Yubikey | PAM | Capistrano | MythVideo | MinorFs | Botnets | Samba

LINUX JOURNAL

Since 1994: The Original Magazine of the Linux Community
JANUARY 2009 | ISSUE 177

MinorFs for Discretionary Access Control

Managing Videos with MythVideo

Get Lazy with Capistrano

SECURITY

- COLD BOOT ATTACKS
- SECURITY ASSESSMENT STRATEGY
- ONE-TIME PASSWORDS WITH YUBIKEY
- IMPLEMENT SECURITY CHECKS WITH PAM

REVIEWED:
Behringer BCF2000

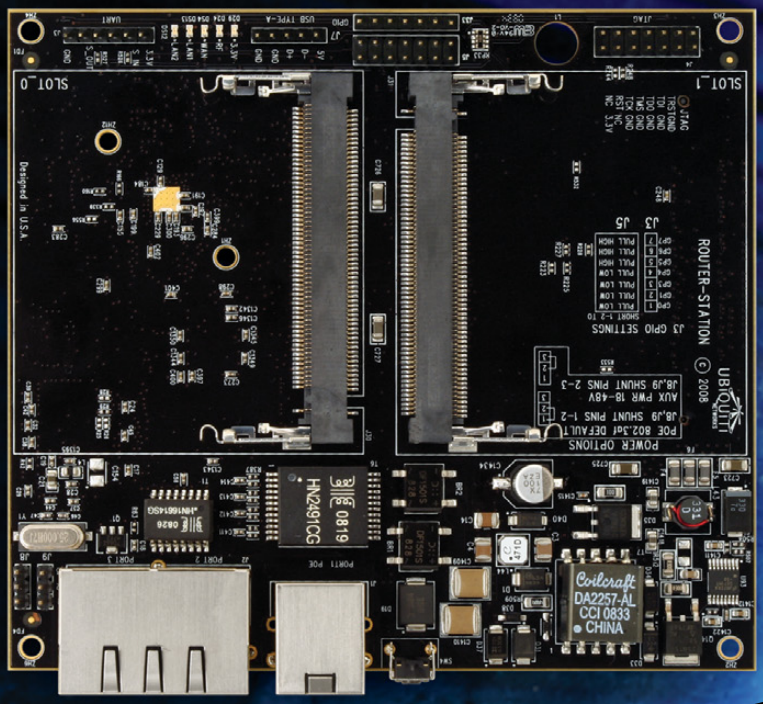
www.linuxjournal.com
$5.99US $5.99CAN
01
0 09281 03102 4

UBIQUITI NETWORKS

The Embedded Wireless Dream Machine.

680MHz CPU, 64MB RAM, 16MB Flash, USB, 3 mini-PCI slots, 5A power supply for multiple hi-power radios, and enhanced temperature performance and ESD protection for carrier applications. Full Linux SDK support and a $59 USD MSRP.

RouterStation



RouterStation UI CHALLENGE

CASH PRIZE

$200000

www.ubnt.com/challenge

www.ubnt.com

RouterStation UI Challenge: For Contest Details Visit: www.ubnt.com/challenge
Cash Prize $200000 (Two hundred Thousand) USD.

User Interface development based on OpenWrt Linux firmware. OpenWrt is a registered trademark of OpenWrt.

The website you've been wishing for...

As the world's largest web host, 1&1 offers website plans for every skill level and budget.

Start your website today and get a Holiday Credit of up to $300!*

Offer ends December 31st!

DOMAINS

Up to a $5 credit!*

Register your website today! Prices start at just $8.99/year.

WEB HOSTING

Up to a $50 credit!*

Design your professional looking website. Starting at just $3.99/month.

E-COMMERCE

Up to a $100 credit!*

Set up your own online store and start selling! Prices start at just $9.99/month.

SERVERS

Up to a $300 credit!*

Powerful hardware for high performance needs. Starting at $99.99/month.

*Credit dependent on package selected. Setup fee and minimum contract term may apply. Visit www.1and1.com for full promotional offer details. Credit cannot be redeemed for cash. Offer ends December 31, 2008. Product and program specifications, availability and prices subject to change without notice.

MEMBER OF united internet

Call **1.877.go1and1**

Visit us now **1and1.com**

1&1

CONTENTS

JANUARY 2009
Issue 177

FEATURES

46 YUBIKEY
Learn how to increase system and on-line security.
Dirk Merkel

56 COLD BOOT ATTACK TOOLS FOR LINUX
Use open-source tools to dump and scan RAM from a target system for encyption keys and other goodies.
Kyle Rankin

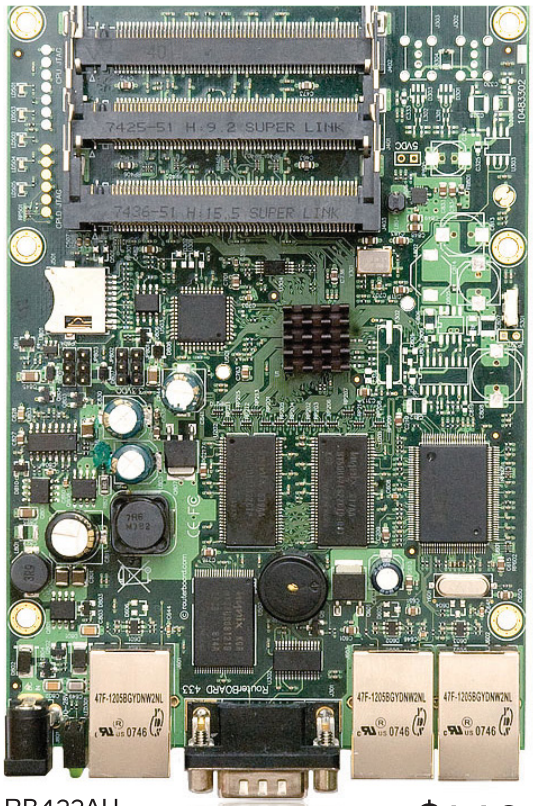
60 PAM—SECURING LINUX BOXES EVERYWHERE
How to implement Linux security checks.
Federico Kereki

66 TESTING THE LOCKS: VERIFYING SECURITY IN A LINUX ENVIRONMENT
Four checks for a more secure network.
Jeramiah Bowling

ON THE COVER
- MINORFS FOR DISCRETIONARY ACCESS CONTROL, P. 72
- MANAGING VIDEOS WITH MYTHVIDEO, P. 84
- GET LAZY WITH CAPISTRANO, P. 90
- COLD BOOT ATTACKS, P. 56
- SECURITY ASSESSMENT STRATEGY, P. 66
- ONE-TIME PASSWORDS WITH YUBIKEY, P. 47
- IMPLEMENT SECURITY CHECKS WITH PAM, P. 60
- REVIEWED: BEHRINGER BCF2000, P. 42

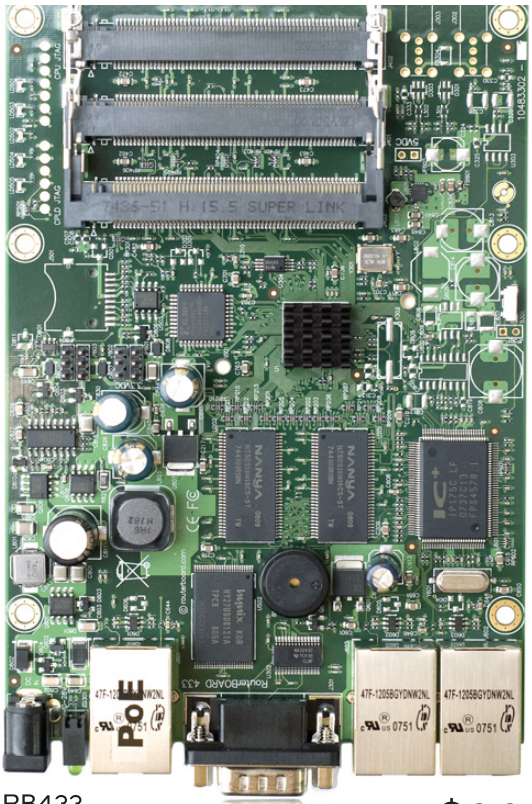
RouterBOARD 433

680MHz MIPS (overclock to 800MHz)



RB433AH $149

300Mhz



RB433 $99

The RB433 is a high speed AP/router. It has a new generation 300Mhz Atheros CPU and 64MB of RAM. It is provided with three 10/100Mbit Ethernet ports and three miniPCI slots.

The RB433AH is a faster, enhanced version of the RB433 with a 680Mhz CPU, 128MB RAM and a microSD card slot for an additional memory card. It supports overclocking to 800Mhz, pushing the price/performance ratio to new standards.

These devices are preinstalled with the MikroTik RouterOS Firewall/QoS/Routing operating system. RB433 comes with a L4 license, and RB433AH with L5

www.routerboard.com

CONTENTS JANUARY 2009 Issue 177

COLUMNS

8 SHAWN POWERS' CURRENT_ISSUE.TAR.GZ
No Room for Smugness (Well, Maybe a Little)

18 REUVEN M. LERNER'S AT THE FORGE
Memcached Integration in Rails

22 MARCEL GAGNÉ'S COOKING WITH LINUX
Evil Agents under the Bed and Other Scary Things that Go Boom!

26 DAVE TAYLOR'S WORK THE SHELL
Special Variables I: the Basics

28 MICK BAUER'S PARANOID PENGUIN
Samba Security, Part III

34 KYLE RANKIN'S HACK AND /
Manage Multiple Servers Efficiently

94 KYLE RANKIN AND BILL CHILDERS' POINT/COUNTERPOINT
Small Laptops vs. Large Laptops

96 DOC SEARLS' EOF
The Power of Definitions

IN EVERY ISSUE

10 LETTERS
14 UPFRONT
36 NEW PRODUCTS
38 NEW PROJECTS
81 ADVERTISERS INDEX

INDEPTH

72 MINORFS
A set of user-space filesystems for enhanced discretionary access control.
Rob Meijer

78 DETECTING BOTNETS
Using Darknet to secure environments from threats in the wild.
Grzegorz Landecki

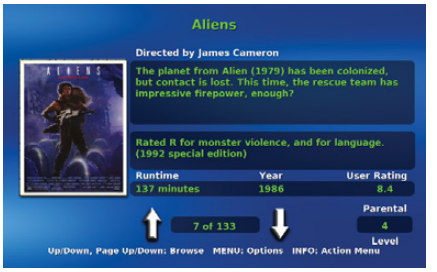
84 MYTHVIDEO: MANAGING YOUR VIDEOS
Too many videos in your MythTV menu? With a little planning, finding your favorite movies can be a breeze
Michael J. Hammel

90 USING CAPISTRANO
Simplify application deployment.
Dan Frost

REVIEW

42 MIXING IT UP WITH THE BEHRINGER BCF2000
Dan Sawyer

78 DETECTING BOTNETS



84 MYTHVIDEO

Next Month

WEB DEVELOPMENT

Web development isn't just for Spiderman anymore. Next month, we look at ways to improve the already venerable Ruby on Rails. That's not where we stop though; we have Django, Pylons and TurboGears for Python as well. If you still want more, the Google Web Toolkit might tickle your fancy, or one of a bunch of other Web development articles is bound to get your spidey sense tingling. Whether you're a new Web programmer or an old hand, you won't want to miss next month.

USPS *LINUX JOURNAL* (ISSN 1075-3583) (USPS 12854) is published monthly by Belltown Media, Inc., 2211 Norfolk, Ste 514, Houston, TX 77098 USA. Periodicals postage paid at Houston, Texas and at additional mailing offices. Cover price is $5.99 US. Subscription rate is $29.50/year in the United States, $39.50 in Canada and Mexico, $69.50 elsewhere. POSTMASTER: Please send address changes to *Linux Journal*, PO Box 16476, North Hollywood, CA 91615. Subscriptions start with the next issue. Canada Post: Publications Mail Agreement #41549519. Canada Returns to be sent to Bleuchip International, P.O. Box 25542, London, ON N6C 6B2

zt Systems

intel Xeon inside™ Powerful. Efficient.

ZT Systems delivers something different: a unique

B A L A N C E

of world-class server performance and cost advantage
joined with extensive flexibility in delivery and support.

ZT Systems 1102Ri 1U Rack Server

Affordable Single Socket Solution

Starting at Only **$999**

ZT1102Ri-82-C00001

- Single Quad-Core Intel® Xeon® processor X3320 (2.50G,6M,1333MHz)
- 2GB DDR2 800 Unbuffered SDRAM
- Intel® 3200 Server Chipset
- (2) 500GB SATAII 32MB Hotswap Hard Drives
- Supports RAID 0,1,5,10 JOBD
- (1) Intel® Gigabit Ethernet Connection
- 400W high-efficiency power supply
- Supports Intel® System Management Software 2.0
- 3 Year Limited Warranty and 24x7 Telephone support

ZT Systems 1203Ri 1U Rack Server

Dual-Socket Data Center Server

Starting at Only **$1399**

ZT1203Ri-84-C00001

- Single Quad-Core Intel® Xeon® processor E5405 (2.0G,12M,1333MHz)
- 4GB DDR2 ECC/REG. SDRAM (2x2GB, Dual Rank)
- Intel® 5100 Server Chipset
- (4) 500GB SATAII 32MB Hotswap Hard Drives
- Supports RAID 0/1/5/10
- (2) Intel® 82573 Gigabit Ethernet Ports
- 400W 80+ high-efficiency power supply
- Optional IPMI 2.0
- 3 Year Limited Warranty and 24x7 Telephone support

ZT Systems 1204Ri 1U Rack Server

Dual-Socket Server with 8 Hard Drives

Starting at Only **$1999**

ZT1204Ri-84-C00001

- (2) Quad-Core Intel® Xeon® processors X5410 (2.33G,12M,1333MHz)
- 8GB DDR2 ECC/REG. SDRAM (4x2GB, Dual Rank)
- Intel® 5100 Server Chipset
- (2) 320GB 2.5" SATA Hotswap Hard Drives
- Supports up to 8 (2.5") SATA/SAS Drives & RAID 0,1,5,10*
- (2) Intel® 82573 Gigabit Ethernet Ports
- 400W 80+ high-efficiency power supply
- Optional IPMI 2.0
- 3 Year Limited Warranty and 24x7 Telephone support

ZT Systems 4201Ci 4U Convertible Server

Convertible Tower/Rack SMB Solution

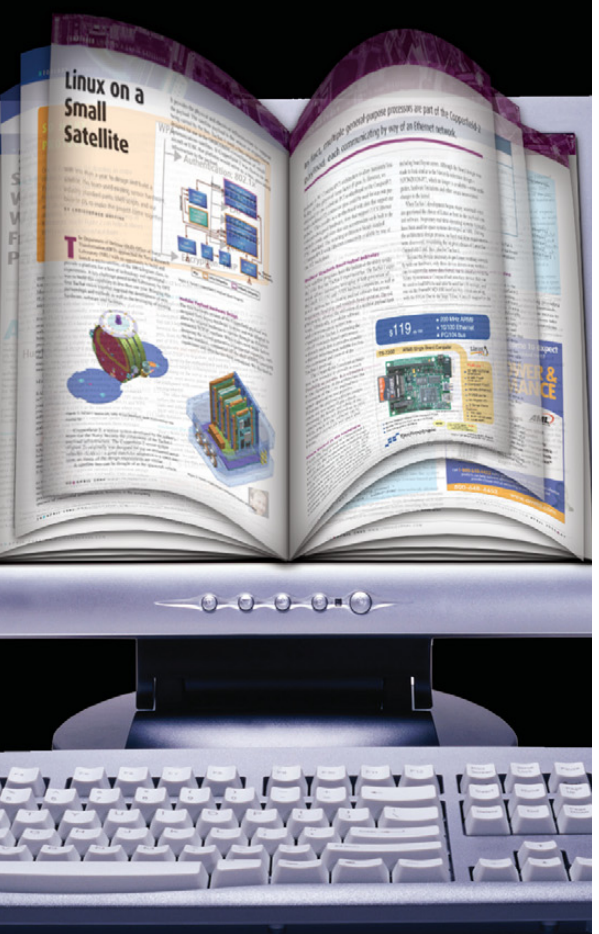
Starting at Only **$1999**

ZT4201Ri-82-C00001

- (2) Quad-Core Intel® Xeon® processors X5410 (2.33G,12M,1333MHz)
- 8GB DDR2 ECC/REG. SDRAM (4x2GB, Dual Rank)
- Intel® 5100 Server Chipset
- (2) 1TB SATAII 32MB Hotswap Hard Drives
- Supports up to 8 Hard Drives & RAID 0,1,5,10*
- 16x DVD-ROM & 1.44MB Floppy Drive
- (2) Intel® 82573 Gigabit Ethernet Ports
- 600W high-efficiency power supply
- Optional IPMI 2.0
- 3 Year Limited Warranty and 24x7 Telephone support

Scalable Custom Server Solutions - Contact Us to Learn More

(866) 984-7687 corpsales@ztsystems.com



LINUX JOURNAL

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
PHONE: +1 818-487-2089
FAX: +1 818-487-4550
TOLL-FREE: 1-888-66-LINUX
MAIL: PO Box 16476, North Hollywood, CA 91615-9911 USA
Please allow 4–6 weeks for processing address changes and orders
PRINTED IN USA

The Straight Talk People℠
SINCE 1991

ABERDEEN

HOW MUCH NAS DO YOU NEED?

AberNAS Network Attached Storage appliances feature:

Capacity
- From 1TB to 50TB in a single appliance

Scalability
- Start with just a few drives, add as your needs grow

Expandability
- Easily add storage to well beyond 400TB via XDAS and JBOD units

Functionality
- Integrated iSCSI, optional iSCSI box-to-box mirroring

Reliability
- Redundant power supplies, mirrored OS drives, RAID 6, hot-swap drives and recovery DVD

Diversity
- Available in Windows or Linux-based OS

Flexibility
- Easily integrated into a Linux, Mac, Windows or Unix environment

Quality
- Critically acclaimed award-winning servers and storage appliances

Affordability
- Best TB/$ ratio in the industry

Perpetuity
- Industry leading 5-Year warranty

intel Xeon® inside™
Powerful. Efficient.

