but is still aware of the user's touch—through the fabric of a pocket, for example. The goal of the patches, he said, would be to save power by turning off all the hardware associated with that screen, and turn everything on again when the user activates the device.

The problem with this sort of feature

is that it could be implemented along any of a number of different layers of the kernel code. The ideal location could make the difference between a complex, easily broken implementation and a simple, efficient implementation. Several folks felt that Irina and Octavian's approach was in the wrong part of the kernel, and the discussion devolved into a consideration of completely different approaches.

No consensus arose, although the allure of power-savings will undoubtedly keep the debate alive.

Mounting a filesystem under a virtual machine can be tricky. Security privileges and restrictions need to be respected, or else a filesystem could become a vector of attack by a malicious user. This particular area of kernel development also tends to have a wide appeal among companies trying to support their products, so it's possible for a variety of developers to find themselves working at cross purposes and need to accommodate each other before their patches can be accepted.

Seth Forshee and **Eric Biederman**, for example, recently wrote some patches to allow mounting **Ext4** and **FUSE** filesystems by unprivileged users, ignoring the security information that otherwise might prevent those users from accessing that data.

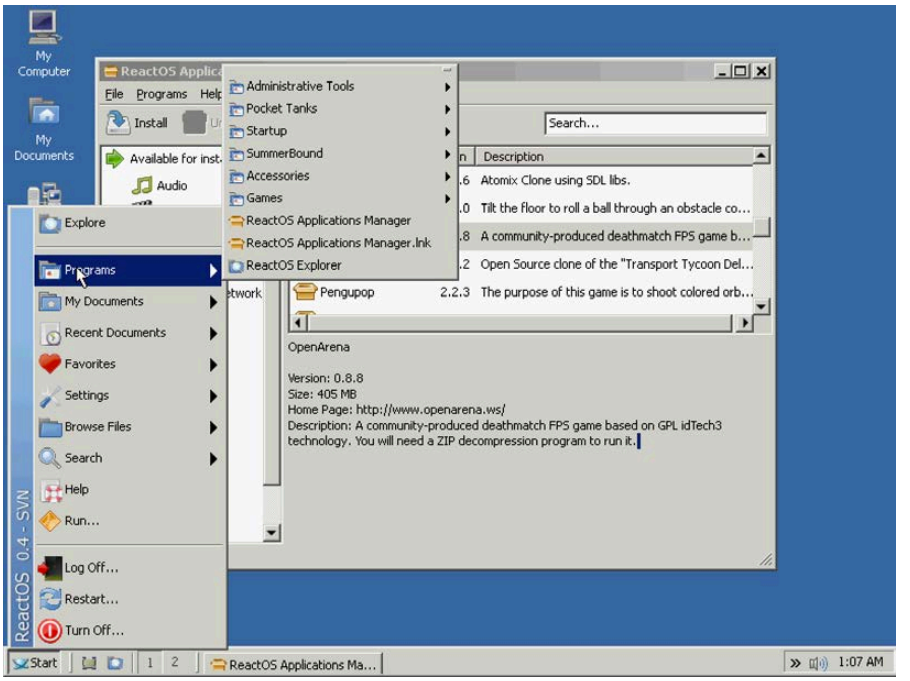
Meanwhile, **Lukasz Pawelczyk** was working on code specifically to support that same security information.

A debate sprang up over the particular context involved. **Andy Lutomirski** suggested that if a filesystem contained a user's own data, it would be fine to override security features, on the grounds that users should be able to do what they wanted with their own data. While **Casey Schaufler** replied that the kernel shouldn't care what the user knew about the data, it had to follow the security protocols or else it wouldn't be able to enforce them at all.

On the other hand, as Eric pointed out, filesystems like **FAT** and **Minix** weren't capable of storing the same type of security information as more modern filesystems. There had to be a way, he said, to mount such filesystems without requiring them to support security features they couldn't support.

It's an ongoing debate. Security trumps all other considerations, including dire need, so an issue like unprivileged filesystem mounts inevitably will involve a consideration of the specific context in which a user might try to do something. Often there's some kind of crazy nuance that makes something feasible when you could have sworn it never would be feasible.—**ZACK BROWN**

Non-Linux FOSS: Open-Source Windows?



I have mixed emotions about ReactOS. It's open source. It's freely available. But, its goal is to be binary-compatible with Windows! ReactOS is not a Linux operating system. In fact, it doesn't share the UNIX architecture at all. It looks like Windows NT, and it behaves much like Windows NT.

It's just odd!

The best way I can think to describe it is to imagine if Wine evolved into an entire operating system that booted on hardware instead of running inside Linux. That's basically what ReactOS feels like. It's not ready for prime time (and the developers make that very clear—it's alpha software), but it's worth checking out. Since it's early in the development process, if you get involved now, you can have a say in what compatibilities get priority.

ReactOS is the perfect solution for folks who need to run Windows apps, but absolutely refuse to run Microsoft code. I'm personally not convinced that ReactOS is a better idea than Wine running inside Linux, but I'm sure running it as its own operating system will provide possibilities that just can't happen in a Wine environment. The folks at ReactOS provide installers and prebuilt VM instances that can be launched in order to try it out on your existing system. Whether you are just morbidly curious about a non-Windows Windows or are interested in getting involved in the development, go to http://reactos.org for more details.

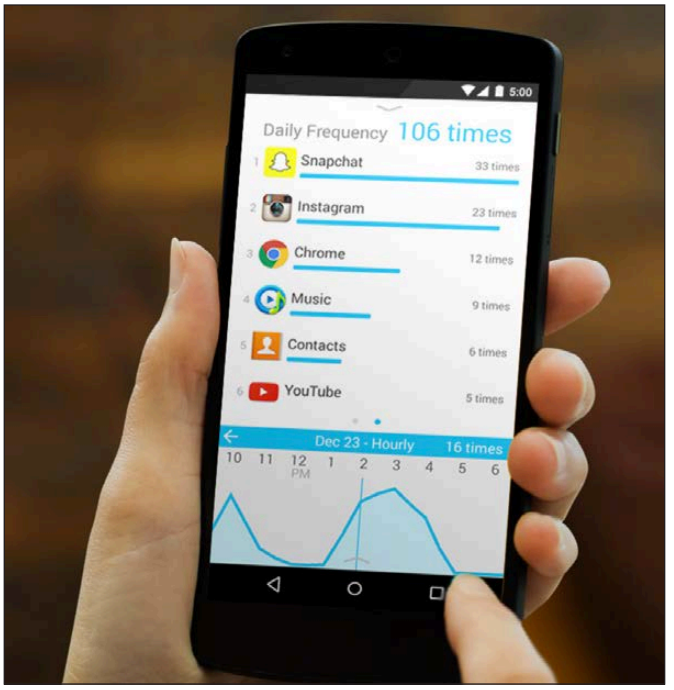
—SHAWN POWERS

Android Candy: Quality Time, or Not?

This is the season of resolutions, and in the technological world we live in, spending time off-line is a difficult but healthy activity. The problem is our lives have become so intertwined with our phones that it's easy to whip out our cell phones inadvertently to check our social networks quickly.

The QualityTime app is designed to help curb the habit just a bit. Ironically, it's an Android app designed to help you stop using Android apps so much. Still, it's just geeky enough to make limiting technology time a fun endeavor. If you like graphs, data, numbers and goals, QualityTime can help you identify where you spend most of your time on-line and then assist in lessening your face time with FaceTime (okay, not actually FaceTime, since that's an Apple app, but the word play was too fun to leave out).

If you're forgetting what your family members actually look like, or if you're surprised to see your friends as anything but their on-line avatars, you really need to give QualityTime a try. If you just want to see how much time you spend on various applications on your Android device, you should try QualityTime as well. I found the data alone worth the installation, and it inspired me to spend a little less time texting my kids and a little more time talking to them (while they text their friends—baby steps...).

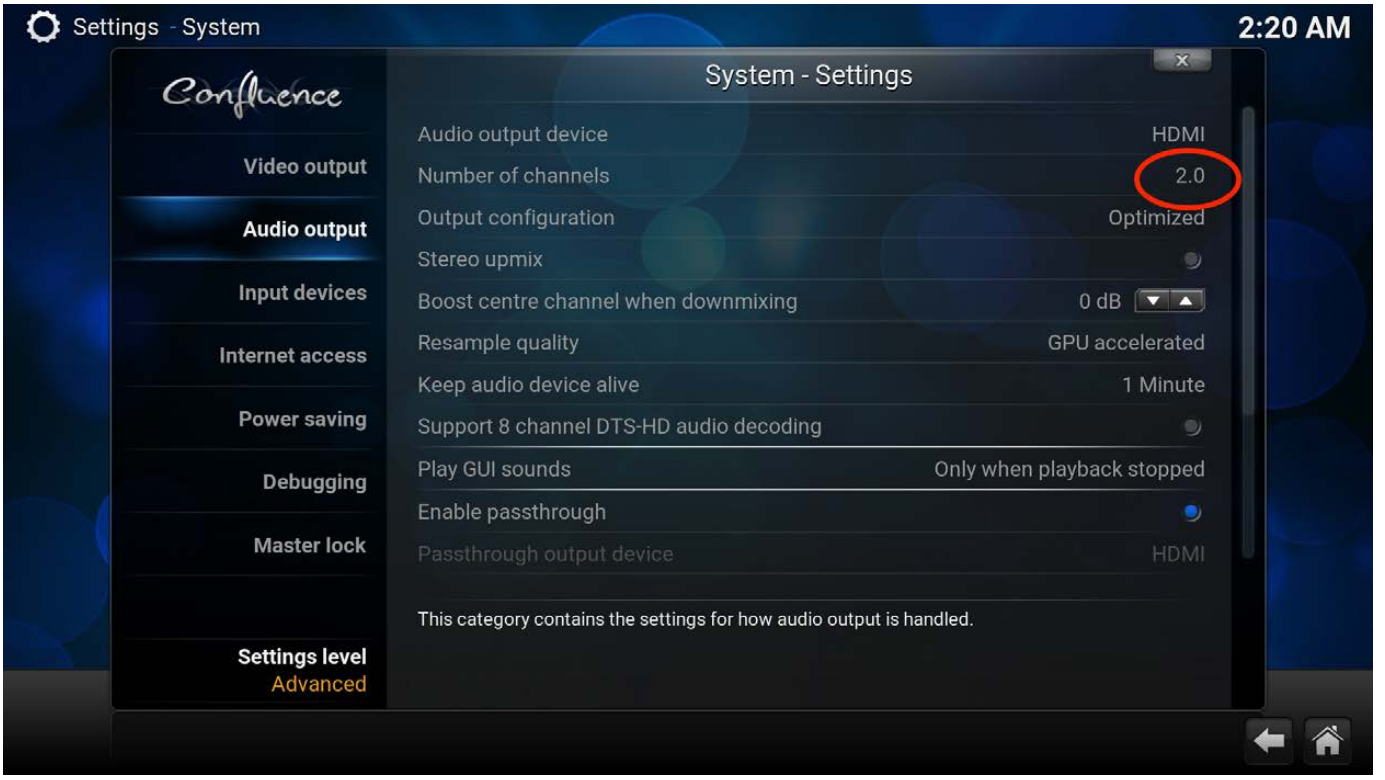


(Photo from http://qualitytimeapp.com)

Check it out at http://qualitytimeapp.com.

—SHAWN POWERS

Dear Kodi, Where's My Surround?!?!



I love Kodi. (This is just an evolution of my love for XBMC, since it's the same thing with a new name.) In fact, although I've expressed my love for Plex over and over (and over) the past few years, I still use Kodi as my main interface for the televisions in my house. We gave Plex a try as our main media center software when it was released for TiVo, but after several months, we found its interface to be cumbersome and the transcoding for local media frustrating.

So during the holidays, I once again installed Kodi on Raspberry Pi devices around my house. Using OpenELEC, the installation process itself is painless. Heck, even centralizing the library database was painless. The frustrating part was getting 5.1 surround sound to work.

On the bedroom televisions, surround sound is a moot point, because I just use whatever stereo speakers are included in the TV. For our main media center, however, I have a fancy Sonos PLAYBAR with subwoofer and rear channel speakers. The only audio connection the PLAYBAR accepts is optical audio, so I bought an inexpensive HDMI audio extractor. (This one works great: http://smile.amazon.com/dp/B00BIQER0E.)

The problem is that when Kodi is set to 5.1 audio output, the center channel is missing! There's a bit of disagreement as to whether it's a bug in Kodi/OpenELEC or just a result of optical audio supporting only two channels of audio. (If that seems odd to you, it was to me too. But apparently, it supports only two channels, which contain all the surround information, or something like that.) The non-intuitive solution is to force Kodi to 2.0 audio. Although it doesn't seem to make sense, I can vouch for it working. Kodi sends the audio as 2.0 stereo, which is transferred over optical (or HDMI, whatever you're using), and then the receiver decodes the surround information from that two-channel signal.

The tl;dr version is that Kodi will send the surround sound information over two-channel audio, so if you are missing your center channel, try switching to 2.0 audio. **—SHAWN POWERS**

They Said It

Don't watch the clock; do what it does. Keep going.
—Sam Levenson

What you do today can improve all your tomorrows.
—Ralph Marston

Life is 10% what happens to you and 90% how you react to it.
—Charles R. Swindoll

It does not matter how slowly you go as long as you do not stop.
—Confucius

Keep your eyes on the stars, and your feet on the ground.
—Theodore Roosevelt

ABINIT for Chemists

The single largest group of users on high-performance computing clusters has to be the chemists. Their CPU-year count is definitely at the very top of the list. Because of this heavy use, several different packages have become standard tools that most computational chemistry researchers use. So in this article, I take an introductory look at one called ABINIT (http://www.abinit.org).

ABINIT calculates the energy and structure of groups of nuclei and electrons. The method used to make these calculations is Density Functional Theory (DFT, https://en.wikipedia.org/wiki/Density_functional_theory). If you want to know more about the underlying theory, feel free to go talk to your nearest computational chemist.

Although my exposure has been with people running ABINIT on scores of machines in parallel, at least in a learning environment or for small systems, nothing is stopping you from running it on your own desktop. The first step, of course, is to install it on your machine. You may have packages within your distribution to make installation easier. For example, on Debian-based systems, you can install it with:

```
sudo apt-get install abinit abinit-data abinit-doc
```

The only issue with that method is you probably will get an older version of ABINIT. At the time of this writing, the Ubuntu package installs version 7.8.2, while on the Web site, you can download version 7.10.5.

If you need the latest available code, you always can get the source code from the main home page and compile it yourself on your local machine. In order to build it yourself, you need the usual utilities to build other packages, such as make, libtool and autoconf. Because the majority of the code is written in FORTRAN, you also need a compiler capable of compiling F90 code. This will allow you to build a basic version of ABINIT. You can include extra functionality, such as MPI or NetCDF, if you have them available on your system.

The main executable to run these calculations is called `abinit`. It takes a number of input files in order to do the actual calculation. One of these input files is actually

a file of files. It is a file that contains a list of other input files that abinit needs to read in. The usual filename ending is ".files". If you have this input file, you can run your simulation with:

```
abinit < my_input.files >& log
```

This tells abinit to read the input data from standard input (attached to the file my_input.files) and to write its results to standard output (attached to the file log). The log file only captures output that gets written out to the standard output stream. There is a lot more output that is written out. These other output files are defined in the my_input.files file. The following list is a more-detailed description of the contents:

- ab_in — main input file.
- ab_out — main output file.
- abi — root filename for other input files.
- abo — root filename for other output files.
- tmp — root filename for temporary files.
- my.psp — the pseudopotential used for this run.

The root names "abi", "abo" and "tmp" are used to create the multiple files for each of those sections.

There are a few rules around the input files that may cause problems if you don't follow them. The first is that you can't have tab characters in your input file. So, be sure that your editor uses space characters when you press the tab key. The second rule has to do with using negative numbers. There can't be any spaces between the negative sign and the first digit of the number. The last formatting rule is that no line can be more than 132 characters. If any lines end up longer than that, ABINIT simply will ignore the extra content. If you get errors when trying to run your own jobs, those are the first few places you should check.

There are a massive number of input variables that allow you to control parameters around file handling, geometry, structure optimization and response functions, among many others. These input variables can be in any order. The entire file gets parsed before the calculations start. When you start creating

your own input files, you probably will want to be able to check them somehow. Luckily, you can use ABINIT itself to do this. The `abinit` executable includes an option (`-d` or `--dry-run`) to take your input files and validate them without starting the calculations. This allows you at least to catch major typos before wasting the time involved in doing a partial run and having it fail.

Along with your own input files, describing the geometry and other descriptive variables, ABINIT needs input files that describe something called the pseudopotential for your system. There are different types, such as Troullier-Martins or Hartwigsen-Goedecker-Hutter pseudopotentials, that can be used for different situations. Luckily, ABINIT includes pseudopotentials for the entire periodic table. This means you simply can build up your molecule by including the pseudopotentials for each of the different types of atoms in your system. Although it isn't necessary in most cases, you can create your own for some very specialized system if needed.

The other thing to be aware of is that ABINIT is released under a GPL license. This means you have access to all of the source code and can investigate exactly how the calculations are being done. When doing fundamental scientific research, that can be very important. You may be trying to do calculations in a region where the available algorithm is no longer valid. All of these calculations make assumptions to try to simplify the calculations so that they are actually doable, and it is very important to keep that in mind. But, with access to the code, you have the opportunity to make changes to those algorithms to fit the assumptions better that are valid for your problem. This open-source code gives you the ability to build on all of the past work and push it into new areas of research. Just remember to pass these extensions and improvements on to the next group of researchers to keep pushing our understanding forward.

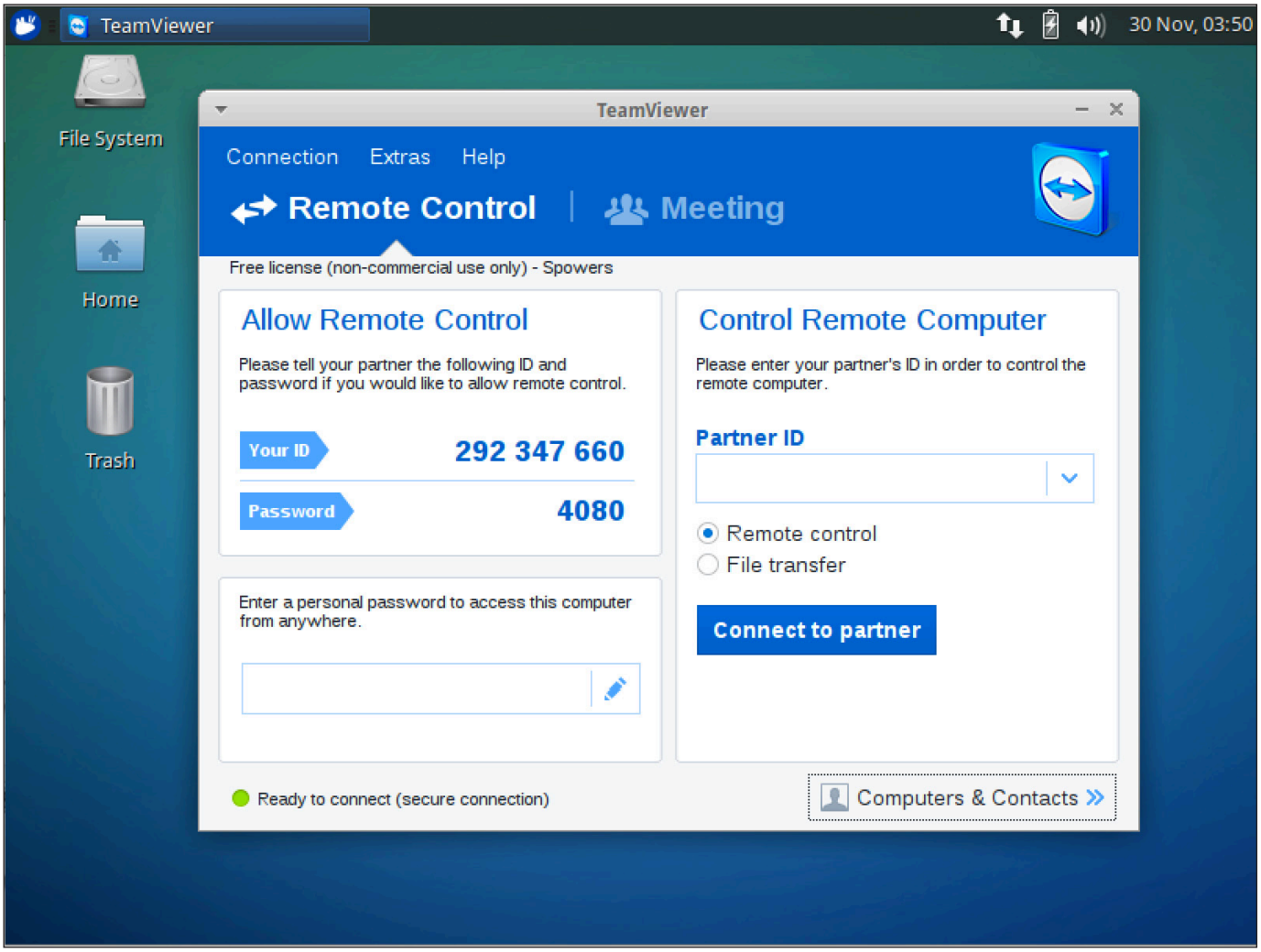
Interpreting the output from ABINIT can be a bit of a job. There is a lot of output describing how the calculated values progressed until they reached the requested accuracy to the actual answer. For example, if you are calculating the energy for a molecular configuration, you probably are interested in when the energy is

at its lowest value. This will be the most stable configuration for these nuclei and electrons. But, how do you interpret this output? Several tools are available to take the geometric portion of this output and plot it so that you can see what the configuration actually looks like. There also will be output describing how strong the various connections are between the nuclei, which you can use to see how reactive your molecule may be.

This is just a very basic introduction to what is involved when using ABINIT. Hopefully, you now feel a bit more comfortable digging in to the massive documentation and using ABINIT to solve whatever molecular problem you have. When you are ready, you can move on to much larger problems by using the MPI capabilities in ABINIT to use as many machines as you have available. —**JOEY BERNARD**

Help Me, Uncle Shawn

If you're anything like me, the holiday season is spent fixing Wi-Fi and removing spyware. Occasionally, I get to install Linux for a relative who is ready to give up Windows or needs something that will run on a circa-Windows 2000 computer (Xubuntu is usually my choice). The problem with helping friends and relatives with their computers over the holidays is that you become their first call when something goes wrong. You either can fight it



or make it easier on yourself by preparing in advance.

I love Team Viewer. It's not an open-source program, but it's free for personal use with no frustrating limitations. Plus, it runs on Windows, OS X and Linux. The best part is how easy it is to use. I generally don't set up the "automatic availability" feature that logs the computer in to the Team Viewer network automatically on boot. I like to use the standard startup, which requires users to call me with the code on their screen.

The best thing about Team Viewer is how easily it handles NAT situations. Since the software connects to the Team Viewer servers, those servers act like a connection broker, meaning there are no router ports to forward and no proxies to set up. As long as the computer is on-line, you should be able to take over and help someone. Again, you might not like the ease with which you'll be able to help, but having access to a user's computer in real time is so much nicer than explaining to Uncle Harry what "right click" means.

Due to its free license for personal use, cross-platform compatibility and incredible ease of use, Team Viewer gets this month's Editors' Choice award. It's not new software, but after a stretch of holidays, I'm reminded just how nice it is to have installed on all my relatives' computers. Be sure to install the client before you leave their houses, or else be prepared to explain software installation over the phone! Get your copy at http://teamviewer.com.

—SHAWN POWERS

REUVEN M. LERNER

Client-Side Performance

Give your users a better experience by improving client-side performance.

In my last few columns, I've covered different ways to understand, analyze and improve the performance of your Web applications. I've shown that between your network connections, server hardware, database design and HTTP server configuration, you can change and improve the performance of your Web application—well, sort of. Web applications, when they first started, were dynamic only on the server side. Sure, they output HTML—and later, CSS and JavaScript—but the overwhelming majority of the processing and computation took place on the server.

This model, of course, has changed dramatically in the last decade, to such a degree that you now accurately can claim to be a Web developer and work almost exclusively in HTML, CSS and JavaScript, with little or no server-side component. Entire MVC frameworks, such as Ember.js, Angular.js and React.js, assume that you'll be writing your application in JavaScript and provide you with the objects and infrastructure necessary for doing so.

If you're worried about the performance of your Web application, you need to concern yourself not only with what happens on the server, but also with what happens in the browser. Some commercial performance-monitoring solutions already take this into account, allowing you to see how long it takes for elements to render, and then to execute, on your users' browsers. However, there is also no shortage of open-source tools available for you to check and improve the ways in which your client-side programs are executing.

This month, I'm concluding this exploration of Web application performance with a survey of things to keep in mind, as well as tools that

help ensure that you're actually doing what you should be.

Client-Side Considerations

Client-side code is written in JavaScript. The code, whether inline in <script> tags or retrieved from a remote server, executes whenever the browser's parser gets to that part of the page. If you have JavaScript at the top of the page, it'll be executed when the parser gets to it, potentially delaying the rendering of the rest of your page. By contrast, if your JavaScript is at the bottom, the parser will execute it only after parsing and rendering the rest of the page. This is why so many developers learned to put their JavaScript commands inside a "document-ready" callback function; in that way, the code was executed only once the entire page had been loaded.

Because so many modern Web applications take place in JavaScript, the fact that you're often loading JavaScript from remote servers means that the time it takes to render a page depends not just on the server speed, the network bandwidth and the page's complexity, but also on the servers and networks serving such JavaScript, as well as those pages' complexity. As a result, it's generally considered to be good practice to load as many libraries as possible late in the game, at the bottom of your HTML page. That is, instead of having your <script> tags, whether local or remote, at the top of your page, you should put them at the bottom—unless it's vital to do otherwise.

Even better, you should consolidate your JavaScript files into a single file. This has a number of advantages. It means the user's browser needs to download a single file, rather than many of them. If you include all of the JavaScript needed on your site in a single file, it also means that the file needs to be loaded only a single time. On every subsequent page load, the JavaScript will be mentioned, but it won't be downloaded, because it'll already be cached in the browser's memory. You can make things even better, of course, by compressing that single JavaScript file. This turns out to be extremely effective, because compression algorithms work well with text, and especially with text that repeats itself, as happens with program code.

Better yet, you can run JavaScript code through a minimizer (or "minifier"), which removes comments, extraneous whitespace and anything else that isn't necessary for client-side programs to run. By minifying JavaScript files, combining the files and then compressing the resulting combination, you can dramatically

reduce the size of the JavaScript being sent to the user's browser and ensure that it is loaded only once per visit to your Web site.

UglifyJS, for example, can be installed via `npm`:

```
npm install uglify-js -g
```

You can run it on a file with:

```
uglifyjs FILENAME
```

Although because that sends output to stdout, you'll likely want to redirect it to a file:

```
uglifyjs FILENAME > ugFILENAME.js
```

I took the JavaScript from my PhD dissertation software and ran it through both uglifyjs and gzip. The original 36KB file was 8.5KB after compression, but 6.0KB after uglifying and compression. Although you might scoff at the small size of a 36KB file in the modern world, the fact is that each file takes time, for both the browser and the server. The faster you can get it off your server and into the browser, the better.

Download Time

Once the JavaScript is in the user's browser, things are both easier and harder to analyze. On the one hand, you (the developer) can download the program, test it and check the performance—and then, you also can use in-browser debugging tools to test and improve things.

One of the most important tools offered by both Chrome and Firefox is the display of files being sent to the browser. Even if your site appears to be loading and rendering quickly, a quick look at the download timeline almost certainly will be somewhere between surprising and shocking to you. You'll see how long it takes for each of the JavaScript (and CSS, and image) files to download and, thus, how much time it takes between the user requesting your page and the content actually appearing on it. This is a great way for you to identify potential bottlenecks and then reduce their effect on the slowness (or apparent slowness) of your site.

Even New Relic, which normally is considered a (commercial) server-side performance monitor, now offers some client-side performance checking. You place a small piece of JavaScript on your site; New Relic collects this information, and then tells you how long it took for your content to get to the user's browser and how long it took to render. This provides a surprisingly insightful view

of whether you need to work on improving the speed with which your files are delivered or to optimize the code further, such that it runs faster. There definitely are other options, but I've found that even the free (not open-source, but free of charge) New Relic client-side benchmarking to be quite useful and helpful.

Once you have combined and compressed your JavaScript files, you seriously should consider putting them, as well as any other static assets (such as CSS files and images), on a content distribution network (CDN). A CDN handles only static content, but given how many large, slow-to-download files are static, that often can provide a significant improvement. CDNs not only have a great deal of bandwidth, but they also copy your content to multiple servers, using the geographically closest one to serve content to your user. Thus, a user in Tokyo will receive data from a local CDN server, whereas a Chicago-based user will receive it from a different CDN server. So, using a CDN reduces the load on your main Web server, while decreasing the actual and perceived download times.

Benchmarking JavaScript

Although JavaScript has a (well deserved, I think) reputation for being a frustrating and quirky language, the fact is that modern JavaScript implementations run very quickly—assuming that you use the language in the right way. However, it's sometimes hard to know where your program is spending most of its time. Fortunately, the Chrome developer tools (a part of the Chrome browser) include a profiling tool. Go to the "profile" tab in the developer tools, and select the CPU option before visiting your site. You can run through your site for a few seconds or minutes before stopping the profiling from taking place. Once you've done that, you'll get a (very long) indication of which JavaScript programs were running, where they came from and how much time the CPU spent in each one.

You similarly can ask the profiler to examine memory usage. The more complex the application you're writing, the more necessary these tools will be. At the same time, you'll likely find that when you profile your JavaScript code, the most frequently used code probably will be code you didn't write yourself, but rather that is part of the underlying framework you're using.

In Firebug, the Firefox-based debugger, you can profile a page by going to the "console" tab and clicking on "profile". You'll see a

table showing how much time was spent in each function, and what percentage of the total time was spent there. If you're a Chrome user, you can open up the developer tools and click on the "profiles" tab. You'll then need to choose whether you want to check CPU performance or memory performance (in two different flavors). After starting and stopping the profiler, you can analyze the resources that JavaScript used—and then, of course, change your code appropriately.

One tool I have begun to use more frequently is PageSpeed from Google. This collection of tools would appear to be an SaaS, an updated version of YSlow, which was my go-to tool for many years. For example, Google's tools will tell you how mobile-friendly your site is.

Moreover, the PageSpeed results always point to documentation that describes, in great detail, why issues are problematic and what steps you can take in order to fix them. This documentation is surprisingly well written, and it points to very practical, clear suggestions for how to improve the performance of your JavaScript and CSS. After running PageSpeed against one of my client's sites, I found that we still had some blocking JavaScript higher up than is perhaps necessary. It also suggested which images could be compressed and how much space we would save in so doing.

Summary

Although server-side programming still is a vital part of the Web, the client is where much of the action is, and where the user often perceives lags and slowness. As a result, it's worth investing time to check your client-side performance and to address problems before your users start to complain (or leave you without complaining). Using a variety of tools to check your performance, as well as to reduce the size and time of JavaScript and CSS downloads, will go a long way toward improving your users' satisfaction with your site.■

Reuven M. Lerner trains companies around the world in Python, PostgreSQL, Git and Ruby. His ebook, "Practice Makes Python", contains 50 of his favorite exercises to sharpen your Python skills. Reuven blogs regularly at http://blog.lerner.co.il and tweets as @reuvenmlerner. Reuven has a PhD in Learning Sciences from Northwestern University, and he lives in Modi'in, Israel, with his wife and three children.

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

The Fourteenth Annual
Southern California Linux Expo

SCALE 14x

The Southern California Linux Expo has grown in size and scope since it began, and given this trend we will be in a new venue as of 2016.

We're happy to announce the dates and location for SCALE 14x...

January 21-24, 2016

PASADENA
CONVENTION CENTER

Pasadena, CA

Featured Speakers:
Jono Bacon
Cory Doctorow
Bryan Lunduke
Mark Shuttleworth

We are pleased to be hosting the return of the UbuCon Summit (formerly UDS)!

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

DAVE TAYLOR

Planetary Age

For his 120th column, Dave looks at time around the universe—programmatically, that is!

This marks my 120th column for *Linux Journal*. 120 times I've delved into the retro world of shell script programming. You've gotten ten years of my puns and wry asides—all available on the http://www.linuxjournal.com site for your reading pleasure!

At approximately 1,200 words/column that represents 144,000 words total. By comparison, *Harry Potter and the Philosopher's Stone* is 76,944 words. *The Hobbit* is 95,356 words, and *Pride and Prejudice* is 122,685. So there you have it. In those 144k words, I could have created a magical universe with an endearing young hero, or a different magical world with a plucky Hobbit adventurer (albeit reluctant adventurer), or I could have written about five daughters of marrying age, the charming Mr. Bingley and the haughty Mr. Darcy.

I have, of course, done none of those things. But on the bright side, I'm hoping you've been entertained while learning about shell script programming, programming techniques and how divide and conquer can help you solve even the most thorny of challenges. I'm sure Harry would approve!

This leaves me with the question: what should I do with the next ten years? That question's only slightly intimidating, of course, but given that UNIX is 45 years old (I'm using 1970 as the first launch date, back on an old PDP-7), and Linux is 24 years old (using 1991 as the date Torvalds begin developing his alternative to UNIX), there's still some time to go.

For this article, I thought it'd be fun to talk about space. Specifically, to write a script that would tell you what your age would be if you lived on one of the other planets in our solar system. So let's jump in!

Calculating Universe Age

Different planets revolve around the sun at different rates. Earth is easy. It takes the planet just a bit more than 365 days (a day being a full day/night cycle and based on the rotation of the

planet) to circle the sun—an Earth year.

But what about the other planets? We could calculate day-length and then number of days for each planet's "native" orbit, but it's more fun to have this be based on Earth days, although when we get to Pluto (yeah, I still consider it a planet), it's rather a long time.

Here's the data, in 24-hour Earth days:

```
Mercury   87.96
Venus     224.68
Earth     365.26
Mars      686.98
Jupiter   11.862 years
Saturn    29.456 years
Uranus    84.07 years
Neptune   164.81 years
Pluto     247.7 years
```

We will need to convert the last five into Earth days, but that's easy: just multiply by 365.26 (to be accurate), which gives us this better reference chart, presented as variables ready for a script:

```
mercury=87.96
venus=224.68
earth=365.26
mars=686.98
jupiter=4332.71412
saturn=10759.09856
uranus=30707.4082
neptune=60198.5006
pluto=90474.902
```

It sure takes a while for Pluto to circle our solar system, doesn't it? It's 90,474 days.

For this script, we'll want users to enter their birthdays, calculate how many days it's been since they were born, then simply divide that number by the "year" on each planet to calculate their universe ages.

I've actually written about how to calculate days between a specific date in the past and the current day, but rather than show the exhaustive calculation, let's just lean on the `date` function—more specifically GNU `date`. It's quite likely what you have on your computer already, and you can find out simply by typing:

```
>
$ date --version
date (GNU coreutils) 8.23
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later
<http://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and
redistribute it. There is NO WARRANTY, to the
extent permitted by law.
```

This is on the latest version of Ubuntu. Sadly, I'm going to be leaving you Mac users who have become

accustomed to working with my scripts in the dust this time. Unfortunately, Mac OS X still ships with the older POSIX version of `date` and therefore has no `date` math available. With GNU `date`, however, it's super easy to calculate the number of days you've been alive. Let's assume you, like me, were born on August 3, 1965 (my birthday, plus or minus a year or three). How many days have I been alive?

The one-liner:

```
daysalive=$(( ( $(date -ud 'aug 3 1965' +'%s') -
 ➥$(date -u +'%s') )/60/60/24 ))
```

This has my birthday hard-coded, probably not what we want, so instead it could be modified to work with user input, as a first step toward actually creating a script:

```
$ cat planetaryage.sh
#!/bin/sh
daysalive="$(( ( $(date -ud "$*" +'%s') -
 ➥$(date -u +'%s') )/60/60/24 ))"
echo "Born on $* means you've been alive $daysalive days."
exit 0
$ sh planetaryage.sh aug 3 1965
Born on aug 3 1965 means you've been alive -18354 days.
```

Negative days. It seems like something you'd get out of an old *Night Gallery* show or perhaps *Black Mirror*, to be a bit more contemporary. But why? Because we're subtracting the current date from the day in the past, not vice versa.

Flipping the math around in the equation solves the problem and gets the desired result:

```
Born on aug 3 1965 means you've been alive 18354 days.
```

Now the rest of the script is quite easy, particularly since we've already translated each and every planet's orbital duration into Earth days. To demonstrate, 18,354 Earth days would make me 26.71 (18354 / 686.98) Martian years old.