1U ABERNAS

Up to 4TB NAS
- Dual-Core Intel® Xeon® Processor
- From 1TB to 4TB
- 2GB DDR2 Memory
- Dual Gigabit NIC
- 300W Power Supply
- 5-Year Warranty

Starting at $2,495

2U ABERNAS

Up to 12TB NAS
- Dual-Core Intel Xeon Processor
- From 2TB to 12TB
- 2GB DDR2 Memory
- Dual Gigabit NIC
- 500W Power Supply
- 5-Year Warranty

Starting at $3,995

3U ABERNAS

Up to 16TB NAS
- Dual Quad-Core Intel Xeon Processors
- From 8TB to 16TB
- 2GB DDR2 Memory
- Quad Gigabit NIC
- 650W Redundant Power
- 5-Year Warranty

Starting at $7,495

5U ABERNAS

Up to 24TB NAS
- Dual Quad-Core Intel Xeon Processors
- From 12TB to 24TB
- 2GB DDR2 Memory
- Quad Gigabit NIC
- 950W Redundant Power
- 5-Year Warranty

Starting at $9,995

6U ABERNAS

Up to 32TB NAS
- Dual Quad-Core Intel Xeon Processors
- From 16TB to 32TB
- 2GB DDR2 Memory
- Quad Gigabit NIC
- 1350W Redundant Power
- 5-Year Warranty

Starting at $12,495

8U ABERNAS

Up to 50TB NAS
- Dual Quad-Core Intel Xeon Processors
- From 20TB to 50TB
- 2GB DDR2 Memory
- Quad Gigabit NIC
- 1500W Redundant Power
- 5-Year Warranty

Starting at $14,495

888-297-7409
www.aberdeeninc.com/lj028

SHAWN POWERS

No Room for Smugness (Well, Maybe a Little)

I remember July 19, 2001, fairly well. Yes, it was my birthday, but more profound than that was the Code Red Internet worm (**en.wikipedia.org/wiki/Code_Red_worm)** that was at its peak infection point. Because I was the network administrator for a school district, the summer was spent upgrading and reinstalling servers to prepare for the next year. The Code Red onslaught was a great reminder that I needed to patch the few Windows servers I administered. Unfortunately, my main Windows machine already was infected, and at that point, we weren't entirely sure how much hidden damage was done to the machines. Because it was summer, I decided formatting the hard drive and starting over would be the easiest way to be sure my server wasn't infected. Because it was summer, the downtime wouldn't really be a problem, and reformatting Windows computers tends to make them work a bit better anyway. So that's what I did.

The problem was that before I even could download the security patch, my Windows server would become infected. I tried the "race" a handful of times, but in the end, I had to put my Windows server behind a Linux firewall/proxy machine that would protect it while it updated. I won't lie; using Linux to protect my Windows server during the upgrade did make me a little smug. I even bragged to my fellow school technology directors (most of whom run Microsoft shops) about how impervious Linux is to attack.

Then, in September, the Nimda worm (**en.wikipedia.org/wiki/Nimda**) crippled my Linux Web server.

Granted, my server didn't get infected with the worm, because like Code Red, Nimda targeted Microsoft's IIS server. The sheer number of concurrent infection attempts, however, effectively caused my poor little Web server to stop responding. It was then that I really began to realize how security is an active process, not just the result of smart planning. We don't all need to be security experts, but if we're in charge of any computers, we need to be aware of the tactics and tools available to protect them. Here at the *Linux Journal* office, we decided the perfect way to start the new year would be with an issue devoted to security.

One of the first obstacles to securing your infrastructure effectively can be the sheer size of it. It's true that command-line administration is quick and easy, but if you have hundreds or thousands of servers, even the command line can be overwhelming. Kyle Rankin shows us a few shortcuts he uses to connect to multiple servers via SSH.

Our own local security expert, Mick Bauer, continues his series on securing Samba. Mick shows us that the best offense is a good defense, and starting with a secure configuration is the key to sysadmin bliss. Jeramiah Bowling broadens the scope and details how to test our entire system's security. If you don't test your security for vulnerabilities, you can be sure someone else will.

If you want to get real serious about catching the bad guys, be sure to read Grzegorz Landecki's article on detecting botnets. They tend to be scary, because a large enough botnet can take down even a secure server. Early detection is key—well, that and a geographically diverse network infrastructure. For most of us though, early detection is about the best we can do.

Speaking of bad guys, this issue will make you happy to know that Kyle Rankin hasn't chosen the Dark Side of the Force. This month, he also explains how to attack computers that aren't even powered up. Did you think powering off a computer cleared the RAM? I did, but Kyle gives us a whole new reason to stay up at night worrying. His article is a tutorial on how to exploit the few seconds it takes for RAM to "forget" its contents. I'm sure the article is intended to teach us how to best secure ourselves from malicious attempts to do the same, but it's truly scary how simple the process can be.

This issue of *Linux Journal* is bound to appeal to everyone on some level. Whether you need to learn about secure authentication with PAM, or you just want to learn about new products, get a few tech tips and catch up on our latest programming column, you'll want to secure this issue under lock and key. Otherwise, someone like Kyle might sneak in and take it.■

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.

servers DIRECT®

MORE PRODUCTS, BETTER SERVICE, GUARANTEED.

GO STRAIGHT TO THE SOURCE!

1.877.727.7887 | www.ServersDirect.com

YOUR HIGH PERFORMANCE COMPUTING HAS ARRIVED.

The ServersDirect® Systems with the Intel® Xeon® Processor helps you simplify computing operations, accelerate performance and accomplish more in less time

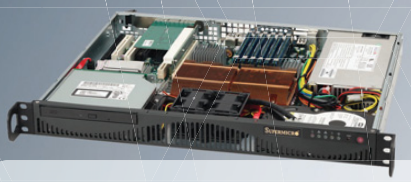
INTEL® MODULAR SERVER

The Intel Modular Server is an integrated system built on Intel Multi-Flex Technology that includes SAN storage, computing and networking to simplify the growing demands of your IT infrastructure.

The Intel Modular Server features:

- Up to six server Compute Modules
- Integrated SAN storage
- Integrated Ethernet Switch Modules
- All hot-swappable components
- Virtual Presence remote management capabilities

For more information visit: http://www.ServersDirect.com/IMS.html



SDR-S1208-T00

STARTING AT $559

- Supermicro Mini 1U Rackmount Server with 260W Power Supply
- Supermicro Server Board w/Intel® 946GZ Chipset
- Support up to a Dual-Core Intel® Xeon® 3000 Series processor
- TPM Support
- 1x 3.5" Internal Drive Bay
- 2x Intel® 82573 PCI-e Gigabit LAN Port

SDR-C1303-T02

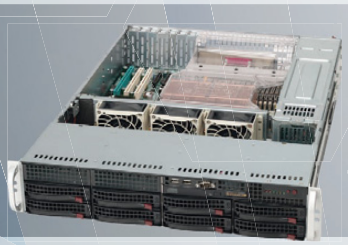
STARTING AT $899

- 1U Rackmount Server with 400W Power Supply
- Supermicro Server Board w/ Intel® 5100 Chipset
- Support Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core
- Support up to 48GB 667/533MHZ DDR2 ECC Reg
- 2x 3.5" Hot-swap SATA Drive Bay
- Intel® 82573V & 82573L Gigabit Ethernet Controller

SDR-C2301-T06

STARTING AT $1,059

- 2U Rackmount Server with 460W Power Supply
- Supermicro Server Board w/ Intel® 5100 Chipset
- Support Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core
- Support up to 48GB 667/533MHZ DDR2 ECC Reg
- 6x 3.5" Hot-swap SATA Drive Bay
- Intel® 82573V & 82573L Gigabit Ethernet Controller



SDR-S2301-T08

STARTING AT $1,499

- 2U Rackmount Server w/700W High-Efficiency Redundant Power Supply
- Supermicro Server Board w/Intel® 5000P chipset
- Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core
- Support up to 64GB DDR2 667 & 533 FB-DIMM
- 8 x 3.5" Hot-swap Drives Trays
- Dual-port Gigabit Ethernet Controller

SDR-C3301-T16

STARTING AT $1,399

- 3U Rackmount Server with 650W Power Supply
- Supermicro Server Board w/ Intel® 5100 Chipset
- Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core
- Support up to 48GB 667/533MHZ DDR2 ECC Reg
- 16 x 3.5" Hot-swap SATA Drives Trays
- Dual-port Gigabit Ethernet Controller

SDR-C4302-T02

STARTING AT $899

- 4U Rackmount Server with 600W Power Supply
- Supermicro Server Board w/ Intel® 5100 Chipset
- Support Dual Intel® 64-bit Xeon® Quad-Core or Dual-Core
- Support up to 48GB 667/533MHZ DDR2 ECC Reg
- 3 x 3.5"Internal SATA Drives Trays
- Dual-port Gigabit Ethernet Controller

SERVERS DIRECT CAN HELP YOU CONFIGURE YOUR NEXT HIGH PERFORMANCE SERVER SYSTEM - CALL US TODAY!

Our flexible on-line products configurator allows you to source a custom solution, or call and our product experts are standing by to help you assemble systems that require a little extra. Servers Direct - your direct source for scalable, cost effective server solutions.

1.877.727.7887 | www.ServersDirect.com

intel Xeon® inside™

Dual-core. Do more.

letters

New Subscriber Love

I just got my first issue of *Linux Journal*, and I must say I'm floored. In fact, I suddenly caught myself getting nostalgic, because there I was, reading code in a computer magazine—I haven't done that since the eighties! It gave me a great idea though. What if there was a regular column that looked just at programming techniques? For inspiration, look no further than columns written by the legendary Commodore guru, Jim Butterfield. Or, how cool would it be to feature complete program listings the readers could type in or download, just like the days of *COMPUTE!* magazine? Only now, of course, instead of being written in Apple or Commodore Basic, it could focus on Python and Pygame, or C++ and Gtkmm. Perhaps some well-known open-source developers would even enjoy stepping through parts of their code they are particularly proud of, and explaining how it works.

I certainly enjoy the features in the magazine focusing on the enterprise side of the Linux world, but I'd also love to see a celebration of the sheer joy of coding.

Anyway, thanks for a great magazine! My only dilemma now is whether to read *LJ* or *Tape Op* first.

--
Sean Corbett

Thanks for the feedback Sean, and stay tuned—you'll see the things you mention in upcoming issues.—Ed.

Simplicity

In his August 2008 column, Dave Taylor uses the following line:

```
pickline="$(expr $(( $RANDOM % 250 )) + 1 )"
```

Although that code is not wrong, I prefer this simpler line:

```
pickline=$(($RANDOM % 250 + 1))
```

--
Antoine

Dave Taylor replies: *Nice! Duly noted.*

Can't Please Everyone

I was noticing that *LJ* has been doing more software articles than in the past and that was the reason I renewed this last month. When I received the programming language issue [October 2008] I thought, "Yes! Finally an issue about languages." I even thought, "I'm going to write them to say thanks." And, then I noticed someone had written in requesting more hardware articles. I guess it's hard to please us all, eh? Keep it up (but please don't forget about the languages!).

--
Louis Juska

Compression Algorithms

The Tech Tip on page 72 in the November 2008 *LJ* uses tar and netcat to copy a directory tree between systems, but the specific command options are often painfully slow on a LAN. The bottleneck is that the gzip compression chosen (tar -z) executes slowly.

It is preferable to choose the compression algorithm according to the network and processor speed. Selecting faster but less efficient algorithms, like lzop, can speed up the transfer for fast connections, while slow but effective compression, like lzma, is preferred for very slow networks.

As a test, I used this Tech Tip with various compression options to transfer 4.6GB from an old server (2.6GHz P4-HT) able to read the ext3 files at about 30Mb/s with a gigabit network able to tcp at about 85Mb/s.

The commands used are:

```
[server] tar $TAR_OPT -cpsf - $dir | pv -b | nc -l 3333
[client] nc server 3333 | pv -b | tar $TAR_OPT -xpsf -
```

Results using these options:

```
TAR_OPT="-z"
TAR=OPT="--use-compress-program=lzop"
TAR=OPT=""
```

are, respectively:

```
gzip     time 679sec, rate 6.38  MBPS
lzop     time 357sec, rate 12.15 MBPS
(none)   time 160sec, rate 27.15 MBPS
```

Here, the network is faster than filesystem I/O, so any compression slows the transfer. For these systems, I calculate that lzop would be helpful below a 62Mb/s network speed and gzip below 4Mb/s. These breakpoints would increase if the computers could compress and decompress faster.

I couldn't bring myself to test lzma, as it is many times slower than gzip, but it may be useful for dial-up transfer.

For a fine comparison of compression algorithms, see the September 2005 *LJ* article by Kingsley G. Morse Jr. at **www.linuxjournal.com/article/8051**.

--
Steve Alexander

It's Not a Vendor Thing

Mr. Bonny's letter ["It's a Vendor Thing", *LJ*, November 2008] raises the hackles of us Linux enthusiasts. Still, he raises important issues.

Despite claims to the contrary, Linux driver support is on par with Windows and is radically superior to OS X. However, most new users are used to buying a computer with an OS pre-installed and configured and trivially installing

vendor-supplied drivers for any widgets they add.

Installing Linux is vastly improved today, and in most instances, it is far easier than installing Windows. But, very rarely do people install Windows themselves anymore. Installing third-party hardware is substantially more challenging.

Googling "3 mobile broadband linux" seems to suggest that there is Linux support, and I would be shocked if there was not Linux support for Mr Bonny's 56K modem. This does not mean getting hardware working that does not have out-of-the-box support from your Linux distribution is inside the skill set of ordinary users.

No OS is perfect. I run Linux on my PowerBook because the internal NIC failed, and I could not find a supported add-on card. I regularly inherit often fairly new "broken" Windows laptops. Virus infections, spyware, conflicting software installs and flaky hardware drivers have resulted in slow and unstable operation. In all instances, a clean re-install restores them to like-new operations. In extremely rare instances, Linux systems suffer the same problems. And in most cases, the problems can be cleaned up, but few Windows machines go 18 months without requiring a clean re-install.

Unfortunately, Mr Bonny and many other users need the skills of a Linux guru and extraordinary vendor support to configure Linux for their needs. But, the payoff is a system that will be more robust. Further, a few months of using Linux regularly inevitably will result in developing a dependence on features that do not exist elsewhere.

Viruses, spyware, corrupted registries, flaky drivers and dll conflicts are of no interest to most Windows users who typically solve those problems by buying new systems.

--

Dave Lynch

Correction

On page 51 of the November 2008 issue, Daniel Bartholomew writes that he mapped the IP address of his Popcorn device using his /etc/resolv.conf file. I'm guessing that he meant using his local /etc/hosts file to map the name to the IP?

--

Jonathan Miner

Daniel Bartholomew replies: *You are correct. This looks like a case of my mind thinking one thing and my fingers typing something completely different. Thanks for catching it!*

Thanks for the HPC Articles

As a number-crunching scientist who has used Linux daily since 1994, let me thank you for two excellent articles in the November 2008 issue: Michael Wolfe's article on GPGPUs and Joey Bernard's article on Python for scientific computing. There is more to Linux than Web 2.0.

That said, I have a minor quibble with Joey Bernard's matrix multiplication example using numpy. By default, numpy objects are arrays, not matrices. So a1*a2 in his example is an element-by-element array multiplication, not a matrix multiplication. To get the result he intended, Joey either should have created explicit matrix objects or used a3 = numpy.dot(a1,a2) or a3 = mat(a1)*mat(a2).

That minor criticism aside, can we have more articles like Joey's and Michael's please!

--

Dave Strickland

Array Multiplication

Joey Bernard's article "Use Python for Scientific Computing", *LJ*, November 2008, is a valuable introduction, and it prompted me to compare Python versus my own language, experix. The most important feature of experix that (as far as I know) is not found elsewhere is the detailed exposure of the kernel device driver interface to user command input. In my lab at Washington University, we are using experix to perform device control and data acquisition on instruments with piezoelectric and stepper motors; to analyze and archive the data; to perform analytic and Monte-Carlo simulations of fluorescence intensity distributions; and to fit photon count records from a Zeiss ConfoCor system to particle distribution models.

I find Bernard's exe times for array multiplication highly questionable. The time for unoptimized C is close to what I get on my Pentium laptop, but the other times (for -O3 and Python) are preposterous unless it was done with massive parallel processing.

Here is a very contrived experix example, demonstrating most of what Bernard did with Python plus some other things, and written in a way that fits in a 40-character column for printing. For info and downloads, see **experix.sourceforge.net** and **sourceforge.net/projects/experix**:

```
;; load some graphics stuff
&~/experix/dist/xpx/graftrix
;; make a [479,503] ramp array and
;; convert to Poisson deviate
.001 479 503 2 ] ]+ ]P
;; make a [503,512] array filled
;; with sin((.00005*j+10)^2)
5e-5 503 512 2 ] ]+ 10 + .sq .sin
;; multiply these and make a scaled
;; graph of the [479,512] product
]m \2k \2k Fgsa \s Igsa \s graph/skW
;; Fourier transform; graph column 1
fft> 1 -1 [s \s\-4r graph/sTzRl \3D
;; create a file called "demo"
''of def/be ''xw of "demo" file/o
;; define a format string
"w  DC: %g  1Hz: %g  hiF: %g %g %g"
''fm1 def/r
;; make a command to write 5 numbers
;; from an array to file, formatted
{ of "w %d" file/w 512 * 5 [r }
{ of fm1 file/wn \d } | ''L1 def/rc
;; do each array column; close file
$0: ,0r L1 ,0i 479 ,0c!=$0 of file/c
```

--

Bill McConnaughey

Democratic Utopia?

In the November 2008 issue, Doc Searls writes about how technology can finally bring us to some democratic utopia. I think that nothing could be further from the truth. I believe de Tocqueville coined the phrase "tyranny of the majority" to describe the

_LJ_ pays $100 for tech tips we publish. Send your tip and contact information to techtips@linuxjournal.com.

almost certain results.

For evidence, just look at current events. Huge numbers of folks (very likely a majority) have no problem with a presidential candidate who announces his plan on the first day in office to shut down opponents on talk radio. No problem at all. "The People", as it were, are too easily swayed and too easily deceived.

As a member of a number of minorities, such as "bicycle commuters", "private pilots", "skiers", "EEs", "tax-payers", "non-smokers who think smokers should be able to smoke" and numerous others, I'm painfully aware that I'm always at the mercy of the majority as it is. The idea that at any moment, some democratic goodwill impulse will cut out another little freedom is all too real. When democracy starts to turn into populism and nationalism, history has shown that things always turn ugly.

I bet that a large number of readers, if not a majority, already view the phrase "tax the rich" with joyous enthusiasm. It gives me a cold chill. To me, the rich are entitled to their riches. I'd like to join them some day. The idea that they are some minority that we should milk for our benefit is an assault on liberty. It means that we no longer have the thirst for equality and justice that once wrote our Constitution.

One can ask what the solution is. I would say a little less democracy and a lot more education—the kind that is no longer taught in our public schools. A little more Adam Smith, and a lot less Karl Marx. Uneducated people historically vote themselves into a kind of servitude.

I do agree that more openness in government is a good thing. Politicians all too often hide behind layers of legalese and obfuscation. But Whitman's ode to democracy is downright scary. Politics 24/7? Every interaction governed by the masses? Please, no. Just keep every bill to a page or two of actual English.

I really don't want to be involved in every nit that needs to be picked, and I really don't want the government to be picking nits anyway. What I want government to worry about are the big things that folks can't do individually. Things that people wiser than myself can handle. Take care of it and don't bother me is my utopia. I'll take a little more wisdom and liberty, and a lot less democracy, anytime.

--
Gene

Brilliant New Slogan

Microsoft has recently launched a new ad campaign that uses the slogan, "Life without walls". I find that interesting. You know what happens if you don't have any walls? Windows crash.

--
Alexander Pennington

PHOTO OF THE MONTH

Have a photo you'd like to share with *LJ* readers? Send your submission to publisher@linuxjournal.com. If we run yours in the magazine, we'll send you a free T-shirt.

Penguins at Kite Fair on Southsea Common, Portsmouth, UK. Photo taken by Simon Wright.

diff -u

WHAT'S NEW IN KERNEL DEVELOPMENT

Tejun Heo has expanded **FUSE** (Filesystem in USErspace) to allow creating character devices as well as filesystems. He calls the new branch of code **CUSE** (Character device in USErspace). Tejun's first example application to use CUSE, however, might have been better chosen. His sound card wasn't working so well with the ALSA drivers, so he implemented an OSS proxy character device using CUSE. It worked for him, which at least demonstrated the usefulness of CUSE itself, but as **Adrian Bunk** pointed out, a better approach for that specific case might have been to fix the ALSA drivers instead of emulating OSS. On the other hand, as Tejun said, even his CUSE-based OSS implementation would let people run old binaries that hadn't been ported to ALSA and compile old source trees that were no longer maintained.