Here's the full script:

```
$ cat planetaryage.sh
#!/bin/sh

mercury=87.96; venus=224.68; earth=365.26
mars=686.98; jupiter=4332.71412; saturn=10759.09856
uranus=30707.4082; neptune=60198.5006; pluto=90474.902

planetaryAge()
{
  orbit=$1
  planetname=$2
  planetarydays=$( echo "scale=5;$daysalive / $orbit"|bc )
  echo "You are $planetarydays $planetname years old."
}

daysalive="$(( ( $(date -u +'%s') -
 ➥$(date -ud "$*" +'%s') )/60/60/24 ))"

planetaryAge $mercury "Mercury"
```

```
planetaryAge $venus   "Venus"
planetaryAge $earth   "Earth"
planetaryAge $mars    "Mars"
planetaryAge $jupiter "Jupiter"
planetaryAge $saturn  "Saturn"
planetaryAge $uranus  "Uranus"
planetaryAge $neptune "Neptune"
planetaryAge $pluto   "Pluto"

exit 0
```

It's actually not too complicated. When I run it for my own birthday, I find out I'm quite a spring chicken on Neptune:

```
$ sh planetaryage.sh aug 3 1965
You are 208.66302 Mercury years old.
You are 81.68951 Venus years old.
You are 50.24913 Earth years old.
You are 26.71693 Mars years old.
You are 4.23614 Jupiter years old.
You are 1.70590 Saturn years old.
You are .59770 Uranus years old.
You are .30489 Neptune years old.
You are .20286 Pluto years old.
```

What about duration since the signing of the Declaration of Independence? Those numbers are a bit more interesting:

```
$ sh planetaryage.sh july 4 1776
You are 993.79263 Mercury years old.
You are 389.05999 Venus years old.
You are 239.31993 Earth years old.
You are 127.24387 Mars years old.
You are 20.17534 Jupiter years old.
You are 8.12465 Saturn years old.
You are 2.84667 Uranus years old.
You are 1.45209 Neptune years old.
You are .96616 Pluto years old.
```

Ah, it was almost one Plutonian year ago. It's simple enough once you know the mathematical capabilities of GNU `date`, for sure. Without it, you definitely can cobble together a script that can calculate the number of days since a specified date in the past, but we've done that before!

Next Month

Now is a golden opportunity, as we head into our next decade of this column, for you to send in some puzzles, ideas or topics you'd like me to cover here (send e-mail to ljeditor@linuxjournal.com). Don't be shy!■

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 more generally at his tech site: http://www.AskDaveTaylor.com.

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

KYLE RANKIN

Preseeding Full Disk Encryption

Automation makes things faster, if you don't count all that work ahead of time.

Usually I try to write articles that are not aimed at a particular distribution. Although I may give examples assuming a Debian-based distribution, whenever possible, I try to make my instructions applicable to everyone. This is not going to be one of those articles. Here, I document a process I went through recently with Debian preseeding (a method of automating a Debian install, like kickstart on Red Hat-based systems) that I found much more difficult than it needed to be, mostly because documentation was so sparse. In fact, I really found only two solid examples to work from in my research, one of which referred to the other.

In this article, I describe how to preseed full-disk encryption in a Debian install. This problem came up as I was trying to create a fully automated "OEM" install for a laptop. The goal was to have an automated boot mode that would guide users through their OS install and use full-disk encryption by default, but would make the process as simple as possible for users. Normally, unless you are going to encrypt the entire disk as one big partition, the Debian installer makes you jump through a few hoops to set up disk encryption during an install.

In my case, I couldn't just use the full disk, because I needed to carve off a small section of the disk as a rescue partition to store the OEM install image itself. My end goal was to make it so users just had to enter their passphrase, and it would set up an unencrypted /boot and rescue disk partition and an encrypted / and swap. As an additional challenge,

My end goal was to make it so users just had to enter their passphrase, and it would set up an unencrypted /boot and rescue disk partition and an encrypted / and swap.

I also wanted to skip the time-consuming disk-erasing process that typically happens when you enable disk encryption with Debian, since the disk was going to be blank to start with anyway.

Unfortunately, although there is a lot of documentation on how to automate ordinary partitioning and LVM with preseeding (I actually wrote a whole section on the topic myself in one of my books), I had a hard time finding much documentation on how to add encryption to the mix. After a lot of research, I finally found two posts (and as I mentioned, one of them referenced the other) that described the magic incantation that would enable this. Unfortunately, the only supported mode for encrypted disks in Debian preseed requires the use of LVM (something I confirmed later when I read the source code responsible for this part of the install). That's not the end of the world, but it would have been simpler in my mind if it didn't have that requirement.

Since you need a basic unencrypted /boot partition to load a kernel and prompt the user for a passphrase, I had to account for both and preserve a small 2GB rescue disk partition that already was present on the disk. After that, the remaining / and swap partitions were encrypted. Here is the partition section of the preseed config:

```
d-i partman-auto/method string crypto

d-i partman-lvm/device_remove_lvm boolean true

d-i partman-lvm/confirm boolean true

d-i partman-auto-lvm/guided_size string max

d-i partman-auto-lvm/new_vg_name string crypt

d-i partman-auto/disk string /dev/sda

d-i partman-auto/choose_recipe select root-encrypted

d-i partman-auto/expert_recipe string                         \
      root-encrypted ::                                       \
              500 500 500 ext3                                \
                      $primary{ } $bootable{ }                \
                      method{ format } format{ }              \
                      use_filesystem{ } filesystem{ ext4 }    \
                      mountpoint{ /boot }                     \
              .                                               \
              2000 2000 2000 linux-swap                       \
                      $lvmok{ } lv_name{ swap }               \
```

```
        in_vg { crypt }                         \
        $primary{ }                             \
        method{ swap } format{ }                \
.                                               \
500 10000 1000000000 ext4                       \
        $lvmok{ } lv_name{ root }               \
        in_vg { crypt }                         \
        $primary{ }                             \
        method{ format } format{ }              \
        use_filesystem{ } filesystem{ ext4 }    \
        mountpoint{ / }                         \
.                                               \
2000 2000 2000 ext4                             \
        $primary{ }                             \
        method{ keep }                          \
        use_filesystem{ } filesystem{ ext4 }    \
        label{ rescuedisk }                     \
.

d-i partman-md/device_remove_md boolean true
d-i partman-basicfilesystems/no_mount_point boolean false
d-i partman-partitioning/confirm_write_new_label boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
```

If you've never worked with preseeding, this entire section of code probably looks incredibly foreign. As preseeding in general is documented well in a number of other places, I'm not going to bother breaking down every setting here. Instead, let me highlight the settings that matter for disk encryption. The most important one tells partman (the preseed partition manager) to use encryption:

```
d-i partman-auto/method string crypto
```

Next, because preseeded encrypted partitions need to use LVM, I must add LVM-specific preseed settings:

```
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-lvm/confirm boolean true
d-i partman-auto-lvm/guided_size string max
d-i partman-auto-lvm/new_vg_name string crypt
```

In the last of these settings, I told partman to create a new LVM volume group named `crypt` that I will use to store my encrypted partitions. Further down when I define my swap and root partitions, you can see where I defined the logical volumes by name and set what volume group they are in:

```
2000 2000 2000 linux-swap                       \
        $lvmok{ } lv_name{ swap }               \
        in_vg { crypt }                         \
. . .
500 10000 1000000000 ext4                       \
        $lvmok{ } lv_name{ root }               \
        in_vg { crypt }                         \
```

Once these settings were in place, I was able to preseed an install and

have disk encryption be almost fully automated, except that the installer prompted me for a passphrase, which I wanted.

The only missing piece to this automation was that the installer started overwriting the existing disk with random information. Now, there are good reasons why you may want to do this before setting up disk encryption, but in this case, the disk was blank beforehand, and I didn't want to wait the many hours it might take. Try as I might, no options to preseed this feature away seemed to work. After poring through the partman code to find the magic option, I finally resorted to patching the partman-crypto script on the fly in the middle of the install so that it skipped the erase process:

```
d-i partman/early_command \
       string sed -i.bak 's/-f $id\/skip_erase/-d $id/g'
/lib/partman/lib/crypto-base.sh
```

This is an ugly hack indeed, but it was the only way I was able to find that worked. With that in place, I was able have an automated partitioning recipe with full-disk encryption that skipped the disk-erasing section. My hope is that the next time other people need to do this and do a search on-line, they at least can find my article and the two other examples and won't have to burn so much time.■

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

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

SHAWN POWERS

Profiles and RC Files

Confused by profiles and bashrc? Read on!

I love Linux, and if you're reading this, chances are you do too. To be honest though, some aspects of the Linux environment are confusing. Near the top of the list for me is the profile system. Conceptually, it's simple. There are system-wide settings that all users inherit, and then there are individual settings people can set on their own. The problem comes when different distributions handle profiles in different ways, and the concept of login shells versus interactive shells comes into play. Usually, it's not something Linux users worry about. But, when you need to make a change, it can be extremely frustrating to figure out what is loaded in what order, and which is seen by login shells only, and so on.

Login Shells

First, let me clarify what I mean by login shells. You've probably noticed that sometimes in order to get to a terminal shell, you're prompted for a user name and password. Other times, you just click on the terminal icon, and you're presented with a terminal already logged in. You'll most often experience this when using a GUI desktop environment. Basically, if you're already logged in to your Linux desktop, and you open a terminal window, it's an interactive shell.

It doesn't have to be inside a graphical desktop environment, however. If you `ssh` in to a remote server, you're prompted for a user name and password (thus, a login shell). If you then type `bash` from inside that SSH session, you're starting a brand-new terminal, but this time, it's an interactive shell (notice you're not prompted for a password). Why it matters is something I'll talk about a little later, but for comprehension sake, just remember that if you're prompted for a user

Having a folder to add custom scripts is important, because if you have system-wide changes you'd like added to everyone's login shell, adding commands to the /etc/profile file is dangerous.

name and password, it's most likely a login shell. If you go directly to a bash prompt, it's most likely an interactive shell. The one fairly common exception to this is if you've set up SSH keys to log in automatically. In that case, even though you aren't prompted for a user name and password, it's still a login shell. It's a pretty safe bet that if you're using SSH to log in, it's a login shell.

The Login Shell Process

The login shell process is far more complicated than interactive shells, so I am going to go over that process first. I'm assuming your users have a bash shell assigned in their /etc/passwd files. It's the most common shell for users to have, so it makes sense to be familiar with its nuances.

Step 1: when you authenticate in to a login shell, the system looks for a file called /etc/profile. That file is a shell script that assigns a few environment variables all users should have set.

Step 2: the /etc/profile script usually ends by calling any shell scripts in the /etc/profile.d folder and executing them as well. Often it will run only shell scripts in /etc/profile.d that end with a .sh extension, so look at the /etc/profile script to see how files should be formatted to run properly. Having a folder to add custom scripts is important, because if you have system-wide changes you'd like added to everyone's login shell, adding commands to the /etc/profile file is dangerous. Any system updates affecting /etc/profile will overwrite your changes. If you simply add a custom file into the /etc/profile.d folder, it will be read by the updated /etc/profile script even if it's updated.

Step 3: the /etc/profile script also executes the user's personal profile. This part is a little messy, as the user profile might be called different things depending on distribution and/or user customization. In general, the system will try loading

the profile by name in this order:

- .bash_profile
- .bash_login
- .profile

If it finds a file with that name in the user's home directory, it executes it and stops. This means if you have a .bash_profile and .profile in your home directory, only the .bash_profile will be executed. This is useful to know if you want to customize your profile, but don't want to make changes to the original user profile assigned to you. By default in Ubuntu, every user has a .profile file, but not .bash_profile or .bash_login. So if you want to customize your profile, simply copy the .profile in your home directory to a file called .bash_profile, and make any changes you want to .bash_profile. Doing that will leave your original .profile intact and still will allow you to customize to your heart's content. Just remember, if you create an empty .bash_profile, the system will see that as your profile of choice and ignore your .profile file completely!

Step 4: finally, the last step along the login shell order of operations is the .bashrc file stored in the user directory. This is another script—this one called from the .profile script in Step 3. Note that if you customize your user profile settings, you'll want to make sure whatever profile file you use actually calls the .bashrc script. It's inside the .bashrc script where personal settings like a custom prompt and color settings go, along with command aliases you might want to set (more on those later).

Step 5: this step doesn't really take place after Step 4; rather, it sort of branches off at Step 1. The /etc/profile script starts the process for loading user profiles, but it also kicks off the process for executing the system-wide bashrc file. Here again various distributions name this file differently, but it's generally either a file called /etc/bashrc or /etc/bash.bashrc. In the case of Ubuntu, it's /etc/bash.bashrc, but historically, it's often /etc/bashrc. Note that unlike the user's .bashrc file, the system-wide bashrc file does not start with a period.

To add insult to injury, some systems don't actually execute the system-wide bashrc file for login shells, so if you don't see it called in the /etc/profile script, that means it's not going to execute for login shells. For the most part, however,

the /etc/profile on the majority of distributions does indeed call the system-wide bashrc file. Since you know the order with which profiles are loaded, you can investigate on your own system to see what is actually loaded during the login shell startup.

Interactive Shell Process

An interactive shell has a far simpler startup procedure. When a person opens an interactive shell (one that doesn't authenticate a user name or password), the following steps occur.

Step 1: the /etc/bashrc or /etc/bash.bashrc file is executed. This takes place whether or not it's referenced in /etc/profile. While a login shell automatically starts the /etc/profile script, an interactive shell automatically starts the /etc/bashrc (or /etc/bash.bashrc) script.

Step 2: the users' .bashrc from their home directory is executed. Again, like the system-wide bashrc file, this isn't called from a user profile; rather, it's executed directly by the interactive shell. So even if you've erased the reference to .bashrc from the .profile script, an interactive shell still will execute it.

And, that's it! An interactive shell doesn't look for any profile information at all, either system-wide or in the user directory. However, because an interactive shell is a "child" process of the login shell used to log in initially (either via GUI or SSH), it inherits all the profile information that initial login shell acquired. So although both the initial login shell and the "child" interactive shell have the same profile information loaded, the important distinction is that interactive shells don't reload the profile information. They never look at the profile scripts, so whatever information was loaded by that initial login script is all they have access to. (This distinction will be more important when you see what the scripts actually do.)

What Do Profiles Do?

First, a disclaimer: I can spell out only what is generally done with profiles and bashrc scripts. It's certainly possible for a person to change what is done by customizing either profiles or bashrc scripts. Generally, it's good practice to stick to the standards.

Profiles mainly are used to load environment variables. Since profiles are loaded by login shells, and login shells are the initial entry point into a system, that's the time when setting up the environment

makes the most sense. One of the biggest environment variables is the PATH variable. When a login shell is initiated, the PATH is set. Other environment variables also can be set in the system-wide profile or individual user profiles, but just know that the profile system is where most variables are set.

The order with which profile information is loaded is very important, because if you want to override the system-wide default profile information, you can do so by specifying environment variables in your personal user profile script. For instance, the PATH variable is usually modified by the user's profile script on login. Usually, the .profile (or .bash_profile, etc., see above) script will add ~/bin to the PATH variable if users have their own bin folder inside their home directory. Because user profiles are loaded after the system-wide profile, user settings take precedent and override system-wide settings.

What Do RC Files Do?

Again, this is a generalization, but the system-wide bashrc file and then the individual user's .bashrc script usually set personal preferences for the command line. If you want a custom prompt, or prefer a specific color scheme, the bashrc system is where that would be set. Much like the profile system, the user's .bashrc file overrides the system-wide bashrc (or bash.bashrc, again see above) settings. That means you can customize the behavior of the command line however you like without affecting other users on the system.

The most common customization inside the .bashrc file is to add aliases. An alias is sort of like text expansion, in that it substitutes your defined alias with whatever command you specify. For example, here's a snippet from a .bashrc file in the user's folder:

```
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
```

The aliases make it so that if the user types `ll` on the command line, the system will execute `ls -alF` instead. It's a great way to make shortcuts for commands with cryptic options or shortcuts for commands you type often.

Although I'm not suggesting tomfoolery, .bashrc aliases are also a great way to prank your fellow users if they leave their system logged in. Say you create an alias

like this:

```
alias ls='echo "Deleting all files..."'
```

Then, every time they type `ls`, they'll be in for a little (innocent) surprise! Yes, it's very easy to do nefarious pranks with aliases, but since we all log out when we leave our workstation, we shouldn't ever have to worry about it, right?!

Some Gotchas

Understanding how shells work really makes troubleshooting a lot easier. You've probably already realized a few things, but they're worth mentioning. If you make changes to any of the profile scripts, those changes won't be recognized until you start a new login shell. The same is true for .bashrc changes, but since you easily can close an interactive shell and start a new one, .bashrc changes are easier to activate.

One of the main problems regarding profile loading is that if you make a change to environment variable settings, like the PATH variable, you'll actually have to log out and log back in to test your changes. You certainly can set a path variable on an interactive shell, but remember, any new interactive shells will inherit the original login shell's profile settings, so often logging out and back in is really the only way to make permanent changes.

Also, although I went over it already, I want to reiterate that while the system-wide profile (/etc/profile) and the user profile generally call the bashrc scripts, they don't have to. If you make changes to your profile settings, it's possible that your login shells will behave very differently from your interactive shells. If that's your goal, great, but usually you want to make sure your login shells also execute the bashrc stuff, since that information is what makes the user experience more useful.

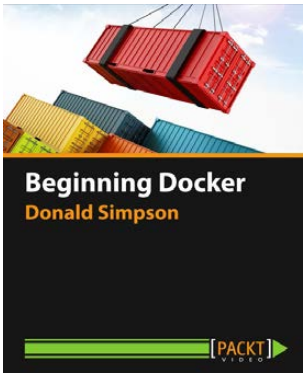
Finally, I want to add that the best way to understand and learn about profiles and RC files is to play with your system. Learning how to manipulate your settings is not only educational, but it can make your computing experience much more convenient.■

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.

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

DJI Manifold

Canonical's Ubuntu operating system serves as the "brains" for DJI Manifold, a new, high-performance embedded computer for drones that reduces processing time and optimizes real-time data analysis. Utilizing DJI's Onboard SDK, the Manifold is a user-friendly system that enables developers to create more powerful professional applications that leverage aerial and ground technologies to solve complex problems. Fully compatible with DJI's Matrice 100 drone, the Manifold is also compatible with third-party sensors and enables developers to connect a wide variety of onboard devices, such as infrared cameras, atmospheric research devices and geographical surveying equipment. Because the Manifold computer both collects and analyzes data in the air, it provides an efficient solution for developers in need of time-sensitive information. Relevant tech specs include Ubuntu 14.04 LTS version, quad-core ARM Cortex A-15 processor, NVIDIA Kepler-based GPU and support for CUDA, OpenCV and ROS.
http://www.dji.com



Donald Simpson's *Beginning Docker* (Packt Publishing)

If you're a developer seeking to learn how to deploy applications in containers using the open-source Docker, you have a new learning tool at your disposal. *Beginning Docker*, taught by automation build engineer and DevOps consultant Donald Simpson and published by Packt Publishing, is a two-hour-long, hands-on video course packed with practical examples to get one started with Docker and create amazing applications. Equipped with basic knowledge of Linux, viewers of *Beginning Docker* will learn how Docker works, how to set it up and how to get started on leveraging its benefits. From there, viewers create and share Docker images, install Docker on their own machines, learn to manage it effectively, and then progress to creating and publishing a custom application. Advanced topics include Docker containers, volumes, mounts, ports, linking and constraining containers, the Docker Web API, handling of complex automation processes with Shipyard and the creation of a mini-Heroku PaaS with slugbuilder and slugrunner. Packt describes the format of *Beginning Docker* as an easy-to-follow and structured video tutorial with practical examples of Docker to help viewers get to grips with each and every aspect.
http://www.packtpub.com

Rogue Wave Software's CodeDynamics

Helped on by a nudge from Rogue Wave Software's CodeDynamics tool, large-scale data modelling and analytics technologies traditionally reserved for HPC are making their way into new enterprise and industrial applications. These big data solutions in financial, energy, science, government and other commercial services require the same elements as HPC: extremely fast systems, highly optimized code and innovative algorithms. Enter CodeDynamics, Rogue Wave's next-generation dynamic analysis tool for data-intensive commercial applications that expands the reach of multithreaded debugging. CodeDynamics looks at complex C and C++ applications at execution time to help identify and correct bugs, memory issues and crashes, cutting "right to the chase" to identifying causes quickly. Enterprises that demand performance, scalability and high availability will find value in the deep thread control, unique reverse debugging and advanced data visualization features of CodeDynamics, adds Rogue Wave. An additional innovation in CodeDynamics is the ReplayEngine feature, built to simplify the troubleshooting process. By recording and saving program execution, ReplayEngine allows developers to work back from a failure, error or crash to find the root cause without repetitive restarts and stops, and it allows developers to store the recording to investigate the error at any time.

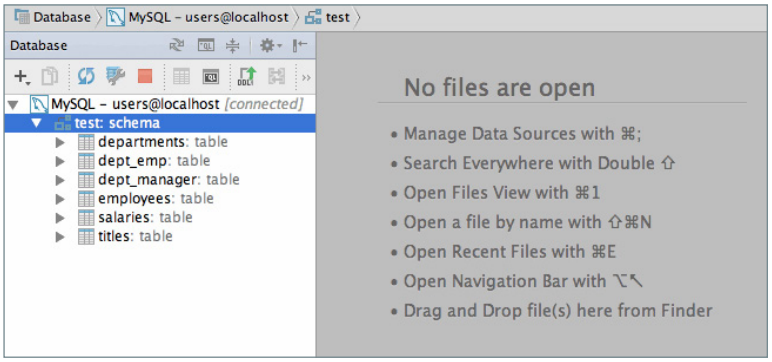
http://www.roguewave.com

IBM's API Harmony

IBM prefaced the announcement for its new API Harmony platform by stressing not just the critical importance of APIs for organizations to become more "cognitive" but also the value of the "API Economy", destined to be worth $2.2 trillion by 2018. To exploit the vast potential of APIs—that is, application programming interfaces—while navigating their risks, Big Blue presents API Harmony, a cloud service on the Bluemix Development Platform that acts as a matchmaker of APIs for developers and IT managers to facilitate the process of building new applications. Armed with advanced cognitive technologies like intelligent mapping and graph technology, API Harmony provides a unique developer experience to anticipate what a developer will require to build new apps, make recommendations, show API relationships and identify what is missing. IBM adds that when core information assets are packaged as APIs and shared or sold, enterprises build awareness, increase customer satisfaction through more personalized services and expand partner networks. API Harmony is one of many services and solutions from IBM that provide the foundation for clouds to behave as one, leading to more consistent cloud integration regardless of the cloud infrastructure.

http://www.ibm.com/apieconomy

JetBrains Toolbox



In this space, I typically cover about eight new products for your reading pleasure. This month, however, I feature more than double the normal output, thanks to a "big day" of updates from JetBrains s.r.o. The tool developer simultaneously upgraded the nine elements in its JetBrains Toolbox, thus smashing the *Linux Journal* New Products record for most products announced in a single issue. These nine elements include IntelliJ IDEA 15 IDE for Java, PhpStorm 10 IDE for Java, WebStorm 11 IDE for JavaScript, PyCharm 5 IDE for Python, AppCode 3.3 IDE for Objective-C on Mac OS X, CLion 1.2 cross-platform IDE for C and C++, RubyMine 8 IDE for Ruby and Rails, 0xDBE 1.0 IDE for DBAs and SQL Developers and ReSharper Ultimate 10 productivity tool for Visual Studio. In addition to the product improvements for each tool, JetBrains added a new "All Products" pack that allows customers to use any of the above products according to their current needs.

http://www.jetbrains.com

Undo Software to Deliver Support for 64-bit ARM Devices

Debugging, asserts Undo Software, is the number one challenge when moving existing code to new architectures like the 64-bit ARM v8-A. To simplify porting code to the ARM v8-A, Undo Software Ltd. and ARM have teamed up to produce a portfolio of advanced Linux and Android reversible debugging tools—the most recent and notable of which is the Undo Software's Live Recorder. Live Recorder's new "software-implemented trace" debugging technology helps simplify porting code from alternative hardware architectures by enabling Linux and Android programs to make a detailed recording of themselves while they are running. The recording, executed in a highly compressed and efficient way, contains everything needed for a developer to debug an exact copy of the bug as it occurred in production. This includes everything a program does, such as every memory access made and every instruction executed. This information can be used to run and step their programs backward and forward in time, enabling developers to fix bugs more quickly. Live Recorder delivers particular benefits to telecoms, IoT, enterprise server, HPC, mobile and automotive industries, the sectors most advanced in porting existing software to the 64-bit ARM architecture.

http://www.undo-software.com

1248's DevicePilot

Unlike Web or smartphone apps, connected Internet of Things (IoT) devices must be deployed into the physical world, where lots of things can go wrong. To overcome physical-world barriers that stand in the way of effective IoT at scale, IoT specialist 1248 unveiled DevicePilot, a new as-a-service solution for managing the growing IoT ecosystem. DevicePilot continuously monitors and manages connected devices over their complete life cycles and presents a simple dashboard showing how many devices have been deployed, where and by whom, and how many are not working and why. DevicePilot's automatic asset management, monitoring and lifetime support enable scaling projects from pilot stage to deployment with thousands or even millions of devices with universal coverage, from applications as variable as smart energy to smart homes and cities to transport systems, as well as industrial monitoring and control. DevicePilot is integrated with the ARM mbed IoT Device Platform, based on open standards, technology and services to accelerate wider adoption of IoT systems at scale. The goal of 1248 is to fill one of the few remaining gaps in the set of services required for successful IoT deployment—that is, in device management.
http://1248.io

SUSE OpenStack Cloud

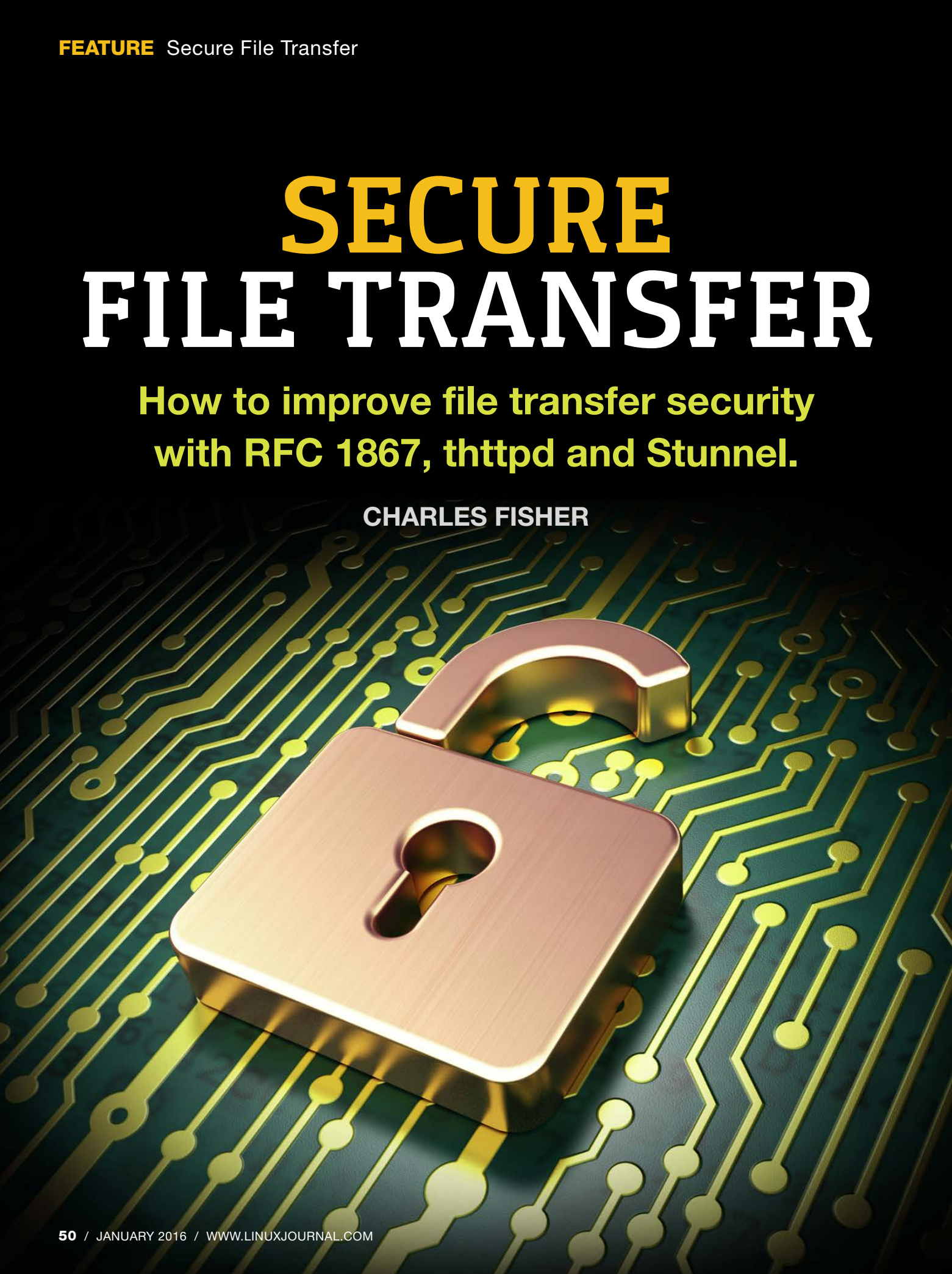
Based on the OpenStack release Liberty, the upgraded SUSE OpenStack Cloud 6 offers enterprise customers the latest features that further ease transition of business-critical applications and data to the cloud. SUSE OpenStack Cloud 6 is SUSE's solution for building Infrastructure-as-a-Service private clouds. In addition to high-availability enhancements and non-disruptive upgrades to future OpenStack releases, this new version also adds Docker for containerized applications and IBM z Systems mainframe support to existing support for Xen, KVM, Hyper-V and VMware hypervisor options. Finally, full support for OpenStack Manila provides direct access to the performance, scalability and management of the open-source Manila shared filesystem service.
http://www.suse.com

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

SECURE FILE TRANSFER

How to improve file transfer security with RFC 1867, thttpd and Stunnel.

CHARLES FISHER



File transfer between Linux systems (and perhaps all POSIX systems in general) is in some ways a neglected subject. The arcane protocols in common use are far from secure, and the SSH replacements offer too much power and complexity. Servers holding highly sensitive data (such as credit card numbers, SSNs, birthdates and so on) often must accept file transfers, but greatly restrict remote visibility and administration, which is hard with the well known tools.

File transfers with RFC 1867 (https://www.ietf.org/rfc/rfc1867.txt) can offer a number of benefits over most other methods: the highest security and optional encryption, all without requiring entries in /etc/passwd or other credentials for the operating system.

The tools I cover in this article to implement this protocol are sthttpd, an upload CGI utility, stunnel and curl. The examples here were developed on Oracle Linux 7.1, but most of the code is portable and should run on other platforms with minimal changes (with the exception of the systemd configuration).

Why Not FTP?

There have been substantial improvements in security and performance through the years in the FTP server software that is commonly bundled with Linux (https://security.appspot.com/vsftpd.html#security). It remains easy to configure FTP clients for batch activity with automatic logins:

```
echo machine a_server.com login YourName password
 ➥a_Password >> ~/.netrc
chmod 600 ~/.netrc
echo -e 'ls -l \n quit' | ftp a_server.com
```

Unfortunately, this is a terrible idea that gets progressively worse with the passage of time:

- The login, password and file payload are all sent in clear text over the wire in the normal configuration, and there are many utilities to capture them that might be used over an untrusted network.
- Classic FTP servers listening on port 21 must run as root. If attackers find and exploit a weakness, your OS belongs to them.
- In "active" FTP, the client and server switch roles in running the `connect()` and `listen()` system calls. This causes the TCP connections to open in both directions, introducing problems for firewalls.

- Unless the FTP server supports chroot() and it is individually and specifically configured for a target user, that user is able to fetch recursively all accessible files on the system that have world-read permission.

- An FTP account created for a few files can give visibility to just about everything. Most modern FTP clients allow such recursive transfers. An FTP user requires an entry in /etc/passwd on the server that creates an OS account. If not properly managed, this allows the remote user to log in to a shell or otherwise gain unwanted access.

- Password aging often is mandated in high-security environments, requiring synchronized password changes on the client and server (usually after a failed overnight batch run).

Later revisions to the FTP protocol do add TLS/SSL encryption capabilities, but it is unwise to implement them:

```
man vsftpd.conf | col -b | awk '/^[ ]*ssl_enable/,/^$/'
    ssl_enable
        If enabled, and vsftpd was compiled against OpenSSL,
        vsftpd will support secure connections via SSL. This
        applies to the control connection  (including  login)
        and also data connections. You'll need a client with
        SSL support too. NOTE!!  Beware enabling this option.
        Only enable it if you need it. vsftpd can make no
        guarantees about the security of the OpenSSL libraries.
        By enabling this  option, you are declaring that you
        trust the security of your installed OpenSSL library.
```

The reason for the above warning is that because the FTP server runs as root, it exposes the encryption library to remote connections with the highest system privilege. There have been many, many encryption security flaws through the years, and this configuration is somewhat dangerous.

The OpenSSH suite of communication utilities includes "sftp" clients and servers, but this also requires an account on the operating system and special key installation for batch use. The recommended best practice for key handling requires passwords and the use of an agent:

> Our recommended method for best security with unattended SSH operation is public-key authentication with keys stored in an agent....The agent method does have a down side: the system can't continue unattended after a reboot. When the host comes up again automatically, the batch jobs won't have their keys until

someone shows up to restart the agent and provide the passphrases to load the keys.—*SSH, the Secure Shell*, 2nd Edition, Daniel J. Barrett, Richard E. Silverman and Robert G. Byrnes.

Those who blindly rush from FTP to sftp due to security pressures do not understand the complexities of key generation, the ssh-agent and ssh-add. Forcing such sophisticated utilities on a general population that is attempting to migrate away from FTP is sure to end badly.

OpenSSH also extends the ability to run a shell to the client in the default configuration. It is possible to constrain a user to file transfers only and configure for a higher-security chroot(), but extensive modifications to the server configuration must be performed to implement this. The main focus of SSH is secure interactive login—file transfers are a sideline. The lack of "anonymous" sftp or keyed file dropoff highlight this (lack of) focus.

The classic Berkeley R-Utilities include an rcp program for remote file copy. This does eliminate the clear-text password, but improves little else. The use of these utilities is highly discouraged in modern systems, and they are not installed and configured by default.

None of the above programs work well for secure batch file copy when receiving files from untrusted sources, and for these reasons, let's turn to RFC 1867.

thttpd in a chroot()

RFC 1867 is the specification behind the "file upload gadget" found on Web pages. The HTML to implement the gadget is relatively simple:

```
<form action="script.cgi" enctype="multipart/form-data"
 method="post">
<input type="file" name="Whatever">
<input type="submit" value="Upload">
</form>
```

Various browsers render the gadget with a slightly different appearance, but the function is the same (Figures 1–3).

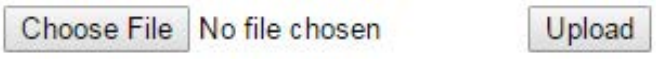


Figure 1. Google Chrome

Figure 2. Microsoft Internet Explorer

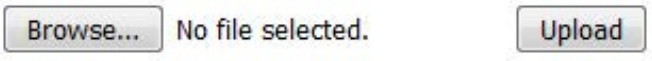


Figure 3. Mozilla Firefox

For this article, I will be using the "curl" non-graphical, command-line tool to perform file transfers using this protocol. Since the RFC 1867 protocol is implemented over HTTP, a Web server is needed. The server software choice here will be unconventional, for I'm going to require native support for the chroot() system call, which isolates running processes in the filesystem tree. This prevents access to powerful programs in /sbin and any other sensitive data stored in restricted locations.

Liberal use of chroot() and privilege separation recently saved OpenBSD's new mail system from disaster in a code audit (http://undeadly.org/cgi?action=article&sid=20151013161745):

> First of all, on the positive side, privileges separation, chrooting and the message passing design have proven fairly efficient at protecting us from a complete disaster. [The] Worst attacks resulted in [the] unprivileged process being compromised, the privileged process remained untouched, so did the queue process which runs as a separate user too, preventing data loss....This is good news, we're not perfect and bugs will creep in, but we know that these lines of defense work, and they do reduce considerably how we will suffer from a bug, turning a bug into a nuisance rather than a full catastrophe. No root were harmed during this audit as far as we know.

The common Web servers on Linux, Apache and Nginx repeatedly have refused to implement native chroot() security (http://www.openbsd.org/papers/httpd-asiabsdcon2015.pdf):

> OpenBSD has run its Web servers in a chroot for many years; Apache and nginx have been patched to run chroot'ed by default. These patches have never been accepted by upstream, but yet they provide a significant benefit.