Jonathan Corbet has announced the election of several new members to the **Linux Foundation Technical Advisory Board** (TAB). **Kristen Carlson Accardi**, **James Bottomley**, **Dave Jones**, **Chris Mason** and **Chris Wright** will each serve for two years, and **Christoph Hellwig** will serve for one year. Christoph replaces **Olaf Kirch**, who resigned recently. The vote actually was split between Christoph and **Theodore T'so**, so the folks decided by a coin toss.

BtrFS seems to have been selected as the filesystem of the future by a number of influential kernel folks, including Theodore T'so. This was partially the result of back-room discussions about the need for a "next-generation" filesystem for Linux, and about which of the available options it might be. BtrFS, thus, has gained the focused attention of a wide-ranging group of developers and companies (including **HP**, **Oracle**, **IBM**, **Intel** and **Red Hat**), and we can expect its development to proceed along carefully considered lines. We also can expect BtrFS to be accepted into the main-line kernel tree fairly quickly, even though it hasn't yet stabilized, as part of an effort to recruit a wider body of users and contributors. **Andrew Morton** supports this plan, and **Linus Torvalds**' new policy of favoring early merges in general seems to support it as well. However, folks like Adrian Bunk caution that the code may not be ready yet, and that merging it into the main tree may not get the users and developers that folks expect.

David Vrabel has created a git repository for the **Ultra-Wideband (UWB) radio**, **Certified Wireless USB** (WUSB) and **WiMedia LLC Protocol** (WLP) subsystems that he maintains, and he made some motions to get the code accepted into the main kernel tree. At the time he did this, it wasn't 100% clear whether he was submitting the code right then or looking for final feedback before submission. But one way or the other, it does seem as though the code will be going into the kernel soon.

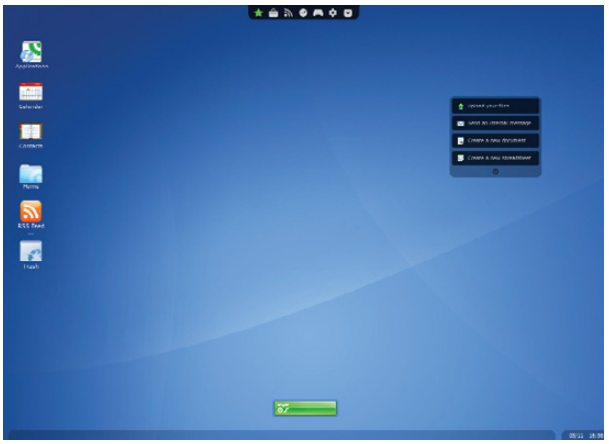
In a step along the road to running multiple operating systems on a single machine at the same time, **Yu Zhao** has written code to allow those various OSes to share the same PCI device during concurrent operation. This **single-root I/O virtualization** (SR-IOV) is part of a general trend of allowing very different operating systems to coexist productively, almost as different subsystems of an overarching OS, that may in time come to communicate with each other and rely on each other in more and more integrated ways. —ZACK BROWN

LJ Index, January 2009

1. Number of finds in a search among Twitterers for "linux": 1,540
2. Number of OLPC followers on Twitter (which runs on Linux): 969
3. Percentage of surveyed students who said college would be much harder without Wi-Fi: 79
4. Percentage of surveyed students who said they wouldn't attend a college without Wi-Fi: 60
5. Percentage of surveyed students who have checked Facebook or MySpace and sent or received e-mail while in class: 50
6. Percentage of projected Wi-Fi penetration at universities by 2013: 99
7. Number of acres in the University of Minnesota's 802.11n deployment: 1,200
8. Percentage running Linux or BSD among Netcraft's most reliable hosting companies for August 2008: 50
9. Position of Linux-based Hurricane Electric among Netcraft's most reliable hosting companies for August 2008: 1
10. Number of Linux-based companies among Netcraft's top 50 most reliable hosting companies for August 2008: 26
11. Percentage of Internet traffic growth between mid-2007 and mid-2008: 53
12. Percentage of Internet capacity utilized in the same period: 29
13. Percentage of Internet peak utilization in the same period: 43
14. Median wholesale $/Mb price in for a 1Gb IP transit port in New York in Q2 2008: 10
15. Median wholesale $/Mb price in for a 1Gb IP transit port in Hong Kong in Q2 2008: 37
16. Number of Ubuntu servers on which Wikipedia now runs: 400
17. Millions of visitors to Wikipedia per year: 684
18. Millions of articles in Wikipedia: 10
19. Thousands of active contributors to Wikipedia: 75
20. Number of languages used in Wikipedia: 250

Sources: 1–2: Twitter | 3–5: Wakefield Research, via *InformationWeek*
6: ABI Research, via *InformationWeek*
7: *InformationWeek* | 8–10: Netcraft
11–13: TeleGeography's Global Internet Geography | 14, 15: *ars technica*
16–20: *Computerworld*

eyeOS: Clouds for the Crowd



Cloud computing from the likes of Google and Amazon has become quite the rage in the last few years. Nick Carr's *The Big Switch* and other works have pointed toward a future of "utility" computing where we'll all use hosted apps and storage, thanks to the "scale" provided by big back-end companies and their giant hardware and software farms. But, there also has been pushback. Most notable among the nay-sayers is Richard M. Stallman, who calls it "worse than stupidity" and "a trap".

At issue is control. Of Web apps, RMS says, "It's just as bad as using a proprietary program. Do your own computing on your own computer with your copy of a freedom-respecting program. If you use a proprietary program or somebody else's Web server, you're defenseless. You're putty in the hands of whoever developed that software."

We wrote about it on-line at LinuxJournal.com, and among the many comments was one that pointed to eyeOS: a cloud computing approach by which people can make their own clouds: "...all you need is a Web server that supports PHP and OpenOffice.org to get the most out of the included office suite", the commenter said. "It's cloud computing, but at the same time you still have control over your data."

eyeOS is based in Barcelona, and obviously, it doesn't believe you need to be a Google or anyone special to run a "cloud" Web service environment. Unlike Google's cloud, you don't need to run the eyeOS's hosted apps. You can upload your own or choose ones from eyeOS or other developers. The UI is a virtual desktop, inside a browser (just as with Google), and the initial suite of apps are the straightforward set you'd expect, plus many more. These come with user ratings and a very active set of forums for developers and users.

eyeOS is a commercial company, privately held (and debt-free, it says). Its business model is service and support. If you need help installing eyeOS or adapting apps for your company, they're available.

- Stallman vs. Clouds: **www.linuxjournal.com/content/stallman-vs-clouds**
- eyeOS: **eyeos.com/en**
- eyeOS Blog: **blog.eyeos.org**

—DOC SEARLS

Find It at LinuxJournal.com

This month's issue of *Linux Journal* is all about security. At LinuxJournal.com, searching for the term "security" returns 435 results, which might take some time to wade through. Here are my picks from articles that recently have been popular on-line:

- "Add Web Porn Filtering and Other Content Filtering to Linux Desktops": **www.linuxjournal.com/article/9044**
- "The DNS Bug: Why You Should Care": **www.linuxjournal.com/content/dns-bug-why-you-should-care**
- "Understanding Kaminsky's DNS Bug": **www.linuxjournal.com/content/understanding-kaminskys-dns-bug**
- "Debian Security Flaw": **www.linuxjournal.com/content/debian-security-flaw**

You'll also want to check in with our on-line News Editor from time to time. Security is frequently a topic of discussion:

- "Security Is the Name of the Game": **www.linuxjournal.com/content/security-name-game**
- "With Linux, Even Rootkits Are Open Source": **www.linuxjournal.com/content/linux-even-rootkits-are-open-source**

Stay safe out there!

—KATHERINE DRUCKMAN

They Said It

"You cannot bundle abundance with scarcity; it's like trying to implement region coding of the air that you breathe. But then some people will try anything."

—JP Rangaswami, confusedofcalcutta.com/2007/07/08/prince-ly-returns-from-the-because-effect

"The market right now is just too good for individual developers who have experience in writing open-source software for Linux, especially the low-level plumbing of Linux, to waste their time working for companies who do not allow them to contribute back, if they want to."

—Greg Kroah-Hartman, www.kroah.com/log/linux/lpc_2008_law_and_gospel.html

"When you tell me I should give proprietary software a fair technical evaluation because its features are so nice, what you are actually doing is saying "Look at the shine on those manacles!" to someone who remembers feeling like a slave."

—Eric S. Raymond, esr.ibiblio.org/?p=556#more-556

"I worry about the idea of trying to centralize everything. The Washington tactic is, when there's a problem, you appoint a czar, and the czar is responsible. It's like the War on Drugs or the War on Poverty. But it never quite works; you don't get very good solutions."

—Vint Cerf, www.cioinsight.com/c/a/Expert-Voices/Vint-Cerf-Keeping-the-Internet-Healthy

"Always beware of wolves dressed as Grandma, they may be more like Microsoft than they admit."

—Bob Bickel, bobbickel.blogspot.com/2008/09/ringside-winding-down.html

What They're Using

Tom Limoncelli

I first met Tom Limoncelli on a cold January day in Burlington, Vermont, where he was a volunteer geek at the Howard Dean campaign headquarters. I was extremely impressed not only by his technical know-how, but by his real-world wisdom about where technology and humanity intersect.

At the time, Tom was coming out with his first book, *The Practice of System and Network Administration*, cowritten with Christine Hogan. Since then, he also has written *Time Management for System Administrators* for O'Reilly.

These days, Tom works as a System Administration Manager for Google in New York. Although he wrangles many platforms, he remains a devoted Linux user and advocate. Here's how he runs down what he's using right now:

> The bumper sticker on my car reads, "My other computer is a massive Linux cluster!" It's true. At Google, we use massive clusters of Linux boxes for our Web services and nearly everything else too. (The actual number of computers is a company secret.) Once I used MapReduce (Google's parallel scheduling system) just to copy a database (each machine copied less than 1% of the total rows). In our remote offices, we deploy small Xen clusters and manage them with Ganeti (a package we recently open-sourced). The Xen clusters run Ubuntu, as does my desktop and one of my laptops. My phone runs Android, which is also Linux.
>
> Since all my data is on servers, I can do all my work with an SSH client and a Web browser. My documents are all in Web-based office applications, and thanks to "Gears", they work whether or not I'm connected to the network. My preferred SSH client is OpenSSH with an old-school xterm, but Mac OS X's Terminal app is winning me over. My favorite Web browser is Chrome, but I use Firefox as a close second. When I use Windows, I immediately install Cygwin's OpenSSH and rxvt to reduce the pain.
>
> I cowrote my first book using vim, CVS, make and teTeX. My next book was written using vim and Subversion. Now I'm moving everything to Git. Even for solo projects, I can't live without a source code repository on a safe, backup'd, server.
>
> I couldn't live without screen, rsync, wget and curl. I think more system administrators should use make to maintain servers as I described in *TM4SA*. I program in Python at work, Perl at home, and awk so much it makes younger sysadmins cry. I also love cat, tee, sed, grep, bc, mount, man, date, cal, ftp and ping...but doesn't everyone?
>
> When people ask me, "When will Linux be usable by a typical grandmother?", I reply, "She uses Linux every time she uses Google! So there!"

You can keep up with Tom at his blog, **EverythingSysadmin.com**.

—DOC SEARLS

Tribler: BitTorrent and Beyond

P2P (peer-to-peer) is the nature of the Net. You can fight that, or you can embrace it. Here in the US, the mainstream entertainment business has mostly been fighting it. Hollywood and its phone and cable company allies have long regarded P2P, and BitTorrent in particular, as a copyright piracy system and a bandwidth hog. In the European Union, however, P2P is more than accepted: it's supported by the Union itself.

Early last year, the EU granted 14 million euros to P2P-Next, a consortium of 21 media companies and universities, including the BBC, Delft University of Technology, the European Broadcasting Union, Lancaster University, Markenfilm, Pioneer Digital Design Centre Limited and VTT Technical Research Centre of Finland. The purpose of the grant is "to develop a Europe-wide 'next-generation' Internet television distribution system, based on P2P and social interaction". (An additional 5 million euros is also being donated by some of the P2P-Next partners, for a total of 19 million euros.) The project has a four-year span and will include technical trials of new media applications on many devices.

"Everything we're doing is based on open source", says Johan Pouwelse, PhD, scientific director of P2P-Next and Assistant Professor of Computer Science at Delft. The good doctor also runs P2P-Next's first trial application: Tribler (pronounced "tribe-ler"), a BitTorrent-based client with no servers and a "zero-cost" business model. Tribler provides an all-in-one way to find, consume and share media.

But Tribler goes beyond BitTorrent to support live streaming and other enhancements. The project's Research page lists 26 allied development projects, including six that are already completed and operational. If you're looking to help media evolve past the TV model, there's a rich pile of possibilities on the Tribler project list.

The Tribler download page lists two Linux sources: Ubuntu Linux and "GNU+Linux/Source".

Check it out, and let us know how it works for you (or, you for it).

- 19 Million Euro for P2P Research: **www.tribler.org/P2P-Next/19Million-for-P2P**
- P2P Next: **www.p2p-next.org/?page=content&id=264A360A217FB3FE8BD82CB9C928CBCF&mid=6BED2EAC3D127503EF53456A25D9204E**
- Tribler: **www.tribler.org**
- Tribler Research Page: **www.tribler.org/TriblerResearchSubjects**
- Tribler Download Page: **www.tribler.org/Download**

—DOC SEARLS

REUVEN M. LERNER

Memcached Integration in Rails

Integrating memcached into your Rails application is easy and fast, with big benefits.

Last month, we talked about memcached, a distributed caching system that is in widespread use among Web sites. The reason for memcached's popularity is its simplicity. With a minimum of overhead and setup, it's possible to set and retrieve nearly any value. Caching values that otherwise would come from the database makes it possible to avoid the database altogether on many occasions, speeding the throughput of a Web application and reducing the load on the database server.

Memcached is a wonderful tool, and it is something nearly every Web developer should have in his or her arsenal to improve site performance. But with the release of Ruby on Rails 2.1, it got even better. Rails now has integrated support for memcached, allowing you to use it almost for free from within your application. There are some caveats and tricks to its use, but once you have those under your belt, you quickly will discover that memcached has improved your site performance dramatically.

This month, we take a look at how to make memcached work inside your Rails applications. We further explore some issues you might encounter when using memcached, some of which are easier to work around than others.

Cache Integration

Ruby on Rails has, since its inception, tried to make Web developers' lives easier by coming out with many tools such developers might need. It comes with an excellent object-relational mapper (ORM), ActiveRecord. It comes with a way to test your code at a variety of different levels (called, in Rails-speak, unit, functional and integration). It comes with a first-class JavaScript library and associated effects, in Prototype and Scriptaculous. As numerous demonstrations and tutorials have shown, Rails allows you to jump right in to Web development, writing and testing your code with a minimum of dependencies. If you need to include some functionality that was left out by the Rails authors, it's not very difficult to include a Ruby gem (downloadable library) or even a "plugin" that sits inside your Rails application.

Rails has long come with a multilayered caching system that programmers can tap to speed up applications. You can cache individual pages, controller actions or even page fragments. And indeed, judicious use of the Rails caching commands can result in serious improvements to performance.

But, it was only in version 2.1 that Rails integrated support for caching individual objects. The support for object caching not only has the potential to improve your application's performance dramatically, but it also allows you to work with a variety of different storage facilities, so you can choose the one that's most appropriate for you. Although this article concentrates on the use of memcached, you should know that it's possible to work with not only memcached, but also with caches on the local filesystem, in local memory or even on another Rails-aware server using DRb (distributed Ruby, available as a Ruby gem).

Caching a Simple Object

To demonstrate how to use memcached, I'm going to create a simple Rails application, using PostgreSQL as the database:

```
createdb atf
rails --database=postgresql atf
```

Next, I create a simple object, person, for my application, with the Rails built-in scaffolding that includes a RESTful interface:

```
./script/generate scaffold person firstname:string
➥lastname:string email_address:string
```

To import this definition into the database, I run the migration that it created:

```
rake db:migrate
```

Sure enough, if I connect to the database, I can see that the table has been created (Listing 1).

And, if I run the application, I have access (via the RESTful interface) to the various CRUD functions associated with a Person object: Create, Retrieve, Update and Delete. I simply type:

```
./script/server
```

Listing 1. Example Table

```
atf_development=# \d people
                      Table "public.people"
 Column        |             Type               | Modifiers
---------------+--------------------------------+-----------------------------
id             | integer                        | not null default nextval
                                                   ➥('people_id_seq'::regclass)
firstname      | character varying(255)         |
lastname       | character varying(255)         |
email_address  | character varying(255)         |
created_at     | timestamp without time zone    |
updated_at     | timestamp without time zone    |
  Indexes:
      "people_pkey" PRIMARY KEY, btree (id)
```

And, I point my Web browser to port 3000 on my server: http://atf.lerner.co.il:3000/people/.

So far, so good. With a few commands on the UNIX command line, I've managed to create a simple database of people. I'll use the scaffolded application to add several people, clicking on the New person link and then adding the first name, last name and e-mail address of each of my friends.

Now, if I look at the Rails development log, I easily can see that each act I perform from within the scaffolded environment results in an SQL query being built and sent to the PostgreSQL server. I often do this by typing:

```
tail -f log/development.log
```

For example, if I click on the show link for the first person I created, I see the following in the development log:

```
Person Load (0.001571)   SELECT * FROM "people"
 ➥WHERE ("people"."id" = 1)
```

In other words, Rails knows that I want to load a Person object. It also knows that I retrieve such objects from the database. This is where ActiveRecord steps in, turning the Ruby:

```
Person.find(1)
```

into:

```
SELECT * FROM people WHERE people.id = 1
```

As you can imagine, it's not a big deal to do this sort of simple query, particularly if you have a limited number of fields, a small data set and a well-indexed primary key. But, as the number of fields increases, you might find yourself wanting to reduce the load on the database. Moreover, modern dynamic Web sites might need to retrieve 5–10 different objects from the database, only some of which are particular to the current user. If you get even 1,000 visitors to your site each day, and if there are three objects on each page that could be cached, that's 3,000 database queries you are foisting upon your database unnecessarily.

Memcached is an obvious solution to this problem. With previous versions of Rails, you needed to use a plugin or Ruby gem to do that. Now, however, you can do it via a configuration file. The gem that you previously needed to install, memcached-client, now is included along with the Rails gem. Every Rails application contains a main configuration file (config/environment.rb), which allows you to configure your application using Ruby code. This is where you should put configurations that are common to all three standard Rails environments: development, testing and production. For configurations that are specific to one environment, you instead would modify config/environments/ENV.rb, where ENV should be replaced with the environment of your choice.

Because we're still developing our example application, and using the development environment, we can confine our changes to config/environments/development.rb. Open that file in the editor of your choice, and add the following line:

```
config.cache_store = :mem_cache_store
```

This tells Rails that you want to use memcached and that the server is on the local computer (localhost), using the default port 11211. However, you can override these, and even put things into a separate namespace, if you're worried about stepping on someone else's objects.