Although this refusal precludes the use of Apache and Nginx in high-security applications, the recently updated sthttpd Web server (http://opensource.dyc.edu/sthttpd) does offer this capability. thttpd lacks many modern features (FastCGI, SPDY and SSL/TLS), but the native chroot() trumps the disadvantages. Here are the steps to download and install it:

```
wget ftp://opensource.dyc.edu/pub/sthttpd/sthttpd-2.27.0.tar.gz

tar xvzf sthttpd-2.27.0.tar.gz

cd sthttpd-2.27.0/


./configure
```

```
make install exec_prefix=/home/jail


mkdir /home/jail/etc

mkdir /home/jail/logs

mkdir /home/jail/htdocs

mkdir /home/jail/upload

chown nobody:nobody /home/jail/logs /home/jail/upload


echo 'port=80

dir=/home/jail

chroot

data_dir=/htdocs

#data_dir=/home/jail/httpd/htdocs

user=nobody

cgipat=**.xyz

pidfile=/home/jail/logs/thttpd.pid

logfile=/home/jail/logs/thttpd.log' > /home/jail/etc/thttpd.conf
```

Note above the `cgipat=**.xyz` for executing programs that adhere to the Common Gateway Interface (https://en.wikipedia.org/wiki/Common_Gateway_Interface). The thttpd documentation mentions using the conventional .cgi extension, but I suggest you pick your own random extension and rename any CGI applications that you deploy to make them harder to find and exploit by an attacker.

After you have installed the thttpd Web server, you can start a copy with the following command:

```
/home/jail/sbin/thttpd -C /home/jail/etc/thttpd.conf
```

If you point a Web browser at your machine (first try http://localhost—your firewall rules might prevent you from using a remote browser), you should see a directory listing:

```
Index of /
    mode  links  bytes  last-changed  name
    dr-x   2           6  Oct 22 22:08  ./
    dr-x   6          51  Oct 22 22:08  ../
```

If you wish, you can explore your new chroot() environment by downloading a copy of BusyBox. BusyBox is a statically linked collection of "miniature" UNIX/POSIX utilities, with several tools specific to Linux. When BusyBox binaries are prepared in such a way that they have no external library linkage, they are perfect for running inside a chroot():

```
cd /home/jail/sbin

wget http://busybox.net/downloads/binaries/busybox-x86_64

chmod 755 busybox-x86_64


ln -s busybox-x86_64 sh


cd ../htdocs

echo 'Keep out! This means you!' > index.html



echo '#!/sbin/sh


echo Content-type: text/plain
```

```
echo ""
/sbin/busybox-x86_64 env
echo "---"
/sbin/busybox-x86_64 id
echo "---"
/sbin/busybox-x86_64 ls -l /
echo "---"
/sbin/busybox-x86_64' > script.xyz

chmod 755 script.xyz
```

Notice first that an index.html blocks the directory list. Ensure that your CGI applications are protected this way, so they are not seen unless you have chosen to expose them as a `<FORM>` action. Also observe that a softlink was created from /sbin/busybox-x86_64 to /sbin/sh. Calling BusyBox with the link changes the program's behavior and turns it into a Bourne shell. The program examines `$argv[0]`, and if the contents match an "applet" that has been compiled into it, BusyBox executes the applet directly.

If you now load http://localhost/script.xyz with your browser, the shell script will run, and you should see:

```
GATEWAY_INTERFACE=CGI/1.1
SHLVL=1
REMOTE_ADDR=::1
HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
 ➥Gecko/20100101 Firefox/38.0
CGI_PATTERN=**.xyz
HTTP_ACCEPT=text/html,application/xhtml+xml,application/
➥xml;q=0.9,*/*;q=0.8
HTTP_HOST=localhost
SERVER_SOFTWARE=thttpd/2.27.0 Oct 3, 2014
PATH=/usr/local/bin:/usr/ucb:/bin:/usr/bin
HTTP_ACCEPT_LANGUAGE=en-US,en;q=0.5
SERVER_PROTOCOL=HTTP/1.1
HTTP_ACCEPT_ENCODING=gzip, deflate
REQUEST_METHOD=GET
PWD=/htdocs
SERVER_PORT=80
SCRIPT_NAME=/script.xyz
SERVER_NAME=localhost.localdomain
---
uid=99 gid=99 groups=99
---
total 0
drwxr-xr-x    2 0    0        24 Oct 22 22:08 etc
drwxr-xr-x    2 0    0        40 Oct 24 15:03 htdocs
drwxr-xr-x    2 0    0        40 Oct 22 22:10 logs
drwxr-xr-x    2 0    0        97 Oct 24 15:02 sbin
---
BusyBox v1.24.0.git (2015-10-04 23:30:51 GMT) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
  or: busybox --list[-full]
  or: busybox --install [-s] [DIR]
  or: function [arguments]...

    BusyBox is a multi-call binary that combines many common
    Unix utilities into a single executable. Most people will
    create a link to busybox for each function they wish to
```

```
    use and BusyBox will act like whatever it was invoked as.

Currently defined functions:
    [, [[, acpid, add-shell, addgroup, adduser, adjtimex, arp,
    arping, ash, awk, base64, basename, beep, blkid, blockdev,
    bootchartd, bunzip2, bzcat, bzip2, cal, cat, catv, chat,
    chattr, chgrp, chmod, chown, chpasswd, chpst, chroot, chrt,
    chvt, cksum, clear, cmp, comm, conspy, cp, cpio, crond,
    crontab, cryptpw, cttyhack, cut, date, dc, dd, deallocvt,
    delgroup, deluser, depmod, devmem, df, dhcprelay, diff,
    dirname, dmesg, dnsd, dnsdomainname, dos2unix, du, dumpkmap,
    dumpleases, echo, ed, egrep, eject, env, envdir, envuidgid,
    ether-wake, expand, expr, fakeidentd, false, fatattr, fbset,
    fbsplash, fdflush, fdformat, fdisk, fgconsole, fgrep, find,
    findfs, flock, fold, free, freeramdisk, fsck, fsck.minix,
    fstrim, fsync, ftpd, ftpget, ftpput, fuser, getopt, getty,
    grep, groups, gunzip, gzip, halt, hd, hdparm, head, hexdump,
    hostid, hostname, httpd, hush, hwclock, i2cdetect, i2cdump,
    i2cget, i2cset, id, ifconfig, ifdown, ifenslave, ifup, inetd,
    init, insmod, install, ionice, iostat, ip, ipaddr, ipcalc,
    ipcrm, ipcs, iplink, iproute, iprule, iptunnel, kbd_mode,
    kill, killall, killall5, klogd, less, linux32, linux64, linuxrc,
    ln, loadfont, loadkmap, logger, login, logname, logread,
    losetup, lpd, lpq, lpr, ls, lsattr, lsmod, lsof, lspci, lsusb,
    lzcat, lzma, lzop, lzopcat, makedevs, makemime, man, md5sum,
    mdev, mesg, microcom, mkdir, mkdosfs, mke2fs, mkfifo,
    mkfs.ext2, mkfs.minix, mkfs.vfat, mknod, mkpasswd, mkswap,
    mktemp, modinfo, modprobe, more, mount, mountpoint, mpstat,
    mt, mv, nameif, nanddump, nandwrite, nbd-client, nc, netstat,
    nice, nmeter, nohup, nslookup, ntpd, od, openvt, passwd, patch,
    pgrep, pidof, ping, ping6, pipe_progress, pivot_root, pkill,
    pmap, popmaildir, poweroff, powertop, printenv, printf, ps,
    pscan, pstree, pwd, pwdx, raidautorun, rdate, rdev, readahead,
    readlink, readprofile, realpath, reboot, reformime, remove-shell,
    renice, reset, resize, rev, rm, rmdir, rmmod, route, rpm,
    rpm2cpio, rtcwake, run-parts, runsv, runsvdir, rx, script,
    scriptreplay, sed, sendmail, seq, setarch, setconsole, setfont,
    setkeycodes, setlogcons, setserial, setsid, setuidgid, sh,
    sha1sum, sha256sum, sha3sum, sha512sum, showkey, shuf, slattach,
    sleep, smemcap, softlimit, sort, split, start-stop-daemon, stat,
    strings, stty, su, sulogin, sum, sv, svlogd, swapoff, swapon,
    switch_root, sync, sysctl, syslogd, tac, tail, tar, tcpsvd, tee,
    telnet, telnetd, test, tftp, tftpd, time, timeout, top, touch,
    tr, traceroute, traceroute6, true, truncate, tty, ttysize,
    tunctl, ubiattach, ubidetach, ubimkvol, ubirmvol, ubirsvol,
    ubiupdatevol, udhcpc, udhcpd, udpsvd, uevent, umount, uname,
    unexpand, uniq, unix2dos, unlink, unlzma, unlzop, unxz, unzip,
    uptime, usleep, uudecode, uuencode, vconfig, vi, vlock,
    volname, watch, watchdog, wc, wget, which, whoami, whois, xargs,
    xz, xzcat, yes, zcat, zcip
```

A few things to point out regarding each section above:

1. The environment in the first section above will include a `QUERY_STRING` if you have referenced it from a GET-method form—that is, if you append ?abc=123 to the URL, you will see `QUERY_STRING=abc=123` as standard GET-method parameters.

2. User 99 above actually is defined as nobody in the local /etc/passwd on the test system. Because there is no /etc/passwd file in the chroot(), all user IDs will be expressed numerically. If you want users to resolve to names for some reason,

define those users in a separate passwd file copy in the jail.

3. It is obvious that the root directory above is confined within the jail. These files also are resolving to numeric ownership—if an entry for root is placed in the passwd jail file, named owners will appear.

BusyBox is useful for exploring a chroot(), but it should not be left on a production server, as it introduces far too much power. This is confirmed on the thttpd Web site with words of warning on the contents of the jail (http://www.acme.com/software/thttpd/notes.html):

> Also: it is actually possible to break out of chroot jail. A process running as root, either via a setuid program or some security hole, can change its own chroot tree to the next higher directory, repeating as necessary to get to the top of the filesystem. So, a chroot tree must be considered merely one aspect of a multi-layered defense-in-depth. If your chroot tree has enough tools in it for a cracker to gain root access, then it's no good; so you want to keep the contents to the minimum necessary. In particular, don't include any setuid-root executables!

The recent "Towelroot" vulnerability demonstrated an ordinary C program compiled into a binary executable with no special permissions that was able to escalate privilege to root on a great many Linux systems by exploiting a mutex bug. If your jail includes the ability to download a binary image and mark it executable, such a flaw could allow an attacker to smash out of the jail and take ownership of your system. Beware of providing such tools.

If you would like to copy utilities from your host operating system for use in the jail, you can use the `ldd` command to find their shared object dependencies. For example, to move a functional copy of GNU AWK into the jail, examine the dependent objects:

```
# ldd /bin/gawk
    linux-vdso.so.1 =>  (0x00007ffe9f488000)
    libdl.so.2 => /lib64/libdl.so.2 (0x00007f7033e38000)
    libm.so.6 => /lib64/libm.so.6 (0x00007f7033b36000)
    libc.so.6 => /lib64/libc.so.6 (0x00007f7033776000)
    /lib64/ld-linux-x86-64.so.2 (0x00007f7034053000)
```

These object targets are usually soft links, requiring a chain of files and links to be moved, demonstrated by the library below:

```
# ll /lib64/libdl.so.2
lrwxrwxrwx. 1 root root 13 Mar 10  2015
 ➥/lib64/libdl.so.2 -> libdl-2.17.so
```

```
# ll /lib64/libdl-2.17.so
-rwxr-xr-x. 1 root root 19512 Mar  6  2015
 ➥/lib64/libdl-2.17.so
```

To copy these objects and re-create their links on Oracle Linux 7.1, follow these steps:

```
mkdir /home/jail/lib64
cd /home/jail/lib64

cp /lib64/libdl-2.17.so .
ln -s libdl-2.17.so libdl.so.2

cp /lib64/libm-2.17.so .
ln -s libm-2.17.so libm.so.6

cp /lib64/libc-2.17.so .
ln -s libc-2.17.so libc.so.6

cp /lib64/ld-2.17.so .
ln ld-2.17.so ld-linux-x86-64.so.2
```

Then, copy the gawk binary and create a test script:

```
cp /bin/gawk /home/jail/sbin

echo '#!/sbin/gawk -f

BEGIN {
print "Content-type: text/plain"
print ""
print "Hello, world!"
print ""
for(x in ENVIRON) print x,ENVIRON[x]
}' > /home/jail/htdocs/awk.xyz
chmod 755 /home/jail/htdocs/awk.xyz
```

If you load http://localhost/awk.xyz, you will see the output of the script above. This means that, with the added libraries, you are free to write CGI scripts in GNU AWK if you wish, even if you remove BusyBox:

```
Hello, world!

HTTP_ACCEPT text/html,application/xhtml+xml,
 ➥application/xml;q=0.9,*/*;q=0.8
AWKPATH .:/usr/share/awk
REMOTE_ADDR ::1
HTTP_ACCEPT_ENCODING gzip, deflate
SERVER_PORT 80
SERVER_PROTOCOL HTTP/1.1
HTTP_ACCEPT_LANGUAGE en-US,en;q=0.5
CGI_PATTERN **.xyz
SCRIPT_NAME /awk.xyz
HTTP_HOST localhost
GATEWAY_INTERFACE CGI/1.1
SERVER_SOFTWARE thttpd/2.27.0 Oct 3, 2014
SERVER_NAME localhost.localdomain
PATH /usr/local/bin:/usr/ucb:/bin:/usr/bin
HTTP_USER_AGENT Mozilla/5.0 (X11; Linux x86_64;
 ➥rv:38.0) Gecko/20100101
Firefox/38.0
REQUEST_METHOD GET
```

GNU AWK is not the best example as it does provide network

connectivity. Brian Kernighan's "One True AWK" might be a better choice at it lacks the extended network functions.

Let's consider additional startup parameters, using systemd to control the thttpd server. If you don't have systemd, examine the following unit file and replicate it with your init system. First, if you are still running thttpd, kill it:

```
kill $(</home/jail/logs/thttpd.pid)
```

Then, direct systemd to start it:

```
echo "[Unit]
Description=thttpd web service
After=syslog.target

[Service]
ExecStart=/bin/ksh -c 'ulimit -H -f 48828; ulimit
 ➥-H -m 48828; /home/jail/sbin/thttpd -C
 ➥/home/jail/etc/thttpd.conf'
Type=forking
#Restart=always

[Install]
WantedBy=multi-user.target" > /etc/systemd/system/
➥thttpd.service

systemctl start thttpd.service
```

Note above the `ulimit` commands executed by the Korn shell before launching thttpd (you may need to install this shell):

```
ulimit -H -f 48828
ulimit -H -m 48828
```

These commands set maximum limits for memory and files written by thttpd and all of its child processes. These are specified in blocks of 1,024 bytes and equate to 50 megabytes of maximum usage. These are hard limits that cannot be raised. The reason for imposing these limits will become clear in the next section.

The thttpd Web server records activity with the system syslog when able, but when running in a chroot(), the /dev/log socket does not exist unless created manually. The rsyslog dæmon can be instructed to listen on an additional socket in /home/jail/dev/log, like so:

```
echo '$ModLoad imuxsock
$AddUnixListenSocket /home/jail/dev/log
$umask 0000' > /etc/rsyslog.d/thttpd.conf

mkdir /home/jail/dev
chmod 755 /home/jail/dev
chcon -v --type=device_t /home/jail/dev

systemctl restart rsyslog.service
systemctl restart thttpd.service
```

After restarting, you should see thttpd entries in /var/log/messages. If you are running on an older Linux system that uses sysklogd, the following option will be of interest to you:

> -a socket: Using this argument you can specify additional sockets from that syslogd has to listen to [sic]. This is needed if you're going to let some dæmon run within a chroot() environment. You can use up to 19 additional sockets. If your environment needs even more, you have to increase the symbol MAXFUNIX within the syslogd.c source file.

You also may find it useful to move or copy the /home/jail/sbin/thttpd binary to a location outside of the chroot(). If a copy remains in the jail, it can be tested at startup and compared to the protected copy. If the files differ, your startup script can mail an alert that your jail has been compromised. The thttpd.conf file might be similarly treated.

Upload.cgi

In 2000, Jeroen C. Kessels released Upload-2.6.tar.gz, which you easily can find using the major search engines. Although the software is quite old, it is likely the most concise implementation of RFC 1867 available (from the perspective of system resources).

Assuming that you have a copy of Upload-2.6.tar.gz, run the default compile with these commands:

```
tar xvzf Upload-2.6.tar.gz
cd Upload-2.6/sources/
make
ldd upload
```

Note that the `ldd` command should not be run as root on untrusted software, as documented in the manual page (run the build as a regular, non-root user).

The final `ldd` above will list the shared object dependencies for the binary:

```
linux-vdso.so.1 =>  (0x00007ffcbe5e1000)
libc.so.6 => /lib64/libc.so.6 (0x00007fbeffdad000)
/lib64/ld-linux-x86-64.so.2 (0x00007fbf00183000)
```

If you previously loaded libraries above for GNU AWK, you will have all of the needed shared objects in place to run this program in the chroot(). If you have not elected to place copies of the shared objects in /home/jail/lib64, recompile the program with static linkage (assuming that your compiler is capable of it—some distributions lack the static libc.a):

```
gcc -static -O -o upload.static upload.c
```

Copy your chosen binary to /home/jail, and set the configuration:

```
cp upload /home/jail/upload.cgi
cp ../html/BadPage.html /home/jail/htdocs/test-fail.html
cp ../html/OkPage.html /home/jail/htdocs/test-good.html

sed 's/action=[^ ]*/action="test.xyz"/' ../html/index.html > \
    /home/jail/htdocs/test.html

cd /home/jail/htdocs
ln -s ../upload.cgi test.xyz

echo 'Config          = Default
  Root          = /upload
  FileMask      = *
  IgnoreSubdirs = YES
  Overwrite     = YES
  LogFile       = /logs/upload.log
  OkPage        = /htdocs/test-good.html
  BadPage       = /htdocs/test-fail.html
  Debug         = 0' > test.cfg
```

If you now point your browser at http://localhost/test.html, you will see a file upload form; test it with a random file. With luck, you should see a success page, and the file that you transferred should appear in /home/jail/upload. You also should see a log of the transfer in /home/jail/logs/upload.log.