When you're working in development mode, you also need to tell the server to use caching, a parameter that is set (and false) by default:

```
config.action_controller.perform_caching = true
```

Caching Objects

Now, let's go in and modify the GET action within the controller that was built for us by the scaffolding system. (The built-in caching is designed to be used from controllers and views, rather than from models.) That'll be:

```
app/controllers/people_controller.rb
```

On line 16 of that file, you'll see:

```
@person = Person.find(params[:id])
```

This is obviously where we invoke Person.find, as shown in the logs earlier. Now, modify that line so it looks like this:

```
@person = cache(['Person', params[:id]]) do
  Person.find(params[:id])
end
```

We still are assigning a value to @person. And, our call to Person.find is still in there. However,

If you get even 1,000 visitors to your site each day, and if there are three objects on each page that could be cached, that's 3,000 database queries you are foisting upon your database unnecessarily.

Person.find now is buried within a block. And, that block is attached to the call to a cache function, which is given an array argument.

What's happening here is actually fairly straightforward. The cache function looks in the cache for its argument, which is turned into a key. If a value for this key exists in the cache, the value is returned. If not, the block is executed, with the result of executing the block stored in the cache and returned to the caller.

With this code in place, let's retrieve person #1 again and look at the logfile. The first time we do this, the value is indeed retrieved from the database, as before:

```
Person Load (0.002212)   SELECT * FROM "people"
 ➥WHERE ("people"."id" = 1)
```

That line is followed by this new entry:

```
Cache write (will save 0.01852): controller/Person/1
```

Sure enough, our memcached server reports:

```
<7 new client connection
<7 get controller/Person/1
>7 END
<7 set controller/Person/1 0 0 224
>7 STORED
```

In other words, our Rails controller did exactly as we asked. It contacted memcached and asked for the value of controller/Person/1. (We can see from this that controller is prefaced to the key name that we create, and that elements of the cache key array are separated by slashes.) When we get a null value back for that, Rails retrieves the value from the database and then issues a set command in memcached, storing our value.

As you might expect, we then can refresh our browser window and see that we are saving a great deal of database time by retrieving information about this person from the cache. So, we refresh the browser window, and...boom! Our application blows up on us, with an error message that looks like this:

```
undefined class/module Person
```

Now, the first time this happened to me, I wasn't sure what hit me. What do you mean, I asked my computer, you don't know how to find a Person class? A little head-scratching and Google searching later, and I found my answer. I needed to tell the controller to load the object definition by putting the following at the top of my controller:

```
require_dependency 'person'
```

This is apparently necessary only in development mode, and it has something to do with the way Rails reloads classes while you are developing your application. With that line in place, you can reload the page. In the logfile, you'll see no trace of a successful call to the database. Instead, you'll find the following:

```
Cache hit: controller/Person/1 ({})
```

Meanwhile, our memcached log will look like this:

```
<7 get controller/Person/1
>7 sending key controller/Person/1
>7 END
```

This is a good time to mention the only other gotcha I can think of: whitespace is forbidden in memcached keys. This can be a problem if you use a value from the database (for example, a parameter name) as the key when storing things in

memcached. The simple solution is to remove the whitespace, either by running String#gsub on each of the keys or by monkey-patching String (as I did for an application I wrote) to add a to_key method. I could then pass `"parameter name".to_mkey` as an argument to cache().

Expiration

Now, it's all well and good that we have cached information about each person in memcached. Our database certainly will thank us for that. But, what happens when data about the person changes? The way we've written this application, we're out of luck. Updated information will make its way to the database, but the cache will continue to give us the data it stored long ago. Even if this weren't the case, we still would want to empty the cache on occasion, allowing data to expire if we haven't used it in a while.

To solve the second problem, we can invoke our cache function in a slightly different way, indicating how long we want it to stick around in a second (and optional) argument:

```
@person = cache(['Person', params[:id]],
                :expires_in => 30.minutes) do
  Person.find(params[:id])
end
```

The :expires_in parameter accepts a number of seconds, which we either can enter by hand or via one of the super-convenient Rails extensions to the Fixnum class.

The second problem, one of expiring data manually, requires that we use a less beautiful, but also convenient, way of accessing the cache storage system:

```
Rails.cache.delete(['controller', 'Person',
 ➥params[:id]].join('/'))
```

Basically, we access the cache system using the Rails.cache object and invoke the delete method on it. That method accepts a memcached key. As you might remember, we previously saw that the elements of our key array (as used by the helpful cache method) were joined by slashes and prefixed with controller. Thus, the above works, even though it's not quite as nice as I might have liked. We can see that this is the case in the memcached logs:

```
<7 delete controller/Person/1 0
>7 DELETED
```

And, sure enough, we then find that our next invocation of show for person 1 retrieves the information from the database and caches it in memcached.

Conclusion

Caching has long been an excellent way to improve performance in the computer industry, from the hardware level all the way up to operating systems and applications. Rails programmers have incorporated memcached into their applications over the last few years, but I believe that its complete integration in version 2.1 will make it even easier, and more widespread, to find memcached-enabled Rails applications. As you can see, adding just a few lines of configuration and application code can speed up an application by many times, without having to sacrifice accuracy.■

Reuven M. Lerner, a longtime Web/database developer and consultant, is a PhD candidate in learning sciences at Northwestern University, studying on-line learning communities. He recently returned (with his wife and three children) to their home in Modi'in, Israel, after four years in the Chicago area.

Resources

If you are looking for information on memcached, you should begin at **www.danga.com/memcached**, the home page for the open-source project and the source of a great deal of good documentation, code and general information.

For information on Ruby on Rails, start by going to **www.rubyonrails.com**, which has pointers to documentation, mailing lists and (of course) software you can download.

For information on the integration of memcached into Rails, try **www.thewebfellas.com/blog/2008/6/9/rails-2-1-now-with-better-integrated-caching**.

There are some Rails plugins that might make it even easier to cache objects. For example, take a look at **www.inwebwetrust.net/post/2008/09/08/query-memcached** and **lucaguidi.com/pages/cached_models**, both of which have gained some attention since Rails 2.1 caching was released.

Finally, a tutorial on the use of memcached with Rails is included in a chapter of *Advanced Rails Recipes*, published by the Pragmatic Programmers. I have greatly enjoyed this book and recommend it to anyone planning to use Rails for more than a simple application. The chapter on memcached is one that has been released as a free sample, and it is available in PDF as **media.pragprog.com/titles/fr_arr/cache_data_easily.pdf**.

MARCEL GAGNÉ

Evil Agents under the Bed and Other Scary Things that Go Boom!

If you are finding yourself losing sleep over possible intruders and ne'er-do-wells, it's time to relax and look at the lighter side of security threats.

Open up, François! I've been knocking for the last ten minutes. *Quoi?* You're afraid? Of what? But, that's ridiculous! Who else would be at the door at this time besides myself? Besides, I told you I was going to Henri's to pick up a case of today's wine. Sadly, the bottle you and I sampled earlier was the last one in the cellar, and I truly wanted to serve it for our guests. None of that explains why you are hiding behind the bar, keeping me outside knocking for ten minutes. Yes, of course, this month's issue is about security, but I still don't know why you are hiding in the dark.

Secret agents? Terrorists? Aside from the fact that none of those things are serious threats in this restaurant, that doesn't explain why all the computers are down. Logic bombs? *Mon ami*, the only bomb I am worried about at this moment is the one in your head. The Security issue isn't about national security or anything quite that dramatic. Usually, we mean computer security, and although that kind of security is serious, you aren't in imminent danger, and a logic bomb won't make your laptop explode. The battery inside is more likely to do that. Now, get up and get ready for our guests, many of them are already approaching. And, turn those computers back on. We will need them shortly.

Welcome, everyone, to *Chez Marcel*, where great wine, Linux and free software combine to make a feast like no other. Please sit and make yourselves comfortable. I won't be sending François to the wine cellar, as I brought the wine with me moments ago. Besides, my faithful waiter would likely cower in the darkness tonight. Don't fret, François. Tonight's wine is a 2004 Xanadu Cabernet Sauvignon from Margaret River in Western Australia.

Let's start with something really simple—slime. That's right, green-gooey slime. The game, written by Joey Marshall, is called *Slime Bomber*, and although it's alpha code and pretty basic, there's a fun element here that oozes you into the whole playing-with-explosives thing. It's also basic Python code and, therefore, open to simple hacking by anybody who might like to build on this theme. And, it's a pretty simple theme. Fly a bomber over various buildings, launch platforms and the occasional tree, and drop slime balls (Figure 2). That's it. Slime the world from overhead using slime bombs. No massive destruction here, just gooey fun.

Figure 1. Tonight's wine, direct from Xanadu, where Kubla Khan did his own sampling.

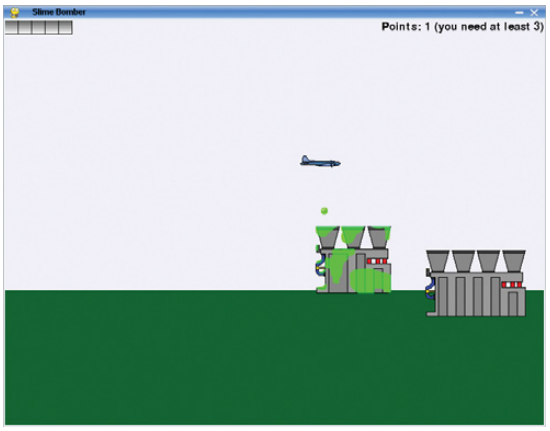


Figure 2. What could be more fun than sliming buildings?

To play the game, simply extract the tarball into the directory of your choosing, open a terminal

window, and from that directory, type the following:

```
python slimebomber.py
```

The game relies on the pygame package, so you need that to play. As for play itself, select a difficulty level, an aircraft type and click Play. Use the cursor keys to move your plane around, and press the F key to drop your slime. Given that this is alpha code, you'll be entertained only for so long with this one, so let's move on to something more explosive—slime, after all, doesn't go boom so much as plop.

It's on that gooey note that I move to a rather endearing game called *ClanBomber*, written by Andreas Hundt and Denis Oliver Kropp. *ClanBomber* itself is inspired by the hugely popular, not to mention long-running (since 1983) *Bomberman* game made famous by Nintendo (but originally created by Hudson Soft). *Bomberman* featured a robot working in a bomb factory, so the story line for *ClanBomber* is somewhat different, as are the characters: Tux, the BSD Demon and others. Each level features different layouts and obstacles. The bombs you detonate aren't just to get rid of your opponents, but also to open up walls and let you find and collect treasures. Meanwhile, a clock counts down the time left in that level's gameplay.

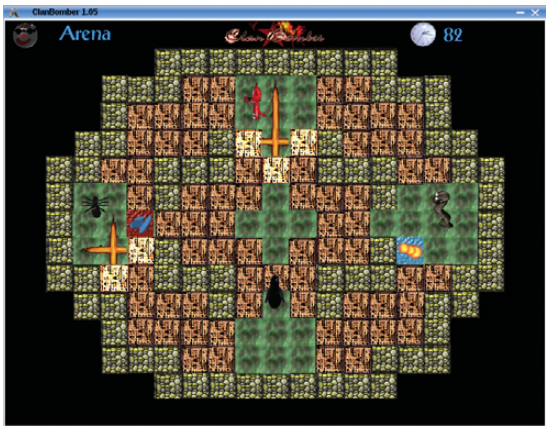


Figure 3. Plant bombs, move away quickly, collect treasures and blow up your opponents before they get you. *ClanBomber* is easy.

ClanBomber has several gameplay options, including defining and renaming AI players, turning off some of the players and more. When bombs go off in this game, body parts go flying, which might not make it a great choice for some, but that too is an option. You can reduce the number of corpse parts that get scattered, or you can switch to the friendlier Kidz mode (Figure 4).

Most distributions offer a version of *ClanBomber*

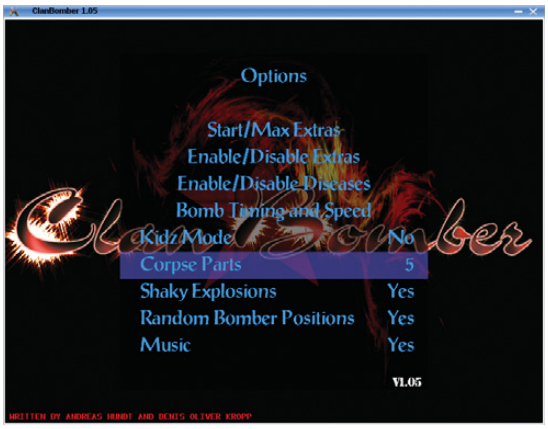


Figure 4. *ClanBomber*'s default display of bloodied characters and flying body parts can be reduced or turned off entirely.

that works with, wait for it, ClanLib. The latest *ClanBomber* has been redesigned and now works with DirectFB instead. If you do decide to check out *ClanBomber2*, you may need to build from source. This is your basic extract-and-build five-step, but it does have the prerequisite of DirectFB's FusionSound library.

Nothing says your bombs have to be bombs, per se. As I mentioned with the first game, slime can be fun. So can potato bombs and even tomato bombs. And, both of these fit in well with the theme of a restaurant. Let's start with the potatoes and a great game called *Hot Potato*. If you have ever played hot potato as a kid, you can probably guess where the computerized *Hot Potato* is headed.

Here's the premise. It is the future. Major-league sports have given way to a deadly form of the old hot potato game, where up to four players enter an arena and only one comes out. *Hot*

When bombs go off in this game, body parts go flying, which might not make it a great choice for some, but that too is an option.

Potato is a network-enabled, multiplayer game (although you can play it against a computer opponent) that is played inside an enclosed space. You race around this arena, along with up to three other players, picking up, tossing around and otherwise trying to get a potato bomb into the hands of the other player, preferably right before it blows up (Figure 5). It's very fast and good for getting your heart racing.

The potato is a bit like a time bomb in the sense that it has a short fuse and, therefore, offers little time before you need to get rid of it. Hit something with the potato, like another player, and it explodes.

Figure 5. This hot potato is something you really want to get rid of. Holding on too long has explosive consequences.

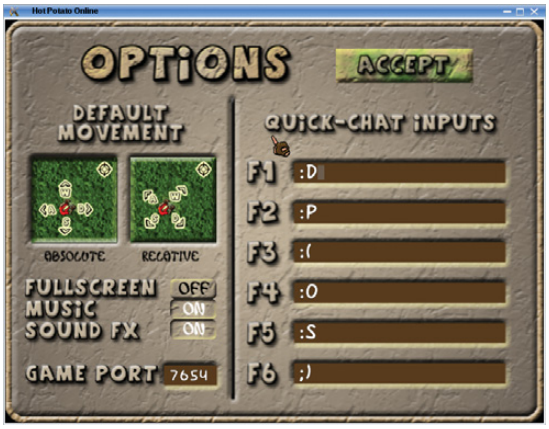


Figure 6. *Hot Potato*'s options screen defines screen and sound modes, network ports and some one-key chat messages.

Catch a potato thrown at you (by facing the thrower), and the timer resets, providing you with a chance to unload it on somebody else. You either can throw it or leave it where somebody else will run into it. The mouse defines direction, and a left-click tosses the potato.

When the game starts, you can select a local game or choose to connect to another server on the network. Should you decide to start your own server session, enter the lobby where you either can wait for other players to join you or start a match against an AI opponent. The AI also serves as your guide for learning your way around the game.

Hot Potato starts in full-screen mode, but you can override that in the Options screen (Figure 6). There you can switch to windowed mode, turn various sounds (including music) on or off, and define some quick chat responses to use during gameplay. When you don't have time to type, a single keystroke has to do.

The last item on tonight's menu reminds me of a

Figure 7. *I Have No Tomatoes* is frightfully mesmerizing. Drop bombs, collect jewels and avoid being crushed by other tomatoes.

bad old film from my youth called *Attack of the Killer Tomatoes*, not so much in the sense that it resembles it in any way, but more because killer tomatoes are generally hard to come by. *I Have No Tomatoes*, by Mika Halttunen, is a colorful, cheerful (despite the explosions), wonderfully addictive and totally engaging game. Your job, should you choose to accept it, is to smash, or blow up, as many enemy tomatoes as possible (Figure 7).

All this action takes place in a surreal landscape, floating in three-dimensional space. You move around a maze of sorts, dropping bombs, running to escape before the fuse blows. All this to smash other tomatoes—you see, you are a tomato as well. Some levels include teleportation devices to get you out of trouble fast, but for the most part, you just need to keep moving. If other tomatoes touch you, you are done for. At least, until you respawn a few seconds later.

I want to touch on some of the gameplay options, and one of those options requires special considerations, so I'll tackle it first. By default, the game starts with full-screen mode enabled. Should you want to play in windowed mode, you can do that; however, it requires that you manually update the game's configuration file. Here's a partial listing of the ~/.tomatoes/config.cfg file:

```
video_mode = 800 x 600
video_mode_color_depth = 32
video_mode_fullscreen = 1
sound_enabled = 1
sound_freq = 44100
```

If you change `video_mode_fullscreen` to 0 instead of 1, the play runs inside a window. Many changes can be made directly from the game's options screen without the need for editing a

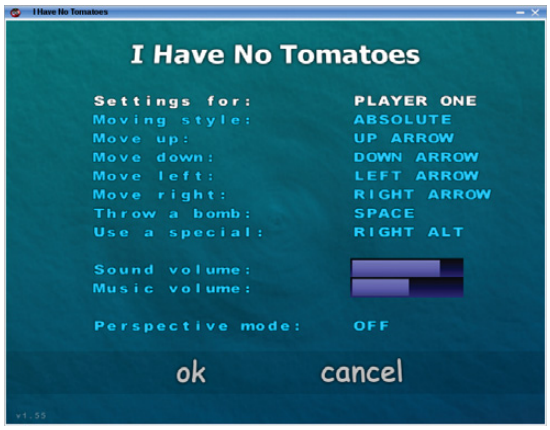


Figure 8. Many of the game's options, including movement, can be set in the Options menu.

Figure 9. Call up your specials by pressing the Alt key—lightning bolts, potato men, traps and more.

configuration file. To do that, simply select Options from the main screen, and you can change many settings, including the very important movement options.

Smashing tomatoes creates gems that you collect while traveling the maze. During gameplay, you may win additional "specials" as you collect these gems (Figure 9)—specials that you can bring into play by pressing the right Alt key (also configurable). These specials include lightning bolts, superhero potatoes, tomato traps and other strange and wonderful goodies.

So you see, *mes amis*, while the news keeps my faithful waiter fearful, we can step back and deal with all this trepidation with a little fun. Remember, no electrons were harmed in the making of these games, and everything is recycled. Exploding tomatoes, potatoes and slime balls won't make the six o'clock news, but they won't keep you awake at night either. Hmm...perhaps that's not the right sentiment. I recall spending many late hours playing games. François, I think this is where you refill our guests' glasses a final time and save me from trying to come up with a better example. Besides, it's closing time. Please, *mes amis*, raise your glasses and let us all drink to one another's health. *A votre santé! Bon appétit!*■

Marcel Gagné is an award-winning writer living in Waterloo, Ontario. He is the author of the *Moving to Linux* series of books from Addison-Wesley. Marcel is also a pilot, a past Top-40 disc jockey, writes science fiction and fantasy, and folds a mean Origami T-Rex. He can be reached via e-mail at marcel@marcelgagne.com. You can discover lots of other things (including great Wine links) from his Web sites at www.marcelgagne.com and www.cookingwithlinux.com.