You can use the curl binary for batch transfers with this mechanism—for example:

```
curl -F file=@/etc/passwd http://localhost/test.xyz
```

Curl should return the HTML to your standard output:

```
<html>
<body>
<center>
<h1>Success!</h1>
<hr>

File uploaded: passwd<br>
Bytes uploaded: 2024
<p>

</center>
</body>
</html>
```

Uploaded files were configured to be stored in /home/jail/upload in this case:

```
# ll /home/jail/upload
total 1012
-rw-r--r--. 1 nobody nobody 1028368 Oct 25 10:26 foo.txt
-rw-r--r--. 1 nobody nobody    2024 Oct 25 10:29 passwd
```

This configuration is powerful in the respect that it removes a client's ability to browse your file store if you so choose. In FTP or its descendants, the ability to `PUT` into a batch directory also grants `GET`; with this mechanism, you can constrain your

clients to transmit only, with no capability to catalog or retrieve any content.

One potential problem with this upload program is memory consumption. Let's examine the source code to upload.c:

```
/* Allocate sufficient memory for the incoming data. */
Content = (char *)malloc(InCount + 1);
...
p1 = Content;
RealCount = 0;
/* For some reason fread() of Borland C 4.52 barfs if the
   bytecount is bigger than 2.5Mb, so I have to do it
   like this. */
while (fread(p1++,1,1,stdin) == 1) {
  RealCount++;
  if (RealCount >= InCount) break;
  }
*p1 = '\0';
```

You can see above that the entire file is read from the network (off standard input) and stored in memory. This could be a potential "denial of service" exploit, thus the 50mb ulimits set at the end of the previous section. Adjust these ulimits to meet your needs but prevent abuse. It also might be possible to use the `tmpfile()` function to spool to disk instead of memory, but extensive modifications to the C code would be required.

Because there isn't much code behind upload.cgi, it is relatively easy to extend. Consider these additional blocks for the `ShowOkPage()` function:

```
if (strnicmp(p1,"<insert sha256sum>",18) == 0) {
    char scratch[BUFSIZ];
    FILE *H;

    *p1 = '\0';
    sprintf(scratch,"%s%s",Root,LastFileName);
    strcpy(s1, "/sbin/sha256sum '"); strcat(s1, scratch);
     ➥strcat(s1, "'");

    if((H = popen(s1, "r")) != NULL && fgets(scratch, BUFSIZ,
     ➥H) != NULL)
    { sprintf(s1,"%s%s%s",Line,scratch,p1+18); strcpy(Line,s1);
     ➥fclose(H); }
}

if (strnicmp(p1,"<insert md5sum>",15) == 0) {
    char scratch[BUFSIZ];
    FILE *H;

    *p1 = '\0';
    sprintf(scratch,"%s%s",Root,LastFileName);
    strcpy(s1, "/sbin/md5sum '"); strcat(s1, scratch);
     ➥strcat(s1, "'");

    if((H = popen(s1, "r")) != NULL && fgets(scratch,
     ➥BUFSIZ, H) != NULL)
    { sprintf(s1,"%s%s%s",Line,scratch,p1+15);
     ➥strcpy(Line,s1); fclose(H); }
}
```

Compiling in these fragments allows you to report the md5 and sha256 signatures of the on-disk data received from the client optionally, if you so specify in the template, enabling the client to confirm that the server's on-disk data is correct:

```
$ curl -F file=@Upload-2.6.tar.gz
 ➥http://localhost/test.xyz
<html>
<body>
<center>
<h1>Success!</h1>
<hr>

File uploaded: Upload-2.6.tar.gz<br>
Bytes uploaded: 2039<br>
sha256sum: bed3540744b2486ff431226eba16c487dcdbd4e60
➥2268349fdf8b7f1fb25ad38
/upload/Upload-2.6.tar.gz
<br>
md5sum: d703c20032d76611e7e88ebf20c3687a
 ➥/upload/Upload-2.6.tar.gz

<p>

</center>
</body>
</html>
```

Such a data verification feature is not available in any standard file transfer tool, but was easily implemented in this elderly code because the approach is flexible. Adding custom processing on file receipt would be a nightmare for FTP, but it's relatively straightforward for upload.cgi. The need for such features is echoed in Solaris ZFS, which performs aggressive checksums on all disk writes—controller firmware makes mistakes, and the ability to report such errors is mandatory for some applications. Also note the signatures for Jeroen C. Kessels' package above, and further be warned that md5 signatures are vulnerable to tampering (http://www.mathstat.dal.ca/~selinger/md5collision)—they are useful for detecting media errors, but they do not guarantee data to be free of malicious alteration.

Other useful changes to the upload.c code include a prefix always added to the incoming filename read from the .CFG file (I added this feature in three lines), and replacement of `strcat()`/`strcpy()` functions with the safer `strlcat()`/`strlcpy()` from OpenBSD's libc (http://www.sudo.ws/todd/papers/strlcpy.html).

There is also an extended CGI processing library (http://www.boutell.com/cgic) in the C programming language written by Tom Boutell (author of the GD graphics library). The CGI-C

library offers more efficient RFC 1867 processing (without excessive memory consumption), but building applications with it is beyond the scope of this discussion.

Stunnel TLS

Because thttpd has no encryption support, those living in areas where encryption is legal (http://www.cryptolaw.org) can use Michal Trojnara's stunnel "shim" network encryption dæmon to provide https services on port 443. First, install the package from the standard Oracle Linux repositories:

```
yum install stunnel
```

You also can install stunnel from source. The package pulled by yum is in fact quite old (4.56, one major version behind the current 5.25), but it also includes stunnel security contexts for SELinux, so it is recommended that you install the package even if you plan to build a newer release.

After installation, stunnel will require a keypair for TLS. The public portion of the key can be signed by a Certificate Authority (CA) if you wish, which will allow error-free operation with most browsers. You also may use a "self-signed" key that will present browser errors, but will allow encryption.

Free signed SSL certificates should be available by the time you read this article from the Web site for the Let's Encrypt project (https://letsencrypt.org). Instructions should appear shortly on how to generate and maintain signed keys that are honored as valid by most browsers. Preliminary documentation on the Let's Encrypt Web site indicates that the tools will use .PEM files, which likely can be used by stunnel.

If you want to purchase a valid key for stunnel, there is a guide on the stunnel Web site on having your key signed by a CA (https://www.stunnel.org/howto.html).

For more informal use, you can generate a self-signed key with the following commands:

```
cd /etc/pki/tls/certs
make stunnel.pem
```

The process of key generation will ask a number of questions:

```
Generating a 2048 bit RSA private key
.....................................+++
.............................+++
writing new private key to '/tmp/openssl.hXP3gW'
-----
```

```
You are about to be asked to enter information that will
be incorporated into your certificate request.
What you are about to enter is what is called a
Distinguished Name or a DN. There are quite a few fields
but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []:IL
Locality Name (eg, city) [Default City]:Chicago
Organization Name (eg, company)
 ➥[Default Company Ltd]:ACME Corporation
Organizational Unit Name (eg, section) []:Widget Division
Common Name (eg, your name or your server's hostname)
 ➥[]:darkstar
Email Address []:linus@posix.org
```

The key produced above will be set for expiration 365 days from the day it was created. If you want to generate a key with a longer life, you can call openssl directly:

```
openssl req -new -x509 -days 3650 -nodes -out
 ➥stunnel.pem -keyout stunnel.pem
```

The key will look something like this (abbreviated):

```
# cat /etc/pki/tls/certs/stunnel.pem
-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQC23m+w0BLxI2zB
/p8/TiuFcEurTLbLCQwcO/FE+vNcJpddckuF6/VgpBAJk+d9i7NZNqrjMH711H18
3AYhewZTCbRUMQE3ndaYEIxSt4Qhbm8XbfUfx6Fmg4CnWh/XzE7B8Z7XbHpwRQ4d
kQOzICzb1nt96QKdWoAob73+hv7qdi3UjJ3/20z3Cx5LWfWoa32Y50//tvBjBtcQ
H7QpiE2tfLWHTQ5tztkqVY/MZJWVgoT5LnqQlZeZB/C4izSYNo9EGAnw4ThaFJ/y
NdvmyK6sYaO3Dq4eFO78O+zzqyfhPCtcfb8lMuRTZa8uiv7ziVf0A3eGSwKYonUf
 ...
-----END PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIID/TCCAuWgAwIBAgIJALT/9skCvdR5MA0GCSqGSIb3DQEBCwUAMIGUMQswCQYD
VQQGEwJVUzELMAkGA1UECAwCSUwxEDAOBgNVBAcMB0NoaWNhZ28xGTAXBgNVBAoM
EEFDTUUgQ29ycG9yYXRpb24xGDAWBgNVBAsMD1dpZGdldCBEaXZpc2lvbjERMA8G
A1UEAwwIZGFya3N0YXIxHjAcBgkqhkiG9w0BCQEWD2xpbnVzQHBvc2l4Lm9yZzAe
Fw0xNTEwMzAwMzI2NTJaFw0yNTEwMjcwMzI2NTJaMIGUMQswCQYDVQQGEwJVUzEL
MAkGA1UECAwCSUwxEDAOBgNVBAcMB0NoaWNhZ28xGTAXBgNVBAoMEEFDTUUgQ29y
cG9yYXRpb24xGDAWBgNVBAsMD1dpZGdldCBEaXZpc2lvbjERMA8GA1UEAwwIZGFy
/JMRW5oa/+TFZIRcacTxgAw=
 ...
-----END CERTIFICATE-----
```

The PRIVATE KEY section above is the most sensitive portion of the file; ensure that it is not seen or copied by anyone you do not trust, and any recordings on backup media should be encrypted. The `BEGIN CERTIFICATE` section is presented to TLS clients when they connect to stunnel.

It also is wise to compute custom primes for the Diffie-Hellman key exchange algorithm, following guidance from the stunnel manual page:

```
openssl dhparam 2048 >> stunnel.pem
```

The previous command will add

another section to your stunnel.pem file that looks like this:

```
-----BEGIN DH PARAMETERS-----
MIIBCAKCAQEAoHi5jzY5ZVwGCFFm1EhVsePXxNwCSs/eQbaC3rc+iXENL8xk21uq
6eSwYIQWUeDN/h6wBBDe6dpFoNDJQeqKCmUa8aojGHnkcqsJBdVUKVF5/7rWb1Yi
TzvbeZt8UvYnNUErJEpgBMiKPDYipE2BZ6k61WwkK6WV6svGAHpIc3o/9kU+72uf
dPFaNIygAb2HLaJYvXq9OYGvrMsmyZTh3fnpg2RiZSVJf+i4BfyeLiYkwnSZozAS
2rQ4hf2E5WY6jiAcNZBLKvqR8lUuIaXd9+VkiCSV0c2pXzb2ElxOk8sheAHliwip
SaKC694z9l63eNKQW2J4WI97wkil0qa4MwIBAg==
-----END DH PARAMETERS-----
```

Once the key is in place in /etc/pki/tls/certs/stunnel.pem, a stunnel configuration file must be created, like so:

```
echo 'FIPS    = no
options = NO_SSLv2
options = NO_SSLv3
options = CIPHER_SERVER_PREFERENCE
ciphers =
ECDH+AESGCM:DH+AESGCM:ECDH+AES256:DH+AES256:ECDH+
➥AES128:DH+AES:ECDH+3DES:DH+3DES:RSA+AESGCM:
➥RSA+AES:RSA+3DES:!aNULL:!MD5:!DSS
syslog  = yes
#debug  = 6 #/var/log/secure
chroot  = /var/empty
setuid  = nobody
setgid  = nobody
cert    = /etc/pki/tls/certs/stunnel.pem
connect = 127.0.0.1:80' > /etc/stunnel/https.conf
```

The cipher settings above are from Hynek Schlawack's Web site on the subject (https://hynek.me/articles/hardening-your-web-servers-ssl-ciphers), and they represent the current best practice for TLS encryption. It is wise to visit this site from time to time for news and advice on TLS, or perhaps consider following Hynek's Twitter feed.

The `FIPS` and `NO_SSL` options above are the default settings starting with stunnel version 5. If you are running the version 4.56 package bundled with Oracle Linux, you must provide them for best practice TLS.

The above configuration sets stunnel as an inetd-style service that is launched for each connection. Each process will be in a chroot() in /var/empty. It also is possible to run stunnel as a standing dæmon that forks for each new client. If you do so, remember to restart the dæmon each time an OpenSSL update arrives and the chroot() might need more careful preparation. If you use the inetd approach, updates will apply to all new connections immediately after your updates, but you must use care not to exceed NPROC under high usage. There is a performance penalty for running in inetd-style, but the ease of security administration is likely worthwhile for all but heavy usage.

The following commands configure stunnel for inetd-style socket

activation under systemd:

```
echo '[Unit]
Description=https stunnel
[Socket]
ListenStream=443
Accept=yes
[Install]
WantedBy=sockets.target' > /etc/systemd/system/https.socket

echo '[Unit]
Description=https stunnel service
[Service]
ExecStart=-/usr/bin/stunnel /etc/stunnel/https.conf
StandardInput=socket' > /etc/systemd/system/https@.service

systemctl enable https.socket
systemctl start https.socket
```

At this point, use your browser to visit https://localhost, and you will see your index page. Visit https://localhost/test.html, and you can upload over a secure channel. You also can use curl:

```
curl -k -F file=@/etc/group https://localhost/test.xyz
```

Note above the `-k` option, which disables client CA validation for a server certificate. You will need this option if you are using a self-signed key or if your curl binary lacks access to an appropriate repository of CAs (provided by your operating system):

```
<html>
<body>
<center>
<h1>Success!</h1>
<hr>

File uploaded: group<br>
Bytes uploaded: 842<br>
sha256sum: 460917231dd5201d4c6cb0f959e1b49c101ea
➥9ead3ab91e904eac4d758ebad4a
/upload/group
<br>
md5sum: 31aa58285489369c8a340d47a9c8fc49
 ➥/upload/group

<p>

</center>
</body>
</html>
```

If you are using an older Linux distribution that uses xinetd, this configuration might prove useful:

```
service https
{
    disable       =   no
    socket_type   =   stream
    wait          =   no
    user          =   root
    server        =   /usr/sbin/stunnel
    server_args   =   /etc/stunnel/https.conf
}
```

And if you are in an environment that is still using inetd, this line will enable stunnel:

```
https stream nobody nowait root /usr/sbin/stunnel
 ➥stunnel /etc/stunnel/https.conf
```

If you have problems with stunnel, try using telnet to connect to port 443—you may see helpful status messages there. For example:

```
# cd /etc/stunnel/

# mv https.conf https.conf.tmp

# busybox-x86_64 telnet localhost 443
Clients allowed=500
stunnel 4.56 on x86_64-redhat-linux-gnu platform
Compiled/running with OpenSSL 1.0.1e-fips 11 Feb 2013
Threading:PTHREAD Sockets:POLL,IPv6 SSL:ENGINE,OCSP,
➥FIPS Auth:LIBWRAP
Reading configuration from file
 ➥/etc/stunnel/https.conf
Cannot read configuration

Syntax:
stunnel [] ] -fd  | -help | -version | -sockets
      - use specified config file
    -fd    - read the config file from a file descriptor
    -help     - get config file help
    -version  - display version and defaults
    -sockets  - display default socket options
str_stats: 1 block(s), 24 data byte(s), 58 control byte(s)
Connection closed by foreign host
```

Note that if you reference files (keys or configuration files) that are not in the standard paths above, the "enforcing" SELinux on Oracle Linux 7.1 might deny read permissions. If you see such errors in your syslog, try applying the following:

```
chcon -v --type=stunnel_etc_t /alternate/path/to/https.conf
chcon -v --type=stunnel_etc_t /alternate/path/to/stunnel.pem
```

If your local Linux firewall is enabled, you can open the port for stunnel on https and allow remote browsers to connect. If you leave port 80 closed, you are enforcing TLS-encrypted communication for all connections:

```
iptables -I INPUT -p tcp --dport 443 --syn -j ACCEPT
```

Please note that one drawback to https with stunnel is that the `REMOTE_ADDR` environment variable shown in the above CGI scripts always will be set to 127.0.0.1. If you want to determine the source of a particular connection or transfer from the thttpd logs, you must cross-reference them with the stunnel connection logs. However, this property might be useful for upload.cgi—if `getenv("REMOTE_ADDR") != "127.0.0.1"`, you should call the `exit()` function. The net effect

is that the Web site can be visible in clear text on both port 80 and via TLS on port 443, but file uploads will fail if attempted in clear text.

Finally, if your client must ensure the identity of the server, but you do not want to obtain a signed certificate, you can run a remote stunnel on the client that forces verification on a particular key.

Extract and save the `CERTIFICATE` section from your server's stunnel .PEM (abbreviated below):

```
-----BEGIN CERTIFICATE-----
MIID/TCCAuWgAwIBAgIJALT/9skCvdR5MA0GCSqGSIb3DQEBCwUAMIGUMQswCQYD
VQQGEwJVUzELMAkGA1UECAwCSUwxEDAOBgNVBAcMB0NoaWNhZ28xGTAXBgNVBAoM
EEFDTUUgQ29ycG9yYXRpb24xGDAWBgNVBAsMD1dpZGdldCBEaXZpc2lvbjERMA8G
A1UEAwwIZGFya3N0YXIxHjAcBgkqhkiG9w0BCQEWD2xpbnVzQHBvc2l4Lm9yZzAe
VYckA2gQ+70yXXxpFSD4n2ecq3ebNtej07zR2wAtAkt/JtuGiUjbl1m4ZFTPoTwr
 ...
xDYMcezEgopMzYMihv6CQ0CEU+qL+92CYtEDsd1hzn74SlBK9HMKjMLrbBZPhbE4
/JMRW5oa/+TFZIRcacTxgAw=
-----END CERTIFICATE-----
```

Transfer this file to your client, and set the client's stunnel configuration file:

```
echo 'FIPS    = no
client  = yes
verify  = 4
cafile  = /path/to/publickey.pem
[client-https]
accept  = 127.0.0.1:65432
connect = your.remote.server.com:443' >
➥stunnel-verify.conf
```

The configuration above will run on Windows and a variety of other platforms. If you are running on a UNIX variant, consider also adding the chroot() option in a similar manner as was set on the server. Note, however, that if you intend to use the HUP signal to reload the stunnel configuration, you must copy all of the required files inside the chroot() to which you have confined stunnel. Although this likely would never be done in an inetd-style configuration, this is one of several drawbacks for chroot() operation.

Clients then can point their browser at http://localhost:65432, and they will be routed over TLS to the remote Web server. The curl utility similarly can use the local 65432 port in clear text, allowing stunnel to handle the TLS session.

When client connections are launched, the client stunnel will open a connection to the server's port 443, then thoroughly exercise the server's key to ensure the correct identity and prevent a "man in the middle" from intercepting the transmitted data.

The curl utility does have a number of options to import certificate stores, but it does not

appear capable of verifying a specific certificate as I have just demonstrated with stunnel.