Resources

ClanBomber: **clanbomber.sourceforge.net**

Hot Potato: **www.hotpotatoonline.com**

I Have No Tomatoes: **tomatoes.sourceforge.net**

Slime Bomber: **sourceforge.net/projects/slimebomber**

Marcel's Web Site: **www.marcelgagne.com**

Cooking with Linux: **www.cookingwithlinux.com**

TECH TIP Use netstat to See Internet Connections

Using netstat, you can monitor programs that are making connections to remote hosts:

```
$ netstat -tpe
```

The -t flag limits the output to show only TCP connections. The -p flag displays the PID and name of the program making the connection. The -e flag displays extra information, such as the user name under which each program is running.—**ERIK FALOR**

Special Variables I: the Basics

DAVE TAYLOR

Dave begins a new series of columns on shell variable notation.

There I was, trying to come up with a topic for this column, when I did what I usually do when stumped: I sent a question out to my Twitter followers. This time, I got a great answer, from John Minnihan: "How about how special vars inside a script, for example, `#!/bin/bash script="${0##*/}" current=`dirname "$0"` cd $current; make?`"

That's a good topic, so let's dig into it, starting with the basics this month, shall we?

The Easy Special Variables

The basic notation of variables in the shell is $varname, but I bet you've already used a few special notations without really thinking about it. For example, want to know how many positional parameters (aka starting arguments) you received when the script was invoked? Using $# gives you the value:

```
echo "you gave me $# parameters"
```

This can be quite helpful, because it means you can add multiple commands to your Linux shell with a single shared script base.

Want to get a specific positional parameter from the starting command line? That's done with other special variables: $1, $2, $3 and so on. These are rather odd cases actually, and the shift command shifts them all down one, so you easily can parse and trim command flags.

Try this snippet to see what I'm talking about:

```
echo "arg1 = $1" ; shift ; echo "now arg1 = $1"
```

The variable $0 is a special one in this sequence. It's the name of the script or program invoked. This can be quite helpful, because it means you can add multiple commands to your Linux shell with a single shared script base.

Let's say that you want to add "happy" and "sad" as two new command-line options, but you want to do it within a single script. Easy! Write the script, save as "happy", create a symbolic link that means "sad" points to "happy", and put this in the script itself:

```
if [ "$0" = "happy" ] ; then
  echo "I am so darn happy too, hurray!"
else
  echo "Sorry you're sad. Why not take a walk?"
fi
```

See how that works? It turns out that there's a nuance to this usage, however, because you often get the full path in the $0 variable, so most people use $(basename $0) instead of just utilizing the $0 directly.

Checking Your Status

Another special variable that you might have encountered is the status variable, $?. In a script, this contains the return value of the most recently executed (external) command.

This is where you need to read man pages so you know what to expect on success and failure, but as an example, consider the test command. According to the man page, "if [the expression] evaluates to true, it returns a zero (true) exit status; otherwise it returns 1 (false). If there is no expression, test also returns 1 (false)."

This means you could do this:

```
test 1 --eq 3
if $? ; then
```

Quick, now, would we be within this conditional statement or not? That's where it's tricky because zero = true and nonzero = false, which is somehow opposite to how we naturally think of conditional tests (well, how I think of them, at least). In fact, the above test would be testing 1, because the "test" would evaluate to false, and its return value also would be false.

Now, using test like this is a sort of daft example, but what if you wanted to create a subdirectory and then test to see if it was successful? That's a perfect use for $?, actually:

```
mkdir $newdir
if [ $? --ne 0 ] ; then
   echo "We failed to make the directory $newdir"
```

It turns out that you also can streamline this sort of thing by having the "if" directly evaluate the return code:

```
if mkdir $newdir ; then
```

That's a better coding style, although it can be confusing if you are used to having conditional expressions be value tests, not actually commands that *do something*.

A Few More Useful Special Variables

A special variable that I use with great frequency for helping create temporary file names is $$, which expands to the current process ID in the system. For example:

```
$ echo $$
3243
```

If you're doing a lot with subshells or spawning subcommands, another useful variable is $!, which is the process ID of the most recently spawned background command. I've never used this in any of my shell scripts, but you might find a situation where it's helpful.

The last example I'll talk about here is most useful when you want to hand starting parameters to subshells. The two options are $* and $@, and it's so convoluted to explain the difference that it's easier just to demonstrate.

Let's start with a tiny script that simply reports how many parameters it's given:

```
#!/bin/sh
echo "I was given $# parameters"
exit 0
```

I'll call that subshellcount.sh and utilize it like this:

```
#!/bin/sh
echo "you gave me $# variables and the first is $1"
echo "unprotected parameters:"
./subshellcount.sh  $1 $2 $3 $4
echo "or, more succinctly:"
./subshellcount.sh $*
echo "but when we put \$* in quotes:"
./subshellcount.sh "$*"
echo "by comparison, same thing with \$@:"
./subshellcount.sh "$@"
```

Watch what happens when I invoke it with three parameters, one of which has a space embedded:

```
$ sh test.sh I love "Linux Journal"
you gave me 3 variables and the first is I
unprotected parameters:
I was given 4 parameters
or, more succinctly:
I was given 4 parameters
but when we put $* in quotes:
I was given 1 parameters
by comparison, same thing with $@:
I was given 3 parameters
```

Can you see the difference here? When we don't take efforts to protect the space in the third positional parameter (either by just referencing $3 or using the $@ without quotes), it splits into two parameters to the subshell, and we get a count of four.

Quoting by itself doesn't do the trick either, because of the difference between $@ and $*. With the latter, everything expands without "breaking out of" the quotes, so $* ends up being a single positional parameter to the subshell. Fortunately, $@ works exactly as we'd like, and the subshell gets three parameters, not one, not four.

A special variable that I use with great frequency for helping create temporary file names is $$, which expands to the current process ID in the system.

It seems a bit trivial, but when you start working with filenames that have spaces in them, for example, you quickly will learn just how tricky it is to get all of this correct!

I'm going to stop here, and starting next month, we'll delve into the more obscure and complex shell variable notation. It's interesting stuff.■

Dave Taylor is a 26-year veteran of UNIX, creator of The Elm Mail System, and most recently author of both the best-selling *Wicked Cool Shell Scripts* and *Teach Yourself Unix in 24 Hours*, among his 16 technical books. His main Web site is at www.intuitive.com, and he also offers up tech support at AskDaveTaylor.com. You also can follow Dave on Twitter through twitter.com/DaveTaylor.

MICK BAUER

Samba Security, Part III

Start creating shares on your secure Samba file server.

This month, we continue our exercise in building a secure file server for our local LAN using Samba. In case you missed the first two installments, this is a non-Internet-accessible file server to which users of a LAN can mount virtual disk volumes.

The example scenario I'm using is a boarding house in which I need to provide a world-readable file share containing menus (SUPPER), a group-readable share containing schedules of chores (CHORES) and a private share containing copies of Web logs (BUZZ-OFF).

Last month, we used Samba's Swat tool to configure our Samba server's Global settings. We then created four user accounts: mick, knute, pepe and skippy. Mick, of course, is me. Knute, Pepe and Skippy are the three FBI agents who rent my rooms and who are interested in my daily menus and weekly schedules of chores, but with whom I'd rather not share my Web logs.

This month, we create a public share for menus called SUPPER and a nonpublic but group-readable share for chore lists called CHORES. (We'll save the private share, BUZZ-OFF, for next time.)

Creating a World-Readable File Share

As we've seen, Swat is arguably the best tool for configuring smb.conf, Samba's primary configuration file. Other tasks, like creating new user accounts, are best done from a command line (last month, we used the standard commands useradd and passwd to set up our accounts under Linux, and then smbpasswd to create corresponding Samba accounts).

To create shares, however, we can return to Swat. Unsurprisingly, the navigation button you must click is labeled Shares. After you do that, type the name SUPPER in the box to the right of the Create Share button, and then click that button. You should see something like Figure 1.

Under Base Options, I set comment to Mick's Menus. Then, I set path to /home/mick/supper. This will be our weekly menu folder.

The value of path has to correspond to a real directory on your server. Furthermore, the Linux permissions and ownership of this directory need to be set to allow the desired level of access you want to grant. In this example, the directory listing of /home/mick/supper looks like this:

```
drwxr-xr-x 2 mick users 4096 2008-09-12 01:44 supper/
```

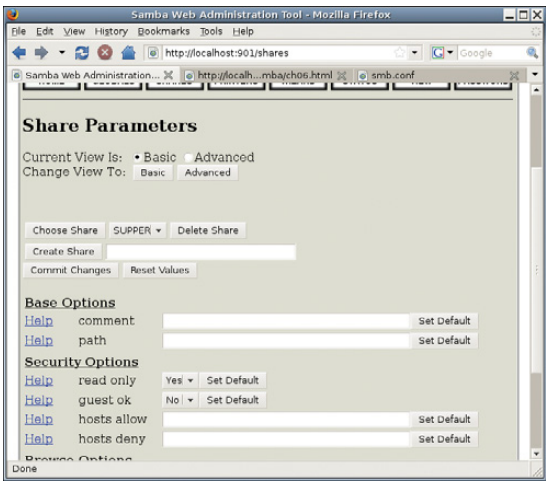


Figure 1. Creating a New File Share

As you can see, the user mick has read-write-execute permissions, but group and other have only read-execute permissions. Now isn't the time for a primer on filesystem security (actually I've already written one: "Linux Filesystem Security", in the October and November 2004 issues of *Linux Journal*). Suffice it to say for now that the commands for creating directories, setting user and group ownership and setting permissions, respectively, are mkdir, chown, chgrp and chmod.

Let's set some security options shown in Figure 1. By default, at least on Ubuntu systems, Swat displays only four options under this section in its basic view, but that's a reasonable starting point.

The first of these is read only, which I leave at the Ubuntu default of yes, even though I want the user account mick to be able to publish new menus. (The setting write list, which I'll describe a little later in this article will override this setting.)

The second security setting shown in Figure 1 is guest ok, which I change to yes. (My guests, and those of my boarders, certainly will be keenly interested to know what side dishes will accompany Tuesday night's Coconut Tater-Tot Casserole.)

I should pause here for a quick review of how guest access works in Samba. Last month, when we configured Samba's global settings, we set the option map to guest to Bad User, which caused Samba to treat clients who log in with nonexistent user names as guests. We set the option guest account to nobody, which means that when people

A Note on Figures 1 and 2

The screenshots in Figures 1 and 2 show Ubuntu's default values for the various settings in Swat. They, therefore, do *not* provide, all by themselves, a model of how to configure Samba securely! Read the accompanying text for my recommended (secure) settings.

log on as a guest (either by providing a bad user name or by actually logging in as nobody), they will be logged in under the account nobody.

None of these global settings has any effect on a given share unless that share's guest ok option is set to yes. As we'll see shortly, that doesn't actually give guests any *permissions* on that share unless we do just a little more work.

First, there are two more security options to attend to in Figure 1: hosts allow and hosts deny can be used to define TCP Wrappers-like, network-level access controls on your share. You can learn everything you need to know about this from the hosts_access(5) man page.

In Figure 1, hosts allow will be set to 192.168.44., which means "allow access from clients whose source IP address' first three octets are 192.168.44". In our example scenario, this corresponds to my local LAN address of 192.168.44.0/24. hosts deny is set to ALL, which means "deny access to all clients who do not match any value in hosts allow."

In my opinion, there's no good reason not to use hosts allow and hosts deny with Samba unless your LAN is very complicated. It's not as important as making proper use of user and group accounts, enforcing the use of strong passwords and other things you should be doing, but it's nonetheless a useful layer in our defense onion.

At this point you may be wondering, how do we tell Samba who has write access and who has read-only access for this share? The four security options we've covered don't address that. The answer is, we've already established some default settings for this in the global section, and share-specific authorization controls can be set by switching from basic to advanced view in Swat, by clicking the Advanced button near the top of the screen. When you do that, you'll see something like Figure 2.

But wait, what's this? Where did those values for valid users, read list and so forth come from, given my earlier sidebar note about these screenshots showing default settings?

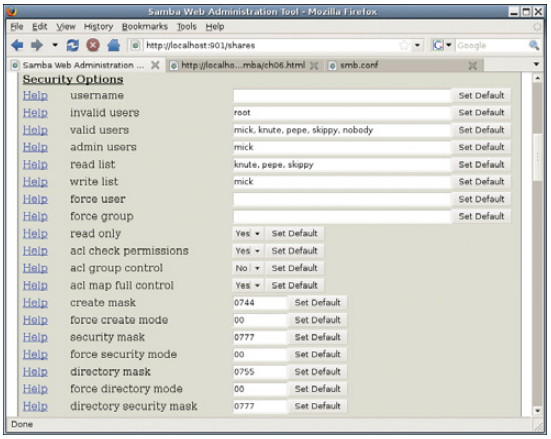


Figure 2. Share Security Options in Advanced View

As it happens, many of Samba's options can be declared *both* as global settings and as share-specific settings. When you set up a new share, Swat copies the values of any such options you set up under the global settings to the new share. So, Figure 2 represents Swat's settings after I've set up the global section but before I've fine-tuned the SUPPER share.

And, I do need to fine-tune it! On the one hand, invalid users is set to root as in the corresponding global option, which is a good value to propagate here; it's never a good idea to log in to much of anything directly as root.

But because I want this to be a public share, I'm going to remove all the users listed in valid users, which will have the effect of allowing clients to log in using *any* user name they provide. (Remember, though, anyone logging in with a user name outside the Samba user database or /etc/password will be logged on as nobody—that is, as a guest.)

Similarly, I'm going to empty read list as well, as read only is set to yes anyhow. (read list is sort of a blacklist: anyone whose user name is listed here will be granted only read access to this share regardless of *any other setting* in this share or under Globals.)

Another setting I'm going to empty is admin users. Like I said last month, this is a dangerous setting, and it's usually unnecessary. (I really shouldn't have set it to mick in the global section!) Not only will admin users operate with full Linux root privileges, all files they create will have a user owner of root, which can complicate both Samba and Linux filesystem permissions. Most of the time you might be tempted to set this option, it's probably sufficient instead simply to give that user write access.

And, you can do that with the option write list. In this case, we can leave the value of mick inherited from Globals.

The last security setting to change is create mask. This option determines the UNIX permissions that will be given to any files moved into or created in the share. Its value must be a chmod-style octal mode, as described in the chmod(1) man page.

The default value 0744, shown in Figure 2, translates to "owner read+write+execute, group read, other read". However, because this share is going to contain text files, there's no reason for the group-execute bit to be set; 0644 (owner read+write, group read, other read) is a better choice.

To review, and for clarity's sake, Figure 3 shows the changed settings for these security options in Swat's advanced view.

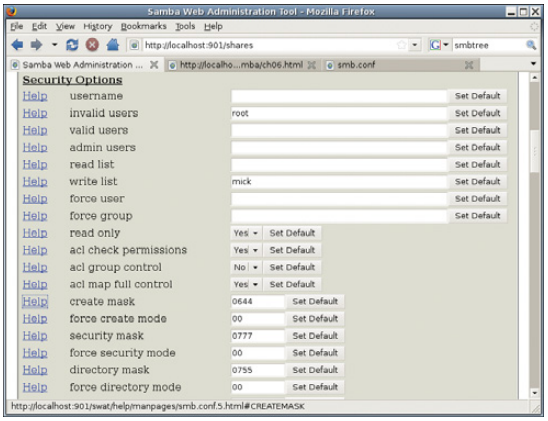


Figure 3. New Share Security Settings

We're almost done configuring this share. There are just two more options to check, and now you can switch back to basic view to find them quickly. The Browse Option browseable is set to yes by default on Ubuntu systems, which is appropriate for a public share.

The EventLog Option available, on the other hand, which is used to enable or disable a share, has the rather sensible default value of no. I say sensible, because it's never a good idea to activate anything before you're finished configuring and securing it! But, we are in fact done securing this share, so we'll change available to yes.

The last step is to click the Commit Changes button near the top of the Swat page. On my system, any time I click this button, the view resets to what appear to be default settings for printer shares! If this happens on your system too, all you need to do is click the Choose Share button again to display the changes you just committed.

After you create, delete or reconfigure a share, the changes will be applied immediately to your running Samba dæmons; there's no need to restart any of them.

Testing Samba Shares

Now that the SUPPER share is configured and available, it should start showing up in the Network Neighborhood (or other Windows network browser) of users connected to the LAN. Your Samba server, which we've configured to be a Browse Master for its workgroup, achieves this by sending out broadcasts.

However, in my experience, network browsers are often unreliable—it can take a while for your new workgroup, servers and shares to show up, and sometimes things disappear for no apparent reason. (Even for Windows clients, using the Map Network Drive feature to specify your share's path is both faster and more reliable than using the Network Neighborhood browser.)

So although you might get decent results testing your new share by simply firing up a network browser, I recommend using Samba's command-line tools instead, namely, smbclient and smbtree, which are included in Debian and Ubuntu's smbclient package, and in Red Hat and SUSE's samba-client package. I'll leave it to you to explore the smbtree(1) and smbclient(1) man pages, but I will give you a couple usage examples.

smbtree is a text-based Windows network browser that sometimes performs better than GUI-based browsers. To view all available workgroups, servers and public shares on your local LAN, use this command:

```
bash-$ smbtree -N -b
```

smbclient is a much more versatile command that can be used both to view and use Samba shares. To use smbclient to connect to our new share as the user nobody (guest), you can type:

```
bash-$ smbclient //CASA_DE_MICK/SUPPER -U nobody
```

Note the share-name syntax: `//<servername>/<sharename>`. You can use an IP address instead of the actual server name; this can result in a quicker login, because it allows smbclient to skip the name-resolution step. (Have I mentioned lately how inefficient the SMB/CIFS protocol is?)

Note also that to test the Bad User (guest-failover) behavior I described earlier, this command should be functionally equivalent to the previous one:

```
bash-$ smbclient //CASA_DE_MICK/SUPPER -U totallyfakeusername
```

You'll be prompted for a password. Simply press Enter without typing one (your nobody account shouldn't have a password!). If everything is working, you should see something like this:

```
Anonymous login successful
Domain=[FED-CENTRAL] OS=[Unix] Server=[Samba 3.0.28a]
smb: \>
```

At this point, you now have the Samba equivalent of an FTP shell—in fact, this environment is designed to be similar to FTP clients. To see a list of all available commands, you can enter ? or `help`. For now, we'll just do a quick directory by entering `dir`:

```
smb: \> dir
  .                                   D    0  Tue Oct  7   13:22:28 2008
  ..                                  D    0  Tue Oct  7   13:21:16 2008
  0-mon_filetmingon.txt               51  Mon Oct  6   21:05:34 2008
  1-tues_gruel.txt                    47  Tue Oct  7   13:05:54 2008
  2-wed_beefmushcasserole.txt          5  Tue Oct  7   13:06:32 2008

        52008 blocks of size 262144. 13782 blocks available
```

I'll leave it to you to figure out how to test copying files in both directions (`put` should work only for the user mick, but everyone else, including guests, should be able to list, get and read files).

Creating a Group-Readable File Share

On the strength of our SUPPER-creating experience, you'll find it fast and easy to create the group-readable share CHORES (which will contain lists of household tasks my boarders can perform in exchange for a rent discount). This share will be very similar to SUPPER: mick will have read and write access; pepe, skippy and knute will have read access only. However, unlike SUPPER, guest access will not be permitted.