The author of stunnel also noted that the rsync utility and protocol can be used for anonymous write access. The standard network configuration for rsync is clear text, but rsync also can be wrapped in either OpenSSH or stunnel. A dated guide for transferring files with rsync over stunnel can be found here: http://www.netbits.us/docs/stunnel_rsync.html. A benefit of RFC 1867 is that curl is the only utility required for command-line transfers; a more complex configuration is required for an rsync binary to be wrapped in services provided by an stunnel binary.

Acknowledgement

Special thanks to Michal Trojnara, the author of stunnel, for his helpful comments on this article and his greater work in stunnel development. Commercial support, licensing and consulting for stunnel is available from his organization. Please visit http://www.stunnel.org/support.html for his latest release.

Hurry Up and Downgrade

The classic UNIX file transfer utilities are woefully inadequate to ensuring identity, integrity and privacy. FTP fails so completely with these questions that the only justification for its continued use is the general familiarity with the command set. It is unbelievable that modern IT has been thus constrained for so long.

We need a file transport protocol with casual dropoff, integrity checking, encryption, privilege separation (of the TLS state machine and the HTTP file processing), chroot() security and hooks for custom processing on arrival. There is no mainstream utility that offers all of those features.

Until such time, RFC 1867 will allow you to "roll your own" transport. While this protocol has many implementations (Perl, PHP, Python and so on), it is rare to find any with chroot() security. Hopefully, this does not remain the case.■

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.

Transferring Conserver Logs to Elasticsearch

Review and search serial console logs using Elasticsearch, Riemann and syslog-ng.

FABIEN WERNLI

If your organization manages Linux, AIX, HP-UX or Solaris servers in-house, chances are your system administrators at least occasionally need low-level access to those devices. Typically, administrators use some kind of serial console—for example, traditional serial port, Serial-over-LAN or Intelligent Platform Management Interface (IPMI). Managing and auditing console access is not trivial, so many organizations rely on the Conserver application to create session logs when accessing these servers via the serial console. These logs can be useful for various reasons—for example, maintenance or troubleshooting (to review why something crashed), security (to find out who did what—connecting user names to actual users) or compliance (to provide detailed session logs).

This article covers the following:

- How to parse and process serial console logs using syslog-ng Open Source Edition (Balabit).
- How to send the logs to Elasticsearch (Elastic), so you get a complete, searchable audit trail of the console access.
- How to integrate the console logs into a real-time monitoring system using Riemann.

Conserver

Conserver is a wonderful piece of software that lets you manage your infrastructure's serial consoles, whether they be old-style hardware serial ports or state-of-the-art Serial-over-LAN (SOL) baseboards. Its distributed design permits a decentralized user experience using a secure, TLS-encrypted protocol. The straightforward workflow consists of the user connecting to any conserver master node, which then forwards the traffic to the node that manages the console you want to access. As all masters share the same configuration file, it is very straightforward to redistribute consoles among servers automatically (provided they are virtual SOL devices, like IPMI) using configuration management (for example, using the Puppet module we developed at CC-IN2P3).

So where is the catch? As far as we are concerned at the Computing Centre of the National Institute of Nuclear Physics and Particle Physics (CC-IN2P3), the logging mechanism could be greatly improved, because conserver's design is quite ancient now. For example, it does not support logging to syslog. From a user perspective, logging is awesome, as you can use a keystroke to access the logs of the console, and the console

Useful syslog-ng Features

The syslog-ng Open Source Edition application is a syslog dæmon that allows you to collect your log data and much more—for example:

- **Flexibly collect, parse, classify and correlate logs from various sources.**
- **Send log data to message queues, including AMQP, STOMP or Apache Kafka.**
- **Store your log data in plain files, HDFS, SQL databases or MongoDB.**
- **Forward your log data to monitoring tools like Riemann, Redis or Graphite.**
- **Process CSV, JSON or plain-text messages.**
- **Rewrite, reformat and transform your log messages.**
- **You can write your own modules in C, Java or Python.**

logs contain the complete session. From an architecture perspective though, things are not so great, as every conserver master stores the logs of the consoles it manages locally in a file.

syslog-ng

This is where syslog-ng Open Source Edition comes into play. The idea is to transport the logfiles of the conserver masters to our favorite event store back ends, which are Riemann and Elasticsearch. They provide powerful real-time stream processing and long-term indexed storage capabilities, respectively. In addition, with syslog-ng, you simply can send the logs to Riemann and Elasticsearch directly; there is no need for any additional agents (like Logstash). To see how this system works before going into configuration details, watch this video: https://webcast.in2p3.fr/videos-syslog_ng_conserver.

The video shows what the user does in the console (in the top-right section of the screen), its effect on the real-time Riemann-dash dashboard (bottom-right) and the near-real-time Elasticsearch front end (Kibana|Elastic, on the left).

As you can see, the user activities and events of the session are transported to the back ends,

including useful metadata like `conserver.is_attached: true`. This tells you whether or not someone was attached to the console (which is obviously the case in this example).

Requirements

To create the system shown in the demo video, you need the following software:

- syslog-ng Open Source Edition 3.7.2 or newer.
- conserver (tested with 8.2.1).
- Riemann (tested with 0.2.10).
- Elasticsearch (tested with 1.6.0).

Note that this article does not cover how to install, configure (in general) and get the above software working. You can find plenty of related tutorials on-line. If you need help with these tasks, check the documentation, mailing lists or on-line forums for the software you need help with. The following sections of this article explain how to configure the components of this infrastructure for the specific needs of our scenario as an example.

Configuring Conserver

The conserver configuration in our setup is nothing special. It creates a unified logfile, which will serve as the glue between conserver and syslog-ng. You can activate the unified logfile using either of the following methods:

1) Run the conserver executable with the `-U /var/log/console.log` flag.

2) Use the following configuration block in conserver.cf:

```
<config * { unifiedlog * /var/log/console.log; }
```

You also can set the server's general logfile (where conserver stores the global messages that are unrelated to individual consoles)—for example, to `/var/log/conserver.log`.

Both `/var/log/conserver.log` and `/var/log/console.log` will be inputs for syslog-ng. You might want to take special care of the log rotation of these files. As you are sending them to Elasticsearch, there is no need to keep them for too long.

Configuring syslog-ng Open Source Edition

You need to install syslog-ng locally on each conserver master and on a central host (that is, your logserver) that will gather all the events from the conserver hosts. The local instances will parse, process and enrich the console output, while the central host will collect them and send them over

to the two back-end systems, Riemann and Elasticsearch. Note that you could get the same results using only local instances, but most people prefer to centralize first, for various reasons.

Configuring syslog-ng on the Conserver Hosts: The following is an example configuration file for running syslog-ng on the conserver hosts. As you can see, it has three sources:

- `s_internal` tracks the internal messages of syslog-ng (very handy for troubleshooting, stored only locally).
- `s_console` reads the logs of the individual consoles.
- `s_conserver` reads the global messages of the conserver master.

The `s_console` and `s_conserver` sources process conserver's unified logfile. Since the format of the console and conserver messages is different, we have to configure syslog-ng to parse them differently, then forward them to the central syslog-ng server (you can add any other sources as needed for your environment):

```
@version: 3.7

@include scl.conf

options {
    threaded(yes);
};

source s_conserver {
  channel {
    source {
      file(
        '/var/log/conserver.log',
        flags(no-parse)
      );
    };
    parser {
      csv-parser(
        columns(tmp.date,PROGRAM,PID,MESSAGE)
        delimiters(' :')
        quote-pairs('[]()')
        flags(greedy)
      );
    };
    rewrite {
      set('$(strip $MESSAGE)', value(MESSAGE));
    };
  };
};

source s_console {
  channel {
    source {
      file('/var/log/console.log');
    };
    junction {
      channel {
        filter{
          program('\*/div>);
```

```
        };
        rewrite {
          subst('\*/div>, '', value(PROGRAM));
          set(
            'true',
            value('.SDATA.console.is_attached')
          );
        };
        flags(final);
      };
      channel {
        rewrite {
          set(
            'false',
            value('.SDATA.console.is_attached')
          );
        };
        flags(fallback);
      };
    };
    rewrite {
      set('$PROGRAM', value(HOST));
      set('console', value(PROGRAM));
    };
  };
};

source s_internal {
  internal();
};

destination d_remote {
  network(
      "",
      transport(tcp),
      port(514),
      flags(syslog-protocol)
  );
};

destination d_internal {
  file("/var/log/syslog-ng.log");
};

log {
  source(s_console);
  source(s_conserver);
  destination(d_remote);
};

log {
  source(s_internal);
  destination(d_internal);
};
```

Global Conserver Logs—the `s_conserver` Source: If you are not familiar with syslog-ng, the `s_conserver` and `s_console` sections can be a bit intimidating. To better understand how they work, take a look at a sample message conserver produces:

```
[Thu Sep  3 22:29:52 2015] conserver (13550):
➥[node42] automatic reinitialization
```

The related source definition contains three blocks:

1. `source`: the file path and the

`no-parse` flag, which tells syslog-ng to read the logfile, but not to parse it, as a syslog message (because it is not exactly in syslog format).

2. `parser`: the `csv-parser` splits the message at the colons (:) and extracts the following fields: `tmp.date`, `PROGRAM`, `PID` and `MESSAGE`. The parser's other options ensure that the field values are parsed properly.

3. `rewrite`: defines a rewrite rule to remove leading and trailing spaces from the `MESSAGE` key. (If you find a way to omit this point using 2., please let me know.)

This configuration parses the above example message into the following structured data:

```
tmp.date: Thu Sep  3 22:29:52 2015
PROGRAM: conserver
PID: 13550
MESSAGE: [node42] automatic reinitialization
```

Console Logs—the `s_console` Source: Here are two example messages from two different consoles:

```
node03: ACPI: No handler for Region [POWR]
 ➥(ffff8808248bb150) [IPMI]

node66*: node66 login: root
```

The first one is an unattended message probably produced by an ACPI signal. The second one, as hinted by the * (asterisk) character appended to the name of the console, is a message produced while someone was attached to the console (using `console node66`). We will use this hint to produce additional metadata. The source consists again of three parts:

1. `source`: the file path, this time without flags. That way, syslog-ng will try to parse the message using the symbolic pattern `%{PROGRAM}: %{MESSAGE}`. As a result, `node03` and `node66*` will be parsed into the `PROGRAM` key.

2. `junction`: a construct with two mutually exclusive (hence the `final` and `fallback` flags) channels (similar to a "try:" "except" structure in Python). The two channels correspond to the two cases in the example: the first one for messages when someone is attached to the console (thus, the `PROGRAM` field contains an asterisk character), and the second for messages without anybody attached. To tell one case from the other easily (for example, when reviewing the messages in Elasticsearch), this information is stored in the `.SDATA.console.is_attached` key.

3. `rewrite`: Rewrites the `PROGRAM` and `HOST` fields to their sane content: `console` and the name of the console, respectively.

So in the above examples, the messages are parsed into the following structured data:

```
PROGRAM: console
HOST: node03
MESSAGE: ACPI: No handler for Region [POWR]
 ➥(ffff8808248bb150) [IPMI]
.SDATA.console.is_attached: false

PROGRAM: console
HOST: node66
MESSAGE: node66 login: root
.SDATA.console.is_attached: true
```

Forwarding the Logs to the Central syslog-ng server, `d_remote`: The rest of the syslog-ng configuration is simple: we just send the structured payload using the syslog IETF RFC5424 protocol (hence the `syslog-protocol` flag) to the central syslog-ng server. All RFC5424 keys, including `.SDATA.*`, are sent over to the central syslog-ng server automatically. The only part that we parsed from the conserver logs that is not transferred to the central server is the `tmp.date` field. Instead, we will use the time when syslog-ng processes the message (which is a good approximation for current logs). If you absolutely want to use the value from `tmp.date` (because, for example, you want to send old conserver logs to the remote server), you can use the date parser from the syslog-ng-incubator project.

Configuring syslog-ng on the Central Logserver: On the central syslog-ng server, we have to route the console and conserver messages received from the conserver hosts to the Riemann and Elasticsearch back ends. The following syslog-ng configuration does exactly that; the only adjustment is that it removes the `.SDATA.` prefix from the fields, so they are more readable:

```
@version: 3.7

@include scl.conf

options {
  threaded(yes);
};

block destination realtime (
  host()
  port(5555)
  type("udp")
  throttle(0)
  flush-lines(1)
)
{
  riemann(
    flush-lines(`flush-lines`)
```

```
    throttle(`throttle`)
    server(`host`)
    port(`port`)
    type(`type`)
    ttl("${ttl:-300}")
    host("$HOST")
    description("$MESSAGE")
    attributes(
      scope(all-nv-pairs)
      key(".SDATA.*"
        rekey( shift(7) )
      )
    )
  );
};

source s_remote_tcp {
  channel {
    source {
      network(
        transport(tcp)
        port(514)
        flags(syslog-protocol)
        tags("syslog")
        so-rcvbuf(8388608)
      );
    };
  };
};

source s_internal {
  internal();
};

destination d_elasticsearch {
  elasticsearch(
    index("syslog-${YEAR}.${MONTH}.${DAY}"),
    type("syslog"),
    flush-limit(1),
    template("$(format-json -s all-nv-pairs --rekey
     ➥.SDATA.* --shift 7 --key ISODATE)")
    cluster("elasticsearch")
    port(9300)
    server("localhost")
    client_mode("transport")
    time-zone("UTC")
  );
};

destination d_internal {
  file("/var/log/syslog-ng.log");
};

destination d_riemann {
  realtime(
    host("riemann"),
  );
};

log {
  source(s_remote_tcp);
  destination(d_riemann);
  destination(d_elasticsearch);
};

log {
  source(s_internal);
  destination(d_internal);
};
```

Conclusion

From this article, you have learned how to create a system that allows you to review serial console logs in real time and make them accessible for free-text searching on a modern user interface. This is helpful for maintenance and troubleshooting purposes, and also for meeting auditing and compliance requirements. To achieve these goals, conserver can be integrated with Riemann and Elasticsearch. To integrate these services, you can use syslog-ng Open Source Edition, a flexible log collecting and processing application that can collect and parse the log messages and forward them to the Riemann Elasticsearch back end.

Improvements

The syslog-ng application is very flexible and has powerful message-processing capabilities. If you learn a bit about its possibilities, you can find several ways to improve the configuration described in the article. Here are some ideas that you can do with syslog-ng:

- Write a smarter parser to extract the name of the console from server messages (where available).

Real-Time Monitoring with Riemann

Riemann helps you monitor distributed systems. It aggregates events from your servers and applications, and it allows you to combine and process these events with a powerful stream processing language. You can query the events and visualize the results of these queries on dashboards. To get notifications promptly, you also can trigger alerts (for example, in e-mail or SMS). Since the clients actively push the data into Riemann, your dashboards display up-to-date information (in contrast with other systems that only pull event data every few minutes). If you use the integrated WebSocket server, you even can have completely synchronous event handling, all the way from the event source to the browser.

- Correlate the console and server messages, extract the name of the user name from the server messages and add them to the console messages. That way, console events contain the name of the attached user, which makes troubleshooting and auditing easier.
- Configure alerts for consoles that are attached for too long.
- Use the date parser from the syslog-ng incubator project to use the timestamp that conserver adds to the messages.

Fabien Wernli (faxm0dem) has been administering Linux clusters at the Computing Centre of the National Institute of Nuclear Physics and Particle Physics (CC-IN2P3) for 10+ years. Among others things, he is an expert on performance-data monitoring and infrastructure management.

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

Resources

Demo Video: https://webcast.in2p3.fr/videos-syslog_ng_conserver

Conserver: http://www.conserver.com

Serial-over-LAN (SOL): https://en.wikipedia.org/wiki/Serial_over_LAN

Intelligent Platform Management Interface (IPMI):
https://en.wikipedia.org/wiki/Intelligent_Platform_Management_Interface

Puppet Module for Conserver, Developed by CC-IN2P3:
http://github.com/ccin2p3/puppet-conserver

CC-IN2P3: http://cc.in2p3.fr

Accessing Conserver Logs with a Keystroke: http://conserver.com/docs/console.man.html

syslog-ng Open Source Edition: http://www.syslog-ng.org

Riemann: http://riemann.io

Elasticsearch: http://elastic.co/products/elasticsearch

Riemann-dash: http://riemann.io/dashboard.html

Kibana: http://elastic.co/products/kibana

syslog IETF RFC5424 Protocol: https://tools.ietf.org/html/rfc5424

Date parser from the syslog-ng-incubator project:
https://github.com/balabit/syslog-ng-incubator/tree/master/modules/date

Using correlation in syslog-ng:
https://www.balabit.com/sites/default/files/documents/syslog-ng-ose-latest-guides/en/syslog-ng-ose-guide-admin/html/patterndb-correlation.html

Alerting for consoles that are attached for too long:
https://www.balabit.com/sites/default/files/documents/syslog-ng-ose-latest-guides/en/syslog-ng-ose-guide-admin/html/patterndb-actions-correlation.html

WEBCASTS

Maximizing NoSQL Clusters for Large Data Sets

Sponsor: IBM

This follow-on webcast to Reuven M. Lerner's well-received and widely acclaimed Geek Guide, "Take Control of Growing Redis NoSQL Server Clusters", will extend the discussion and get into the nuts and bolts of optimally maximizing your NoSQL clusters working with large data sets. Reuven's deep knowledge of development and NoSQL clusters will combine with Brad Brech's intimate understanding of the intricacies of IBM's Power Systems and large data sets in a free-wheeling discussion that will answer all your questions on this complex subject.

> **http://geekguide.linuxjournal.com/content/maximizing-nosql-clusters-large-data-sets**

How to Build High-Performing IT Teams — Including New Data on IT Performance from Puppet Labs 2015 State of DevOps Report

Sponsor: Puppet Labs

DevOps represents a profound change from the way most IT departments have traditionally worked: from siloed teams and high-anxiety releases to everyone collaborating on uneventful and more frequent releases of higher-quality code. It doesn't matter how large or small an organization is, or even whether it's historically slow moving or risk averse — there are ways to adopt DevOps sanely, and get measurable results in just weeks.

> **http://geekguide.linuxjournal.com/content/how-build-high-performing-it-teams-including-new-data-it-performance-puppet-labs-2015-state**

WHITE PAPERS

Comparing NoSQL Solutions In a Real-World Scenario

Sponsor: RedisLabs | Topic: Web Development | Author: Avalon Consulting

Specializing in cloud architecture, Emind Cloud Experts is an AWS Advanced Consulting Partner and a Google Cloud Platform Premier Partner that assists enterprises and startups in establishing secure and scalable IT operations. The following benchmark employed a real-world use case from an Emind customer. The Emind team was tasked with the following high-level requirements:

- Support a real-time voting process during massive live events (e.g., televised election surveys or "America Votes" type game shows).
- Keep voters' data anonymous but unique.
- Ensure scalability to support surges in requests.

> **http://geekguide.linuxjournal.com/content/comparing-nosql-solutions-real-world-scenario**

NEW Forrester Study!

Linux Management with Red Hat Satellite: Measuring Business Impact and ROI

Achieving Application Delivery Velocity with a 482% ROI

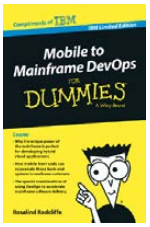
IBM commissioned Forrester Consulting to conduct its Total Economic Impact™ (TEI) study that examines and quantifies potential return on investment (ROI) for IBM UrbanCode Deploy within an enterprise DevOps environment. The study determined that a composite organization, based on the customers interviewed, experienced an ROI of 482%!

Read the Forrester Consulting study and learn learn how these enterprise organizations achieved:

- 97% reduction in the cost of releases.
- Reduction in the risk of failed deployments.
- 75% faster deployment times.

See how IBM UrbanCode brings deployment velocity while reducing release costs.

> **http://devops.linuxjournal.com/devops/total-economic-impacttm-ibm-urbancode**



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**

DOC SEARLS

What We Can Do with Ad Blocking's Leverage

We can do more than save publishing. We can start a renaissance for all of business—including publishing. We just need the code.

"Never waste a crisis", Rahm Emanuel is said to have said (http://www.nytimes.com/2008/11/10/us/politics/10obama.html?_r=0). And, publishers—including *Linux Journal*—have one now. According to PageFair and Adobe, the number of people blocking ads on their browsers has passed 200 million, worldwide, increasing annually by 42% in the US and 82% in the UK. Most of the blockers also block tracking, which is a main way that ads are aimed at readers through on-line publishers.

It's interesting to see how closely the rise in ad blocking follows the rise in discussion of surveillance-fed advertising. Here are the years when interest in those search terms first appeared, according to Google Trends. The ones in italic and boldface are not arcane to tracking-based advertising, but rather our response to it.

- 2005 — ad tag, mobile engagement.
- 2006 — ad-tech, search analytics.
- 2007 — behavioral targeting, retargeting, third party data, SEM tools, content analytics, microtargeting, ***do not track***.

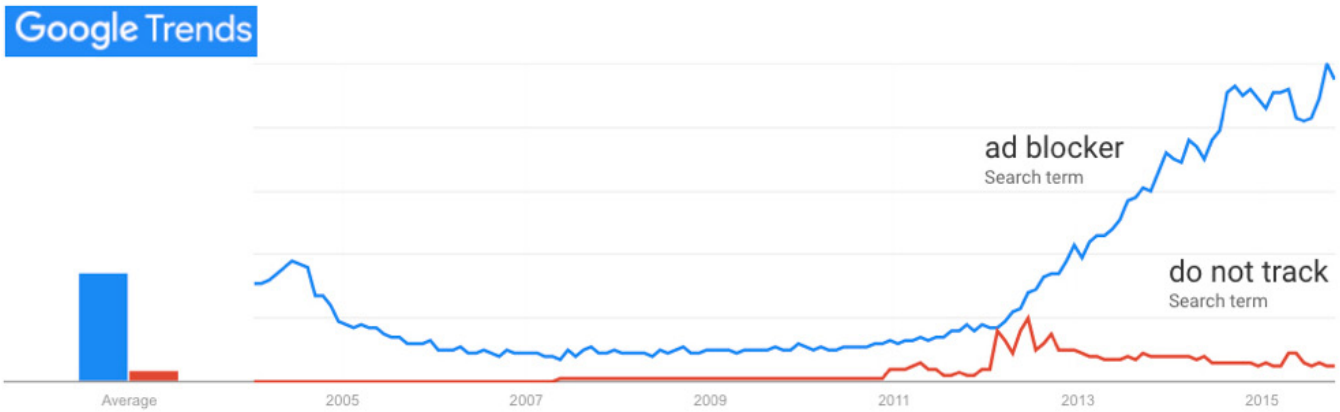


Figure 1. Google Trends Graph

- 2008 — deal id, ad fraud, social marketing management.
- 2009 — real time message.
- 2010 — demand side platform, cross-device, advertising beacon, social ad network, predictive marketing.
- 2011 — in-stream, real time bidding, creative optimization, search retargeting.
- 2012 — clickstream data, data management platform, mobile reengagement, native advertising, ***adblock war***.
- 2013 — programmatic marketing, programmatic advertising, subscription push, agency trading desk, content marketing platform.
- 2014 — supply side platform, data aggregators.
- 2015 — cross device tracking.

The Google Trends graph shown in Figure 1 makes clear how people reacted to all this, especially after Do Not Track failed.

The titles of ad blocking research studies also tell a story (see Resources for links). First came Ad-Blocking Measured, published by ClarityRay (later acquired by Yahoo) in 2012. Then PageFair brought us The Rise of Adblocking, Adblocking goes mainstream and The Cost of Adblocking, in 2013, 2014 and 2015.

The catch-all term for tracking-based advertising is *adtech*, and nobody has studied or written more wisely about it than Don Marti, former

Editor-in-Chief of *Linux Journal*. When I asked Don to define adtech, he said:

> Adtech can have narrow or broad definitions.
>
> The narrowest definition is a system that implements the Fundamental Value Proposition of Adtech (as defined by Michael Tiffany, http://www.whiteops.com/company)—redirecting advertising spending from high-value sites to low-value sites by tracking users. (See "Targeting Failure: legit sites lose, intermediaries win": http://zgp.org/targeted-advertising-considered-harmful/#targeting-failure-legit-sites-lose-intermediarieswin.)
>
> An economic definition of adtech would be any system that relies on information about the user to reduce the signaling value of an advertisement.
>
> Google has search ads that have some adtech built in to them, but could work without it and probably better. Some other Google ad products are pure adtech.
>
> Facebook ads are pretty close to pure adtech.

He defines signaling this way:

> In a market with asymmetric information, signaling is an action that sends a credible message to a potential counterparty (https://en.wikipedia.org/wiki/Signalling_%28economics%29).
>
> Advertising spending is a form of signaling that shows that a seller *has* the money to advertise (which the seller presumably got from customers, or from investors who thought the product was worth investing in), and *believes* that the product will earn enough repeat sales to justify the ad spending.

By blocking ads and tracking, users are marking down the value of digital advertising to that of spam while also giving themselves a great deal of leverage, both individually and collectively.

How will they use that leverage? I see two ways: 1) encouraging advertising with high signal value; and 2) signaling their own intentions, which will be far more valuable than adtech's expensive guesswork could ever be.

The highest signal value is in old-fashioned brand advertising. This is the uncomplicated kind that sustained

all of commercial publishing and broadcasting for centuries, never tracked anybody, and still makes up most of the world's advertising (including your monthly *Linux Journal*). At worst, it's annoying and wasteful; at best, it's useful, interesting and a form of art. (Seen any beautiful adtech lately? Or ever?)

For Madison Avenue, the mania around adtech has marginalized the creative side of advertising, but I expect that to end when the adtech bubble bursts, which it will, inevitably, given the steady growth of ad and tracking blocking. Toward that, Don sees some synergies:

> IMHO, creative ad people and creative Web people could have an **awesome** conference if we managed to exclude all surveillance marketing from it…Web people who think that "advertising" is creepy are just as mixed up as advertising people who think that "the Web" is creepy (http://zgp.org/~dmarti/business/fresh-start/#.VkusW9DvMUU). "Big Data" could be much, much more useful for everybody if people would only stop thinking of it as a way to automate the carny trick of putting chalk marks on audience members' backs (http://www.heyrubecircus.com/fun-facts/origins-of-the-carnie-lingo-mark), and put it to uses that people agree on and don't have to be hidden or made confusing. (Clicking "I agree" is not agreeing.)
>
> Every set of new technologies has the obvious, "hey we could do THIS with it" application, and surveillance marketing is the one that a lot of people have come up with for the Internet. But once we can get past it—and make the Net more trustworthy for more people—there are a lot better opportunities out there.

For expressing intent, we already have *intentcasting*. Here the user, as a prospective customer, tells the market what she is ready to buy (or, if she is already a customer, what needs service).

I see intentcasting as a cornerstone of The Intention Economy, a market development I first envisioned in 2006 and wrote up here in *Linux Journal*. That same year I started ProjectVRM at Harvard's Berkman Center, with the intention of making The Intention Economy happen. I wrote a book by the same title in 2012, reporting on

progress thus far and forecasting what it would do for the business world. Here's the gist, from its introduction:

> Over the coming years customers will be emancipated from systems built to control them. They will become free and independent actors in the marketplace, equipped to tell vendors what they want, how they want it, where and when—even how much they'd like to pay—outside of any vendor's system of customer control. Customers will be able to form and break relationships with vendors, on customers' own terms, and not just on the take-it-or-leave-it terms that have been *pro forma* since Industry won the Industrial Revolution....
>
> Relationships between customers and vendors will be voluntary and genuine, with loyalty anchored in mutual respect and concern, rather than coercion. So, rather than "targeting", "capturing", "acquiring", "managing", "locking in" and "owning" customers, as if they were slaves or cattle, vendors will earn the respect of customers who are now free to bring far more to the market's table than the old vendor-based systems ever contemplated, much less allowed.

Likewise, rather than guessing what might get the attention of *consumers*—or what might "drive" them like cattle—vendors will respond to *actual intentions* of *customers*. Once customers' expressions of *intent* become abundant and clear, the range of economic interplay between supply and demand will widen, and its sum will increase. The result we will call the *Intention Economy*.

This new economy will outperform the *Attention Economy* that has shaped marketing and sales since the dawn of advertising. Customer intentions, well-expressed and understood, will improve marketing and sales, because both will work with better information, and both will be spared the cost and effort wasted on guesses about what customers might want, and flooding media with messages that miss their marks. Advertising will also improve.

The volume, variety and relevance of information coming from customers in the Intention Economy will strip the gears of systems built for controlling customer behavior, or for limiting customer input. The quality of that

information will also obsolete or re-purpose the guesswork mills of marketing, fed by crumb-trails of data shed by customers' mobile gear and Web browsers. "Mining" of customer data will still be useful to vendors, though less so than intention-based data provided directly by customers.

In economic terms, there will be high opportunity costs for vendors that ignore useful signaling coming from customers. There will also be high opportunity gains for companies that take advantage of growing customer independence and empowerment.

Nine years after starting ProjectVRM and three years after that passage was published, we have 23 intentcasting developers listed on the ProjectVRM wiki. Here they are, with descriptions from their literature:

- About2Buy (https://about2buy.wordpress.com) — "A Collaborative Commerce System to Align Internet Buyers & Sellers Via Multiple Channels of Social Distribution."
- Crowdspending (https://www.crowdspending.com) — "...gives each of us the power of all of us."
- GetHuman (https://gethuman.com) — "Need to contact a company? Or have them call you? Get customer service faster and easier."
- Greentoe (https://www.greentoe.com) — "Finally...There's a New Way to Shop! Name Your Price & We Negotiate For You."
- HomeAdvisor (http://www.homeadvisor.com) — "We help you find trusted home improvement pros."
- Indie Dash Button (http://www.homeadvisor.com) — "This... turns traditional advertising on its head, and removes the need for complicated targeting technology. Customers readily identify themselves, creating more valuable sales channels where guesswork is all but eliminated." (Open source.)
- iNeed (http://www.ineedapp.com) — "Your own personal assistant."
- Intently (http://intently.co) — "Request any service anywhere with Intently.co."
- Instacart (https://www.instacart.com) — "The best way to shop for groceries—delivered from the

stores you love in one hour."

- Magic (https://www.getmagicnow.com) — "Text this phone number to get whatever you want on demand with no hassle...."
- Mesh (http://www.meshwithbrands.com) — "Connect with only the things you love....See ads from brands that matter to you. And block the ones that don't."
- MyTime (https://www.mytime.com) — "Book appointments for anything."
- MyWave (https://www.mytime.com) — "'Frank...your very own personal assistant', puts you in control of getting personalised experiences anytime, anywhere, on any device."
- Nifti (http://www.nifti.com) — "One simple place to track prices on the products you love."
- Pikaba (http://www.pikaba.com) — "Pikaba is Social Shopping Platform that captures consumer intent to purchase and connects them with the right local business."
- PricePatrol (http://pricepatrolapp.com) — "monitors nearby stores for what you want at the price you want."
- RedBeacon (http://www.redbeacon.com) — "Trusted pros for a better home."
- TaskRabbit (https://www.taskrabbit.com) — "Tell us what you need, let us know what we can take off your plate, choose a Tasker, hire one of our fully vetted Taskers to get the job done."
- Thumbtack (https://www.thumbtack.com) — "We help you hire experienced professionals at a price that's right."
- TrackIf (https://trackif.com) — "Track your favorite sites for sales, new items, back-in-stock, and more."
- Webofneeds (http://researchstudio-sat.github.io/webofneeds) — "A distributed marketplace driven by customer needs." (Open source.)
- yellCast (http://yellcast.com) — "What you want, where you want it."
- Zaarly (https://www.zaarly.com) — "Hire local, hand-picked home services. We moderate every job and guarantee happiness at virtually any cost."

In "Let's scale #intentcasting", posted on the ProjectVRM blog in November, I call for "an open-source way to scale across multiple vendors with the same signaling method", while also wondering "if there is a semantic-ish approach to Intentcasting. By that I mean a vernacular of abbreviated simple statements of what one is looking for—for example: '2br 2ba apt 10019' means a two-bedroom and two-bath apartment in the 10019 zip code."

In the comments below, Bill Wendell of Real Estate Café adds this to my example above:

> Intentcasting in real estate could deliver billions in consumer savings annually and open up the choke hold on inventory (small number of active listings relative to demand). There's also an important role for a large scale "4th party" to advocate for a consumer-centric open ecosystem as the industry transitions into the future (https://blogs.law.harvard.edu/vrm/2011/04/13/fourth-parties-and-vrm):
>
> http://bit.ly/Back2Billions
>
> If you follow @CRTLabs you'll see Realtor IoT projects, and an Om.ie-like (http://customercommons.org/2013/04/25/meet-omie-a-truly-personal-mobile-device) initiative they're calling "Rosetta Home". I'm generally a critic of the industry but http://RESO.org is an independent standards group and their own hack (PlugFest) suggests that their members might be open to FutureCommerce.

Kevin Cox of Welcomer then offers this:

> Both vendors and customers want the Intention Economy: vendors have the intention to sell us goods and services and customers have the intention to buy the vendor's goods and services.
>
> Ad blocking frees us to communicate with whom we want. Ad blocking is better termed ad choice and ad choice is part of the intention economy.
>
> Ad blocking is a precursor to a change in electronic communication. Ad blocking, spam filters, silent telephone numbers, "no junk mail" signs, are what happens when we get control over who and when and with whom we communicate and connect.
>
> The Permanent Web (http://IPFS.io) gives both vendors and customers the

ability to create electronic permanent identities. Welcomer is an early permanent identity technology built on top of the Permanent Web. The permanent identity from Welcomer is a set of distributed connections where the connections are peer to peer. There is no central directory. There is no single identity. There are just peer to peer connections where each party has rights. This approach is likely to prevail because the connections are simple and include all existing id systems. Putting many connections of different types together leads to complexity in the resulting emergent structures. These are the conditions under which complexity and adaptability to fit the environment evolve.

There are many ways to build a permanent web and we can expect there to be many variations on IPFS.io such as CloudOS.

Those two comments address one vertical market (real estate) and one horizontal market enabler (identity). Meanwhile, on Twitter, James Ladd of MyWave (one of the intentcasting companies listed above) pushes back against the notion that some generalized answer covering all of both (as I had hoped for) might be coming: "same protocols/src is a pipe dream - just build already!" He's right, so far. (And building well.)

Yet lots of pipe dreams do come true. Back in the 1980s, when I worked with Sun Microsystems and other competing UNIX makers, I despaired that there would ever be a free (as in freedom) *nix in the world. Too many giant tech companies were rolling their own, while weirdly trying to reconcile all of them to AT&T's SVR4 (System 5, Release 4). Then Linux happened. And the Web. And universal e-mail, file transfer and the rest of it. Can we do it for intentcasting?

Don't tell me. Just point to whatever work is happening.■

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.

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

Resources

Ad-Blocking Measured: http://www.slideshare.net/arttoseo/clarity-ray-adblockreport

PageFair's The Rise of Adblocking:
http://downloads.pagefair.com/reports/the_rise_of_adblocking.pdf

PageFair's Adblocking goes mainstream:
https://blog.pagefair.com/2014/adblocking-report

PageFair's The Cost of Adblocking: https://blog.pagefair.com/2015/ad-blocking-report

Don Marti: http://zgp.org/~dmarti

Don Marti on Signaling:
http://zgp.org/targeted-advertising-considered-harmful/#signaling

Doc Searls' *Linux Journal* article "The Intention Economy" (2006):
http://www.linuxjournal.com/node/1000035

ProjectVRM: http://blogs.law.harvard.edu/vrm

Harvard's Berkman Center: https://cyber.law.harvard.edu

The Intention Economy by Doc Searls, published in 2012:
http://www.amazon.com/The-Intention-Economy-Customers-Charge/dp/1422158527

ProjectVRM Wiki's List of 23 Intentcasting Developers:
http://cyber.law.harvard.edu/projectvrm/VRM_Development_Work

"Let's scale #intentcasting" by Doc Searls (November 2015):
http://blogs.law.harvard.edu/vrm/2015/11/14/lets-scale-intentcasting

Semantic Web (Wikipedia): https://en.wikipedia.org/wiki/Semantic_Web

Real Estate Café: http://realestatecafe.com

Bill Wendell's comments on Doc's "Let's scale #intentcasting" post:
http://blogs.law.harvard.edu/vrm/2015/11/14/lets-scale-intentcasting/comment-page-1/#comment-26913

Kevin Cox: http://www.welcomer.me/welcomer/?author=55ea61f8e4b05e14ae8bc98b

Welcomer: http://www.welcomer.me

Kevin Cox's comments on Doc's "Let's scale #intentcasting" post:
http://blogs.law.harvard.edu/vrm/2015/11/14/lets-scale-intentcasting/comment-page-1/#comment-26916

James Ladd: https://twitter.com/jamesladd

James Ladd's Twitter post: https://twitter.com/jamesladd/status/665695483529072640