Accordingly, after typing a new share name (CHORES) into the Create Share field and then clicking the Create Share button, we'll need to be sure to leave guest ok set to its default value of no. We'll set comment and path to Chore lists and /home/mick/chores, respectively (having first created this directory in a terminal window, and setting its ownership and permissions to be the same as for /home/mick/supper).

hosts allow and hosts deny can be the same as for SUPPER. browseable can be left at yes, but available should be left at no for now.

Figure 4 shows these settings (except available) for our new CHORES share.

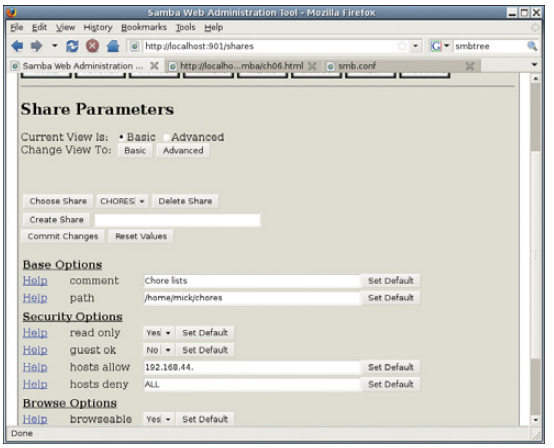


Figure 4. Basic View Settings (Customized) for CHORES

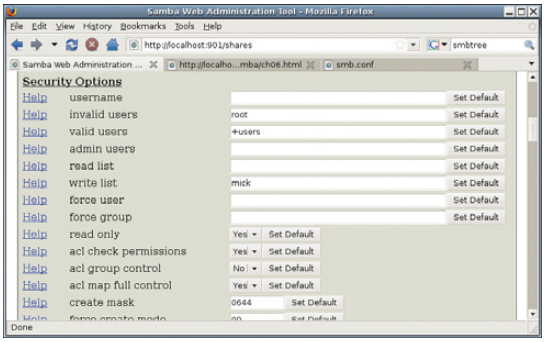


Figure 5. Advanced Security Settings (Customized) for CHORES

Now, we'll switch to Swat's advanced view for this share (if you aren't there already) by clicking the Advanced button. As with SUPPER, we'll blank out admin users, because we're paranoid, and also read users, as read only already is set to yes.

As you can see in Figure 5, however, I'm employing a bit of useful laziness in the valid users field for CHORES.

In the valid users field in Figure 5, the + in front of users instructs Samba to look up the name users in /etc/group, and then replace this entire value with a list of all members of the group users. Because on this server that group consists of mick, knute, pepe and skippy, Samba ultimately will set the value of valid users to mick, knute, pepe, skippy.

Needless to say, be careful with group names in this context. Before using one in Swat (or directly in smb.conf), be sure you know for certain exactly which user accounts belong to that group.

The quickest way to do this is to look up the group name in /etc/group and note its numeric value, noting also any secondary group members it has. Then, see which users in /etc/passwd have that group's number listed as its primary group.

Here's how this looks when enumerating the group users on my Ubuntu system:

```
mick@ubuntu@:~$ grep users /etc/group

users:x:100:

mick@ubuntu:~$ grep :100: /etc/passwd

dhcp:x:100:101::/nonexistent:/bin/false
mick:x:1003:100:Mick Bauer:/home/mick:/bin/sh
knute:x:1004:100:Knute:/home/knute:/bin/sh
pepe:x:1005:100:Pepe:/home/pepe:/bin/sh
skippy:x:1006:100:Skippy:/home/skippy:/bin/sh
```

As you can see, there are no secondary users listed at the end of the user's entry in /etc/group. My second grep command turned up five users, not the four I was expecting, but dhcp matched only because its numeric user ID (not its group ID) is 100.

The other settings we should change are create mask, which we'll again set to 0644, and then browseable, which we now can safely change to yes. Finally, we can click the Commit Changes button, and CHORES is ready to go. Preferably using another system, test it to make sure it works the way you expect.

Conclusion

That's all we've got space for this month. Next time, we'll create that third, mick-only share (I'll bet you can figure that out yourself beforehand), create persistent Samba mounts on our client systems using smbmount and at least briefly address some miscellaneous Samba security topics, such as how to make Samba automatically and safely serve people's home directories. Until then, be safe!■

Mick Bauer (darth.elmo@wiremonkeys.org) is Network Security Architect for one of the US's largest banks. He is the author of the O'Reilly book *Linux Server Security*, 2nd edition (formerly called *Building Secure Servers With Linux*), an occasional presenter at information security conferences and composer of the "Network Engineering Polka".

Resources

"Linux Filesystem Security, Part I":
www.linuxjournal.com/article/7667

"Linux Filesystem Security, Part II":
www.linuxjournal.com/article/7727

OUTRAGED
by the high cost of Fibre Channel or iSCSI Storage ?

Switch to AoE !

SR2461

Coraid Offers a Complete Line of Clustered Modular Storage Products:

- High Performance EtherDrive® SATA+RAID Storage Appliances with 1 GigE or 10 GigE Connections
- Clustered VirtualStorage™ Appliances *(a Revolutionary Logical Volume Manager)*
- Scalable NAS Gateways *(File Sharing with EtherDrive® Storage)*

Coraid's EtherDrive® Storage with AoE is fast, reliable disk storage that's easy to use. And it's much more affordable than iSCSI or Fibre Channel!

- Coraid products use open AoE (ATA-over-Ethernet) block storage protocol, for high performance without the TCP/IP overhead
- With AoE, your shared storage capacity is infinitely scalable – at a fraction of the cost of iSCSI or Fibre Channel storage
- We provide a 3-year warranty and free firmware upgrades on all our products, as well as support from first-rate engineers trained in our technology

Product Excellence Award Winner LinuxWorld Conference & Expo Best Storage Solution

EtherDrive® Storage has a field-proven track record and is 1000+ large data storage customers strong.

CORAID

The Linux Storage People

To learn more about this and Coraid's other products, go online or call

+1.706.548.7200 (toll free: **877.548.7200**)

www.coraid.com

Manage Multiple Servers Efficiently

KYLE RANKIN

Use a few simple techniques and a couple extra tools to simplify things when you must administer a group of machines at a time.

Through the years I've had to manage a wide-ranging number of different servers. At one job, I started with only a few and expanded to around ten, while at another job, I've managed hundreds. In both cases, I've found that you just can't accomplish everything you need to do efficiently when you log in to machines one at a time. Over the years, I've discovered a couple tools and techniques that certainly make it easier. Now granted, even these techniques can scale only so far. If you have a very large environment, you probably will be best served with some sort of centralized management tool like Puppet, cfengine or other tools that you can buy from vendors. Even so, for those of you who have a small-to-medium environment at work (or at home), here are some tricks to help you manage those machines better.

SSH Loops

A common need you have when there are more than a few servers in your environment is to run the same command on more than one machine. When I first had this problem, I came up with a pretty simple shell script:

```
$ HOSTS="machine1 machine2 machine3 machine4";
 ➥for i in $HOSTS; do ssh $i uname -a; done;
```

This one-liner iterates through each machine I've listed in the HOSTS environment variable and runs uname -a. You can, of course, replace uname -a with any command-line command that you would want to run on the hosts. For instance, one need I had was to keep all of my Debian servers up to date. I created a small shell script on each Debian host called /usr/local/bin/apt-automate:

```
#!/bin/sh

apt-get update && apt-get -u upgrade
```

Then, I edited my /etc/sudoers file, so that my regular user could execute that script as root without a password:

```
username ALL=(root) NOPASSWD: /usr/local/bin/apt-automate
```

(Replace username with your local user name for that host.) Once I had the script in place and sudo configured, I set up SSH keys so my user could log in to each of those machines easily. Then, I could update four hosts with a simple one-liner:

```
HOSTS="machine1 machine2 machine3 machine4";
 ➥for i in $HOSTS; do ssh $i sudo apt-automate; done;
```

Ultimately, I found I executed this one-liner so much, it warranted its own script, which I called update-all:

```
#!/bin/sh

hosts="machine1 machine2 machine3 machine4"

# Run the command on each remote host
for i in $hosts;
do
  echo $i;
  ssh $i sudo apt-automate;
done;

# Also run the command on the local machine
sudo apt-automate
```

Now, this system worked for me at the time, but it has plenty of room for improvement. For one, I potentially could set up a set of environment variables for different host groups. Then, instead of defining HOSTS each time I ran the one-liner, I could reference one of those groups.

ClusterSSH

When I had only a few hosts to manage, the SSH loop method worked well for me. However, that plan didn't scale quite so well when I needed to manage a few hundred machines in different data centers. For one, I didn't always just need to run a command on a group of machines. Sometimes, I wanted to make the same change to the same file on each of the hosts. Although I could play with Perl or use awk and sed scripts to edit files in-line, that was prone to mistakes. Lucky for me, I found an invaluable tool for

managing small-to-medium server environments called ClusterSSH (**clusterssh.sourceforge.net**).

ClusterSSH opens a terminal for every machine you want to manage. In addition to these terminals, ClusterSSH opens a small Tk control window. Anything you type into one of the individual terminals will execute just on that server, but anything you type or paste into the Tk window is input into every terminal. The control window also allows you to toggle whether input goes to a particular terminal and allows you to add extra hosts as well.

ClusterSSH is packaged by a number of distributions. If your distribution doesn't have it, you also can download and build the source from the project page. Once the package is installed, execution is simple:

```
$ cssh host1 host2 host3 host4
```

A nice feature of ClusterSSH is that it automatically will tile all of the windows for you so that you get the maximum amount of visible screen space on each (Figure 1). This is particularly useful when you operate on a large number of servers at the same time. If you happen to rearrange the windows or add or remove hosts from ClusterSSH, you can press Alt-R or click Hosts→Refile Hosts to rearrange all the windows.

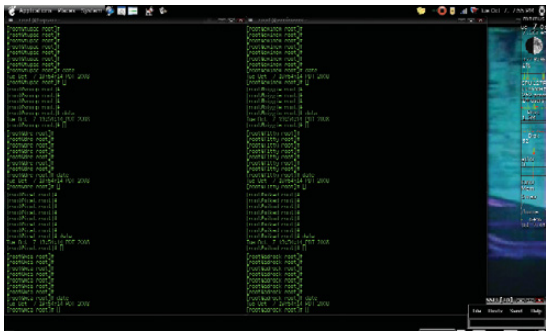


Figure 1. Ten terminal windows tiled by ClusterSSH.

Now you might be saying, "That all looks fine, but you still have to specify all the servers on the command line each time. What if I have a cluster of 30 servers to manage?" Well, ClusterSSH has that covered via its configuration files. In the ~/.csshrc file, you not only can define default settings for ClusterSSH, such as terminal settings, but you also can define groups of servers. If you want to change settings for all users, you can define clusters in the /etc/clusters file and set ClusterSSH parameters in /etc/csshrc. Otherwise, ~/.csshrc works fine as a place to store all the settings for your user. Here's a sample ~/.csshrc that highlights some of the useful options:

```
terminal_args = -fg green
terminal_font = 7x14
clusters = web dbtest dbprod dns
web = web1 web2 web3 web4 web5 web6 web7 web8 web9 web10
dbtest = testdba@db1.test.example.net testdba@db2.test.example.net
dbprod = proddba@db1.prod.example.net proddba@db2.prod.example.net
dns = root@ns1 root@ns2 root@10.1.1.1
```

The first two options in this file configure terminal settings. First, I set the foreground to green on my xterm (since green on black is the one true terminal color), and then I set the terminal font. The third line sets the clusters option and defines aliases for all the clusters you will define below. Note that if you define a cluster in this file but don't remember to add it to the cluster option, you won't be able to access it. Below the clusters option, I've defined a number of different clusters. The syntax is essentially clustername = serverlist with each hostname separated by spaces. As you can see in the examples, you can specify servers strictly by hostname (in which case your DNS search path will attempt to resolve the fully qualified domain name), by the host's fully qualified domain name or by IP. If you want to log in under a different user name, you also can specify that on a host-by-host basis.

Once your configuration file is in place, you can connect any or all of the cluster aliases on the command line. So, if I wanted to run a command on all the Web servers I would type:

```
$ cssh web
```

If I wanted to access both the dbtest and dbprod servers, I would type:

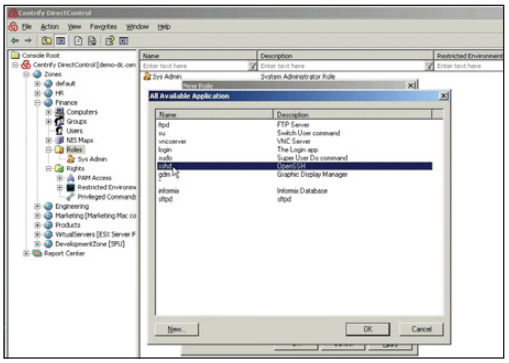
```
$ cssh dbtest dbprod
```

One downside when you specify multiple host groups is that if you don't have SSH keys set up, you might have to type in different passwords for each host. In that case, you need to highlight each terminal window individually and then log in. After that, you can return to the Tk control window and execute commands across all hosts.

All in all, I've found ClusterSSH to be an invaluable tool for managing small-to-medium groups of servers. The interface is pretty straightforward, and there is something so cool about being able to paste 20 lines of configuration to a vim session across 30 hosts or quickly run tail against all of your Web server logs. I've found I use it the most to deploy packages to groups of servers. I can single out one server to make sure the package works correctly, then toggle that server off and apply it to the rest.■

Kyle Rankin is a Senior Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *Knoppix Hacks* and *Ubuntu Hacks* for O'Reilly Media. He is currently the president of the North Bay Linux Users' Group.

Centrify Suite 2008



The new Centrify Suite 2008 is an integrated family of Active Directory-based auditing, access control and identity management solutions for cross-platform environments. The applications also help address regulatory compliance, says its maker Centrify, by adhering to requirements from SOX, PCI, HIPAA, GLBA and FISMA. The Standard Edition contains two applications: DirectControl, which secures non-Microsoft platforms using the same authentication and Group Policy services found in a Windows environment, and DirectAuthorize, which provides centralized role-based entitlement management for fine-grained user access and privilege rights on UNIX and Linux systems. The Enterprise Edition adds DirectAudit, which offers auditing, logging and real-time monitoring of user activity on non-Microsoft systems. The Application Edition, meanwhile, is for organizations using Web/Java applications, databases or enterprise applications, such as SAP or PeopleSoft.

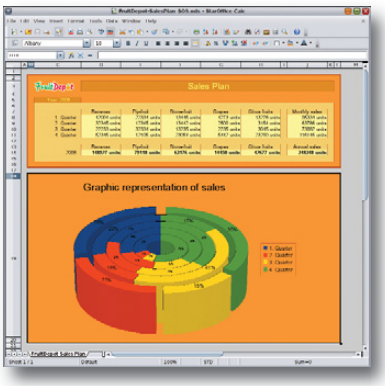
www.centrify.com

Primera Technology's Bravo Disc Publishers

It appears that our constant pestering for Linux support on various devices is paying off. The latest manufacturer to announce Linux support is Primera Technology, maker of a range of disc publishers, which announced support for its Bravo II, BravoPro, Bravo XR and Bravo XRP CD/DVD/BD devices. Primera says that its full-featured Linux printer drivers can be integrated with open-source or commercially available disc-burning engines easily. The drivers can be downloaded from the firm's Web site.

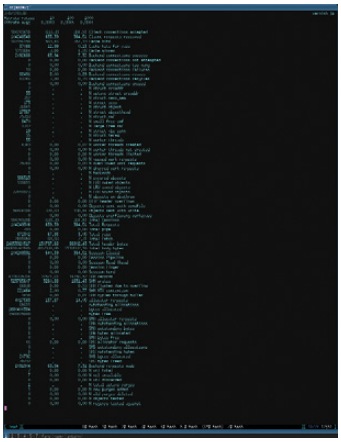
www.primera.com

Sun Microsystems' StarOffice



StarOffice, the enterprise-oriented sibling of OpenOffice.org, has been upgraded to Version 9. This open-source office productivity suite contains the Writer word processor, Calc spreadsheet, Impress presentation, Base database and Draw drawing/graphics applications. StarOffice Version 9 adds features, such as Mozilla Thunderbird for e-mail and Lightning for calendaring, an enterprise migration tool and various extensions for blogging, communicating, wiki publishing and PDF editing. Further, like OpenOffice.org 3.0, StarOffice 9 can read and write Microsoft Office .docx files. A range of support models are available; indemnification against intellectual property lawsuits is included in each. StarOffice comes in Linux, Solaris and Windows flavors.

www.sun.com/staroffice



Redpill Linpro's Varnish

The new Varnish 2.0 from Linpro is an open-source reverse-Web accelerator for high-content Web sites that was designed from the ground up for incoming traffic and not as a client-side proxy or origin server. Varnish temporarily stores the most frequently requested pages in cache memory and offers tools for identifying which pages should and should not be cached, and if they are cached, when to delete them and present fresh content. The result, says Linpro, is a 90% reduction in server requirements. Varnish 2.0 offers new features like improved compression, expanded support for filtering Web content for caching, ESI language support, tighter integration with CMS solutions, load-balancing support, better scaling and improved accelerator tuning. Varnish runs on Linux, Solaris and FreeBSD.

varnish-cache.com

TotalView Technologies Tool for Source Code Analysis and Memory Error Detection

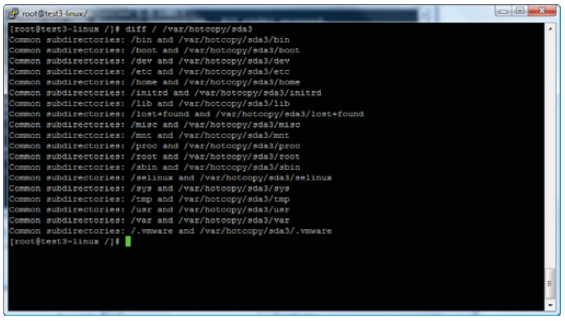
TotalView Technologies recently upgraded to Version 8.6 its TotalView tool for source code analysis and memory error detection. Most notably, this latest release adds TVScript, a new troubleshooting utility offering a streamlined mechanism for automated and unattended debugging. In addition, the new SSH-based Remote Display Client allows users to set up and operate securely an interactive graphical debugging session on remote systems located anywhere. The Remote Display Client is available for 32- and 64-bit Linux and Windows.

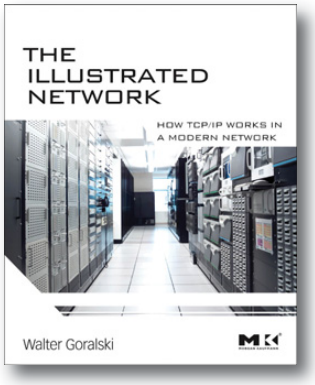
www.totalviewtech.com

R1Soft's Hot Copy

The new Hot Copy from R1Soft is a Linux command-line utility that takes on-line snapshots of disks or volumes on a Linux server. Because Hot Copy does not use LVM, it can work on any Linux system and with any block device. Some sample applications are turning legacy backups into on-line ones, creating a copy before running or testing dangerous scripts and commands (for example, `rm -Rf`), running fsck safely while the filesystem is mounted and viewing changes on systems. Features include instant, non-interrupting point-in-time snapshots of any block device, point-in-time snapshots with the system in a totally consistent state, copy-on-write snapshots, writeable snapshots and no need for dedicated snapshot devices or storage.

www.r1soft.com





Walter Goralski's *The Illustrated Network* (Morgan-Kaufmann)

This new book from Walter Goralski and Morgan-Kaufmann, *The Illustrated Network: How TCP/IP Works in a Modern Network*, updates the classic *TCP/IP Illustrated* from W. Richard Stevens to apply to 2008 equipment, OSes and routers. The book contains 330 illustrations, such as screenshots and topology diagrams, which portray examples from a real, working network configuration, including servers, routers and workstations. The publisher says the illustrated approach "allows the reader to follow the discussion with unprecedented clarity and precision". *The Illustrated Network* is device- and platform-agnostic.

www.elsevierdirect.com

Rupert Howell and Jonathan Wong's *Apache OFBiz Development* (Packt)

If you're setting off on an open-source enterprise-automation project, first download Apache's OFBiz, and then grab the new book *Apache OFBiz Development: The Beginner's Tutorial*. The book is authored by the team of Rupert Howell and Jonathan Wong and published by Packt. Apache OFBiz contains ERP, CRM, POS, e-business and e-commerce, SCM, MRP, CMMS/EAM and other applications. The book's design is to give newcomers a hands-on introduction to OFBiz, covering the main modules and employing illustrated examples that show how to build applications rapidly. In addition to the Model-View-Controller framework, readers will gain working knowledge of Widgets, Entities and the Service Engine. Finally, readers will learn how to tweak OFBiz as well as get tips on performance enhancement and development.

www.packtpub.com

Please send information about releases of Linux-related products to newproducts@linuxjournal.com or New Products c/o *Linux Journal*, 1752 NW Market Street, #200, Seattle, WA 98107. Submissions are edited for length and content.

Fresh from the Labs

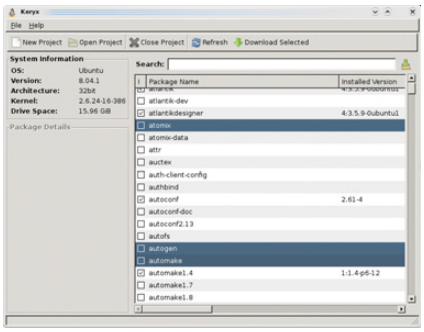
Keryx—Packaging Solution for the Net-Deprived

(keryx.betaserver.org)

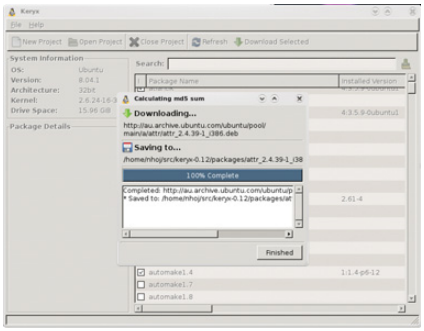
One of my chief bug bears of Linux systems over the last eight years or so is the high level of dependence on Net connectivity and the constant assumption that you even have a connection in the first place. "I'm trying to compile MPlayer, but there are dependency problems." "Just install it with apt, it's easy." "I don't have the Net." Blank stare. My twin brother, for example, is a loyal Linux user and lives in a flat where it's hard to get a connection, and as a musician, it's very hard for him to stay in Linux to do his work, because the programs he needs have niggling dependencies. These can take a day to resolve when he has to go to an Internet café, grabbing random .deb files and hoping they work.

Well for all you Net-deprived people, I feel your pain, and so does Chris Oliver, with his new program Keryx. Keryx is a free, open-source application for updating Ubuntu. The Keryx Project started as a way for users with dial-up or low-bandwidth Internet to be able to download and update packages on their Debian-based distribution of Linux. Mainly built for Ubuntu, Keryx allows users to select packages to install and check for updates and downloads those packages onto a USB key. The packages are saved onto the device and then can be taken back to the Linux box to be installed. Because of the design, Keryx can be run on *any* OS that has Python, GTK and PyGTK installed. For Ubuntu (GNOME) users, everything is pre-installed. Windows users also will have no software to install, because Keryx and everything it depends on will be made to run portably off a USB Flash drive.

Installation If you've got a standard Ubuntu system, you're set. If you have a variant of some sort, make sure you install all of the standard GTK, Python and PyGTK libraries before



Keryx offers an impressive way of managing packages on PCs without a Net connection.



Keryx saves packages that can be used between multiple machines and distributions—very handy.

continuing. Head to the Web site, grab the latest tarball, extract it somewhere locally and open a terminal in the main keryx directory. And, that's it.

Usage In the main keryx folder, enter the following command:

```
$ python keryx.py
```

You'll be greeted by the main screen where the first thing you need to do is start a new project with the aptly titled New Project button. Each project is designed to keep track of a different computer's packages, meaning you can take care of multiple machines with the one USB stick. Once you've entered your project name, you'll be prompted to choose between Local Files or Internet. Local Files is meant for those without any connection at all, but at this point, there's no technical difference between either Local Files or Internet. At this stage, Keryx will appear to hang, but it isn't, it's just processing a whole bunch of repository information—things about local files and so on. Give it a minute or two, and it should be back with you.

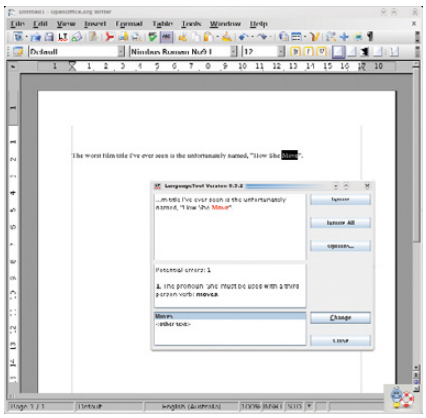
Once Keryx has sprung back to life, you'll be presented with a long list of packages, Synaptic style. For choosing packages to install, the interface is a little quirky. Those tick boxes won't let you choose a package; they just tell you whether it's installed already. To install a package, click on the actual name of the package, and if you want multiple packages, Ctrl-click or Shift-click the same way you would in any modern file manager. When you're ready to download the packages, click Download Selected at the top-right of the screen. Keryx will download everything and save any downloaded packages to the packages folder in the main keryx directory.

From here, you'll have to install these packages yourself manually, either by command line with dpkg or with a package management program under X. It's a bit of pain, I'll admit. However, this project is very young, the interface is still very much in its infancy, and adding the option to install the packages from within Keryx should take only a few GUI shortcuts to some pretty basic commands. It's in its early days, but it does genuinely look promising, with a planned Mac port even in the works once the project becomes more stable. Poor Linux enthusiasts without the Net rejoice. In the near future, your savior may be arriving in the form of Keryx!

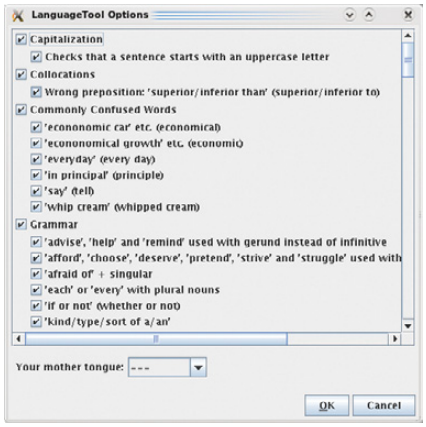
LanguageTool—Style and Grammar Checker for OOo

(www.languagetool.org)

I find that after submitting an article and reading it again a few days later when my brain's fresh, I've made some heinous grammatical error somewhere and not noticed it. And, that's what I've just sent to the editor.



LanguageTool—it's like having your own Noel Coward plugin for OpenOffice.org.



There's an impressive array of grammatical rules available with LanguageTool.

Spiffing. Well, it's not like the spell-checker picked it up, is it? I read through it several times, but still, I missed it. Well, Daniel Naber has just the thing for me with the imaginatively titled LanguageTool.

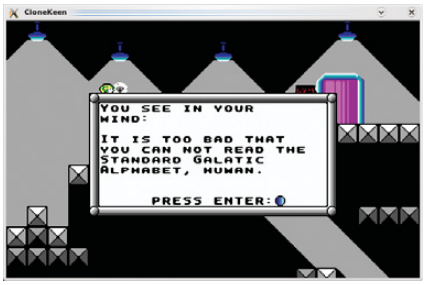
LanguageTool is a grammar-checking plugin for OpenOffice.org based on Java with support for English, Polish, German, French and Dutch, and basic support for some other languages, such as Swedish and Russian. LanguageTool scans words and their part-of-speech tags for occurrences of error patterns that are defined in an XML file, and more powerful error rules can be written in Java and added later.

Installation Head to the Web site, but before you download the plugin, you need to choose between two versions. One is for the 2.x series; the other is for the newer 3.x beta series. If you'd like a demo before you install it, there's a link on the site to do just that, and it'll run in your browser provided you've got basic Java plug-ins. Speaking of Java, you need version 5 of *Sun's* Java, not one of these alternative jobbies. Once you've selected your version, save it to the hard drive and open up your version of OpenOffice.org Writer.

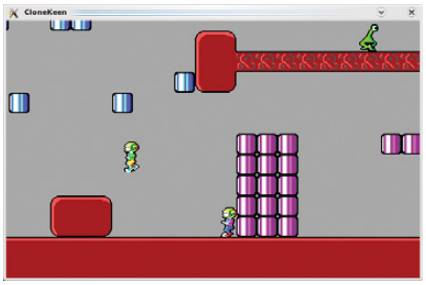
To install the plugin, click Tools→Extension Manager, and once inside the Extension Manager window, click the Add... button and browse for the .oxt file you downloaded earlier. Once you've done this, LanguageTool should be installed. Close OOo and restart it, and it should be good to go. Before we move onto usage though, I can't stress enough how important it is to have the right Java packages installed. If you have Sun Java 5 installed and the following steps aren't working for you, make sure you install all of the other Java packages, like jre and so on.

Usage With LanguageTool installed, the first thing you need to do is choose your language. Click Tools→LanguageTool→Configuration, and once inside the configuration screen, choose your default language under the drop-down box titled Your mother tongue:. Notice that big list of language rules? It's pretty impressive, don't you think? For those with OOo 3.x, life is slightly easier. Simply type some text in the main screen, and it should check it automatically (the Web site recommends typing "This is an test." for some deliberately bad grammar). For those on the 2.x series of OOo, you need to choose Tools→LanguageTool→Check Text each time you want to check some text.

Once installed, I found LanguageTool an intuitive tool with a familiar interface that I now will use in my daily work (much to the joy of our editor I should imagine). Check it out.



Another deeper gameplay element of *Commander Keen*: its very own alphabet that is decoded later in the series.



CloneKeen adds some crazy new elements to the original *Keen* like this insane two-player mode.

CloneKeen—*Commander Keen* Port

(clonekeen.sourceforge.net)

At the very beginning of the 1990s, side-scrolling platformers were the order of the day, and gaming consoles were having unprecedented success with the likes of *Mario Bros.* and *Sonic*. So, what about the PC? Enter *Commander Keen*. Developed by the now-famous id Software, *Commander Keen* (or just *Keen* as it was often called) had unrivaled gameplay, level design, smooth scrolling and a solid feel to it that was missing in other games. id soon would go on to develop other ground-breaking titles, such as

LanguageTool is a grammar-checking plugin for OpenOffice.org based on Java with support for English, Polish, German, French and Dutch, and basic support for some other languages, such as Swedish and Russian.

Wolfenstein 3D, *Doom* and eventually, *Quake*, and in the same way that these landmark games were all superior to their rivals, *Keen* had the gameplay and feel to it that was simply unmatched. Play it now, and it still makes sense. Get six-year-olds to play *Keen* for five minutes, and you won't have to explain why it's good or say how great it was at the time—they'll just know. And, it's not just nostalgic me that sees it as a classic either; any Steam users can download the series and play it through the DOSBox emulator on their modern PCs. But, that's still really just emulation, and Caitlin Shaw has other ideas with *CloneKeen*— a restoration of the original three *Keen* episodes running natively using SDL, making it portable to a large number of platforms including Linux, Windows, the GP2X, the Dreamcast and PSP.

Installation Unfortunately, *CloneKeen* still is in a state of flux and needs some cleaning up on the Linux side. I got *CloneKeen* working and compilation certainly is doable, but any comprehensive instructions would be too long to include here and may well have changed by the time this goes to print, so please check the readme file and the Web site's instructions. That's about all I can say in that regard; however, I can give you a few tips before you embark on a compilation fest. First, you need a copy of the original episodes, and more important, you need to copy these into *CloneKeen*'s data folder. Second, once in the src folder, you need to copy the Makefile.lnx to the Makefile like so:

Unfortunately, *CloneKeen* still is in a state of flux and needs some cleaning up on the Linux side.

```
$ cp Makefile.lnx Makefile
```

Third, enter `make clean` before entering `make`, or you'll run into errors. But finally, Caitlin herself says that she just mostly uses the Windows binary package and copies the compiled Linux keen binary into the folder of the Windows package and runs the keen binary from there (and trust me, for the moment, it's easier). I realise that's not really all that helpful, but hopefully by the time you read this, the installation will be cleaned up.

Usage If you've been lucky enough to get it working, any key will get you into the main screen. Under Options, you can adjust the screen size so that you don't have a tiny little window, but I recommend full screen for the authentic feel with smooth scrolling. Start a new one-player game, and you can control the character using the arrow keys, with Ctrl for jump, Alt for the pogo stick once you have it, and Ctrl and Alt in combination to fire the raygun. Otherwise, I'll let you figure it out from there (especially the two-player mode, which I haven't had the proper chance to explore).

Overall, this project is still a bit unstable, with screen errors, sound errors and the like, but if you can get it working, it's well worth the effort. This game really is a classic, and ten minutes of playing time should speak for itself. Plus, the addition of the crazy two-player mode as well as new options, such as "Fully Automatic Raygun", should give the game a breath of fresh air and a new angle of play. Give it a go or even check it out on Steam if you're lazy. In the meantime, I'm going to have a go at the PSP version.■

John Knight is a 24-year-old, drumming- and climbing-obsessed maniac from the world's most isolated city—Perth, Western Australia. He can usually be found either buried in an Audacity screen or thrashing a kick-drum beyond recognition.

Brewing something fresh, innovative or mind-bending? Send e-mail to knight.john.a@gmail.com.

TECH TIP Handle Compressed and Uncompressed Files Uniformly

When looking at log files or other files that are compressed and rotated automatically, it's useful to be able to deal with them in a uniform fashion. The following bash function does that:

```
function data_source ()
{
 local F=$1

 # strip the gz if it's there
 F=$(echo $F | perl -pe 's/.gz$//')

 if [[ -f $F ]] ; then
  cat $F
 elif [[ -f $F.gz ]] ; then
  nice gunzip -c $F
 fi
}
```

Now, when you want to process the files, you can use:

```
for file in * ; do
 data_source $file | ...
done
```

If you have bzip2 files, just modify the data_source function to check for that also.

—DAVID A. SINCK

EmperorLinux

...where Linux & laptops converge

Powerful Linux: The Rhino

- Based on the Dell Precision M6400/Latitude E6500
- High performance NVidia 3-D on a WUXGA widescreen
- High performance Core 2 Quad, 16 GB RAM
- Ultimate configurability — choose your laptop's features

Features include:

- 2.2-3.0 GHz Core 2 Duo/Extreme (dual-core) or 2.5 GHz Core 2 Quad QX9300 (quad-core)
- Up to 17" WUXGA LCD w/ X@1920x1200
- NVidia Quadro FX 3700M graphics, 128 core CUDA
- 80-320 GB hard drive (7200 rpm SATA) or 64 GB solid state drive, up to 16 GB RAM
- DVD±RW or Blu-ray, Ethernet, 802.11a/g/n, Bluetooth
- One year Linux tech support - phone and email
- Three year manufacturer's on-site warranty
- Choice of pre-installed Linux distribution:

Portable Linux

Toucan T500/W500

- ThinkPad T500/W500 by Lenovo
- Up to 15.4" WUXGA w/ X@1920x1200
- ATI Radeon graphics
- 2.2-2.8 GHz Core 2 Duo
- Up to 4 GB RAM
- 100-320 GB hard drive / 64 GB SSD
- 5.3-6.5 pounds
- Starts at $1260

Tablet Linux

Raven X200 Tablet

- ThinkPad X200 tablet by Lenovo
- 12.1" WXGA w/ X@1280x800
- 1.2-1.86 GHz Core 2 Duo
- Up to 4 GB RAM
- 80-320 GB hard drive / 64 GB SSD
- Pen/stylus input to screen
- Dynamic screen rotation
- Starts at $2365

Rugged Linux

Tarantula CF-30

- Panasonic Toughbook CF-30
- Fully rugged MIL-SPEC-810F tested: drops, dust, moisture, & more
- 13.3" XGA TouchScreen
- 1.6 GHz Core 2 Duo
- Up to 4 GB RAM
- 80-320 GB hard drive
- Call for quote

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

HARDWARE

Mixing It Up with the Behringer BCF2000

The BCF2000 provides pro audio performance at podcasting prices—for Linux!

DAN SAWYER

Linux and open source are practical matters for me. I couldn't run my business without them. But occasionally, the demands of a job grow way beyond what the tools I'm using can handle.

Take Audacity, for example. As far as sound-effects-editing software goes, it strikes almost an ideal balance between user-friendly and extremely powerful. Snd and ReZound let you do a lot more, and Sweep lets you bring in nondestructive editing and some other nifty things, but all of them sacrifice a certain amount of intelligibility in the process (from the non-engineer's perspective). Now, I am an engineer, at least in the practical sense. I've been editing, recording and mixing audio now for almost a decade, and I do know better than to use Audacity for complicated long-form projects. Knowing better and doing better are two different things, and Audacity is just so darn simple that it's easy to get stuck with it even when you know better than to consider using it for certain kinds of jobs. Like, for example, my current big project: a 13-hour full-cast audio book with ambient sound, original music and complex stereo imaging.

I've long used Ardour for recording and for mixing music, but for the past several years, I've used Audacity mostly to do my mixing and sound FX editing. I must confess, I've actually mixed a number of long-form video projects, several short films and countless long-form podcast episodes in Audacity over the last few years, before the post-production work I was doing got complicated enough that I needed to be able to work with the signals in ways that Audacity simply doesn't let me do. The need to change EQ and reverb parameters over time, do complex stereo imaging and subtle sound-layer shifting all jumped out at me in glaring relief when I launched my recent dramatized podcast novel.

Figure 1. Behringer BCF2000

However, shifting to Ardour for mixing (instead of just recording) immediately opened up a whole new wondrous world where my options quickly multiplied to the point of paralysis. A 20-track mix isn't a big deal when you're mixing down mono and you're doing simple, sound separation EQs, but when you're using elements that change over time on each track, the time that goes into mixing a show goes up exponentially with every new element you add. Mixing it all one element at a time with a mouse can be done, but as I found out very quickly, that way madness lies.

In the world of well-monied studios, such things are handled by devices called control surfaces. In the most basic sense, a control surface is a mouse that's shaped like a mixing board. It plugs in to a computer's MIDI port and uses the MIDI command language to control different elements in a given piece of software. Ardour (along with most MIDI programs, like Rosegarden) plays very nicely with control surfaces that are supported by the kernel. Most good control surfaces with motorized faders, like the ones made by Mackie, start selling at around $800 for an eight-track unit. This is well out of the price range for hobbyists, and it's a stretch for small studios like mine. However, there is another surface on the market at $200 that competes very well with the $800 Mackie, and it is completely, gloriously supported by the Linux kernel.

The device is the Behringer BCF2000 (Figure 1), and it has a number of nice little features. It has eight faders, eight pan pots, 16 programmable buttons, an additional bank of four buttons for transport control (play, stop, fast

forward and so on), and all of these buttons, faders, dials and switches are programmable, groupable and toggleable so that, with the proper configuration, you can control up to 32 tracks at any given time.

But, it gets better. The units are stackable—you can link a number of them in a daisy chain and have them act in tandem, and you also can link another MIDI device, such as a keyboard, through the BCF2000. The scalability of the unit is a big deal—a 24-track Digidesign control surface runs around $10,000, while three stackable Behringers cost only $600 plus another $30 or $40 for extra MIDI cables and will give you 80% of the same functionality. (For that last 20% on the Digidesign 24-track systems, you get more sophisticated transport control, more programmability and a real jog/shuttle wheel. If you're creative with your configuration though, you can approximate a jog/shuttle on the Behringer, and stacking the units will give you everything a hobbyist or a small studio really needs.)

Although you can use the Behringer control surface family (the BCF2000 is one of several models in the BC line) with any MIDI program that supports control surfaces, if you're looking to control Rosegarden or TerminatorX, the companion BCR2000 might be a better bet for you. The internal electronics are nearly the same, but the physical interface is better for voice and event triggering, while the BCF2000 is laid out like a mixing board and is ideal for the kinds of complicated mixing that I do for my audio projects.

Setting It Up

Setting up the Behringer is pretty straightforward. Take it out of the box, plug it in to the wall, hook it up to your computer over the USB port or the MIDI port and power it up. Before you actually can use it, it'll need a firmware update. If you go to **www.behringer.com/05_support/bc_download/bc_downloads.cfm**, you'll find the latest version of the firmware. Download the most recent package, unzip it and follow the directions inside. You load the firmware to the unit with a cp command—no Wine or DOSEMU necessary.

Setting up the unit after this actually is pretty simple. Pull up a JACK controller, such as qjackctl (Figure 2), and start JACK. Then, start Ardour. Now, in the Connections window, look at the ALSA tab. If you've plugged the interface in through your USB port, it will show up as an ALSA-MIDI device (Figure 3).

Figure 2. qjackctl Main Interface

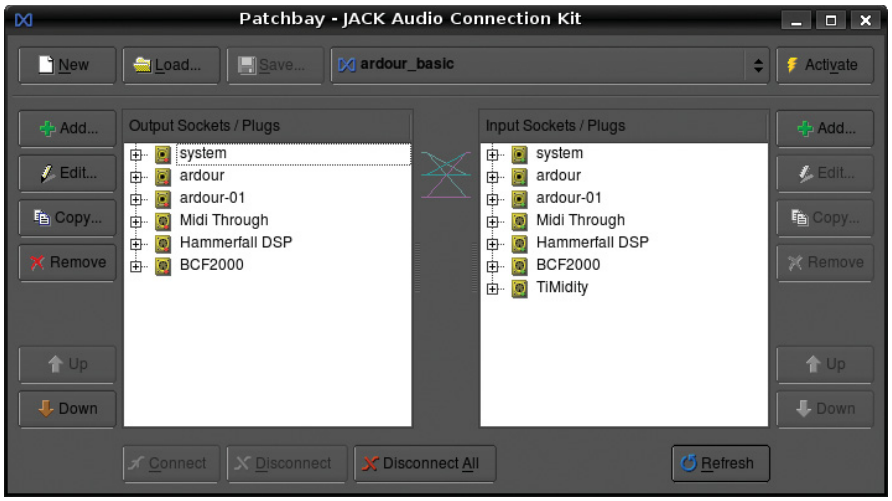


Figure 3. ALSA Tab of the Connections Window in qjackctl

When that's done, cross-connect Ardour and the BCF2000, so that each will control the other. This allows you to control Ardour with the faders and pots on the BCF, and it allows Ardour (with a little extra work) to feed back to the BCF on playback—this sounds kind of gimmicky on the surface, but trust me, it becomes really important, really fast, later on (more on that later).

Once you've cross-connected the surface and Ardour, you can save the setup for future sessions, so you don't have to go through this rigmarole every time. Click on the patchbay button in qjacktcl. In the patchbay window that appears, click New, and then press Yes when you're presented with a dialog that asks whether you would like to "Create patchbay definition as a snapshot of all actual client connections". Save the definition. Now, any time you start JACK, you can load up that patchbay setup by selecting it and clicking Activate.

Making It Work with Ardour

When it comes to working with the BCF2000 in Ardour, once you get the basics down, everything else is pretty straightforward. There is a caveat though. Depending on your distribution and the version of Ardour you're running, everything might not work. So first, let's check to see whether everything's kosher.

First, using the presets controller on the mixer, set it for preset 2 (this is the factory preset most congenial for mixing). This preset designates the bottom right-hand bank of four buttons as your transport controls, controlling the following (starting from the top left and going clockwise): Locate 0, Fast Forward, Play and Stop.

Open Ardour, and set up a project suitable for mixing. Under File, select Add Tracks, and add seven new tracks,

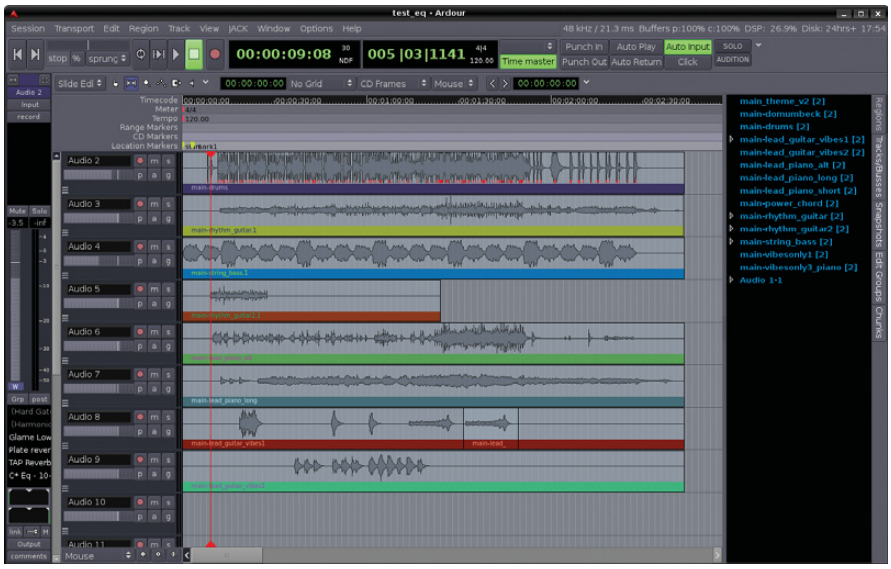


Figure 4. An Eight-Channel Song Mixdown in Ardour

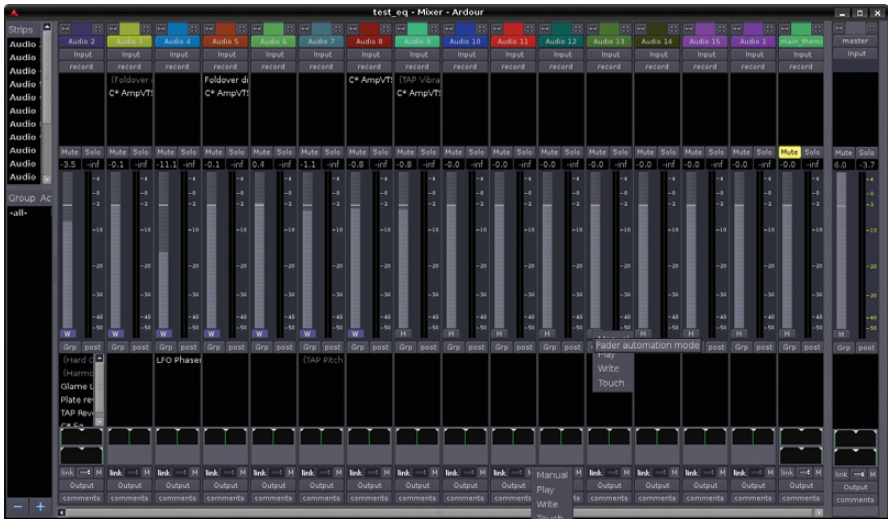


Figure 5. Ardour's Mixer Window with the Automation Modes for a Pan Pot and a Fader Showing

just for kicks (mono or stereo doesn't matter—pick what you prefer). When presented with the editor window, before you do anything, go to the Options pull-down menu and select Control Surfaces. Under the secondary menu that appears, make sure that General MIDI is checked and Mackie is unchecked. Then, under the tertiary menu Controls, check Feedback. Once this is done, you should be able to assign controls to the faders, pan controls and the jog/shuttle control. In order to do this, simply mouse over the control you want to assign (chose a fader first), then hold down Ctrl and click your middle mouse button. You'll see a little floating window pop up that says Operate Controller Now. Do what it says—operate the controller on the BCF2000 that you want to have control the interface element you're trying to assign. As you move the control on the mixer, you should see a corresponding change in the program's GUI.

Now, here's the fun part. Take your mouse and move the fader in Ardour—that same one you just assigned. You should see the fader you assigned to the track move on the mixer in response to manipulating the interface. If everything is working both ways, you're ready to roll.

If you run into problems, particularly problems with getting the faders to fly properly, take a look at the relevant portion of the manual for instructions on debugging: **ardour.org/files/manual/sn-bcf2000.html**.

Using the Surface

Now that your surface is up and running, it's time to mix your first project. To start, you're going to need some sounds. Record or import a few sound files, and line them up on your tracks (Figure 4).

In the Window pull-down menu, select the Show Mixer option, and switch over to the mixer window. At the bottom of each track's fader, you'll see a little blue button that says either M, W, P or T. This sets the automation mode of the track: Manual, Write, Play or Touch, respectively. Manual mode is what you use if a track needs a constant volume level throughout the project—sometimes. For a simple mix, this might be all you need, but if that was all you were doing, you wouldn't have bought a control surface (Figure 5).

To perform your mix and write automation to the project, you need to set a track to "write". Be careful though; if you leave it set on write and then play the transport, it *will* write—and overwrite all automation you may have programmed already. Always, always, turn write mode off unless you're actively writing automation.

To play back and check your work, set the mode button to Play. To play it back and make adjustments as you go, set it to Touch mode, which plays through the existing automation, but begins writing if you adjust a fader, for as long as you're writing a fader.

An analogous situation works for pan pots at the bottom of the track—these pots can be assigned to pots on the board so that you can automate stereo imaging (instruments or people moving through the audio space, bullets whizzing across the room and so forth).

So, set the pots and faders for the tracks you want to work with to Write mode, press Play and ride your controls. That's all there is to it.

Stepping It Up: Mackie Emulation Mode

Using the Behringer as a MIDI control surface is nice, but it does require

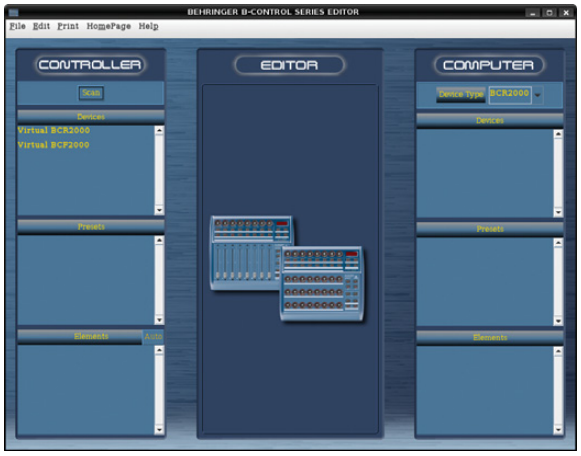


Figure 6. Behringer's cross-platform preset writer—works well in Linux.

hand-assigning every button for every project. In my experience, it also doesn't do a good job at honoring the bank selectors—in MIDI mode you have eight tracks' worth of controls, and only that. If you want to mix a 24-track project, you have to be good about grouping your submixes and break your project down into passes. It's a viable way to work, but it can become a pain, and reassigning your faders as you go can confuse you when you change over (naturally, if you're running a number of BCFs in tandem, this limitation ceases to be a serious problem).

There is a better way to use the BCF2000 with Ardour, and that's in Mackie Emulation mode. Basically, you tell Ardour you're already connected to an eight-track Mackie control surface. The Mackie preset gives you a seven-plus-master mix layout, with pan pots at the top (except for the master track—there your pan pot is a jog/shuttle wheel) and each track having mute and solo buttons—very handy. It assigns the tracks in numbered order from left to right (corresponding to your track order in Ardour from top to bottom), with track eight being the Master bus.

Why is this a better way? It gives you access to all the controls on the BCF. MIDI mode allows easy assignment of pots and faders, but try assigning one of the buttons, and you'll find yourself quickly tempted to burn the thing at the stake. Button presses seem to register on assignment, but then when you go to use them, they don't work. This problem may be correctable by building a preset in Behringer's preset building software (Figure 6), depending on the preset you build and your version of Ardour. Your mileage may vary.

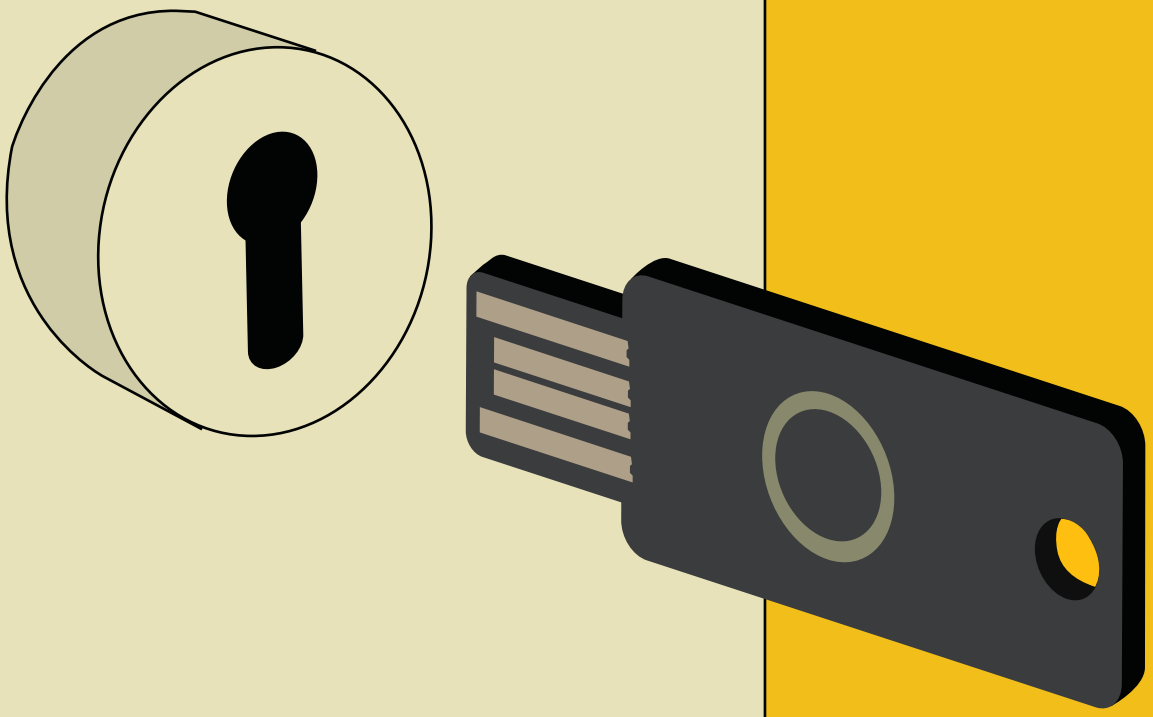
Section 10.6 of the manual gives detailed (and accurate) setup instructions for putting the Behringer in Mackie Emulation mode. Unfortunately, the effectiveness of Mackie mode seems to be in flux in Ardour's current development cycle. Some versions work very well—others don't work at all. Again, your mileage may vary (**www.ardour.org/files/manual/sn-mackie.html**).

Conclusion

Despite the bumps in the road due to Ardour's rapid development cycle, I wouldn't trade this little mixer for the world. It's easily saved me ten hours a week mixing down my podcasts, and the quality of the mixes has gone up as well. Mixing software faders with a mouse is a sucker's game compared to the precision you get mixing hardware faders with your fingers. For $200, this control surface delivers motorized faders and high-definition response in a well-designed, solid package that's fully supported by the Linux kernel and ALSA-MIDI.

That means it's also useful in a number of other high-level MIDI and audio programs for Linux, such as Rosegarden or LMMS or other programs that can accept MIDI control symbols. Let the mixing begin!■

Dan Sawyer is the founder of ArtisticWhispers Productions (www.artisticwhispers.com), a small audio/video studio in the San Francisco Bay Area. He has been an enthusiastic advocate for free and open-source software since the late 1990s. He currently is podcasting his science-fiction thriller *Antithesis* and his short story anthology *Sculpting God*. He also hosts "The Polyschizmatic Reprobates Hour", a cultural commentary podcast. Author contact information is available at www.jdsawyer.net.



YUBIKEY One-Time Password Authentication

How to add one-time passwords to your own system for added security without investing in an expensive authentication infrastructure.

A number of factors inspired me to take a closer look at the Yubikey. For starters, it is such a simple and elegant solution to two of the major problems the security industry is facing these days: authentication and identity management. Furthermore, I really like how Yubico, the manufacturer of Yubikey, is trying to integrate the Open Source movement into its business strategy. In this article, I cover three topics related to this little device. First, I explain what the Yubikey does and how to use it. Second, I examine how it works. Third, I show how to integrate the Yubikey authentication service into your own infrastructure without too much trouble.

DIRK MERKEL

What Is It?

A Yubikey is a small plastic rectangle that basically consists of a USB connector and a button. It resembles a tiny USB Flash drive, and as it measures only 18x45x2mm and weighs only 2 grams, it easily can be carried on a keychain or in a wallet (Figures 1 and 2). When you plug it in to your machine's USB port, it identifies itself as a keyboard, implying that the Yubikey is platform-independent as long as the host device supports data entry via the USB Human Interface Device (HID) specifications. It draws power from the host device and, thus, does not have to depend on an internal battery. The whole device is quite compact and can be attached to an actual key ring using the small hole near the top of the device. The gold surface connectors are quite robust and are expected to last the lifetime of the device. According to a Yubico representative, Yubikeys still were usable after running them through a washing machine's cycle.

Each time you press the button on the device, it generates a one-time password and sends it to the host machine as if you had entered it on a keyboard. This password then can be used by the service to authenticate you as a user.

Figure 1. Yubikey Plugged In

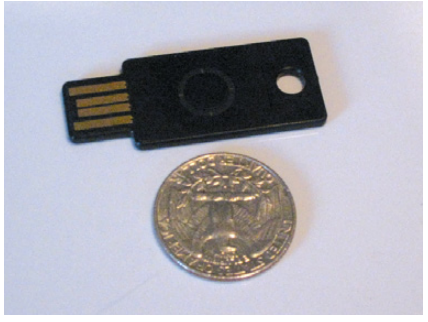


Figure 2. Yubikey Size

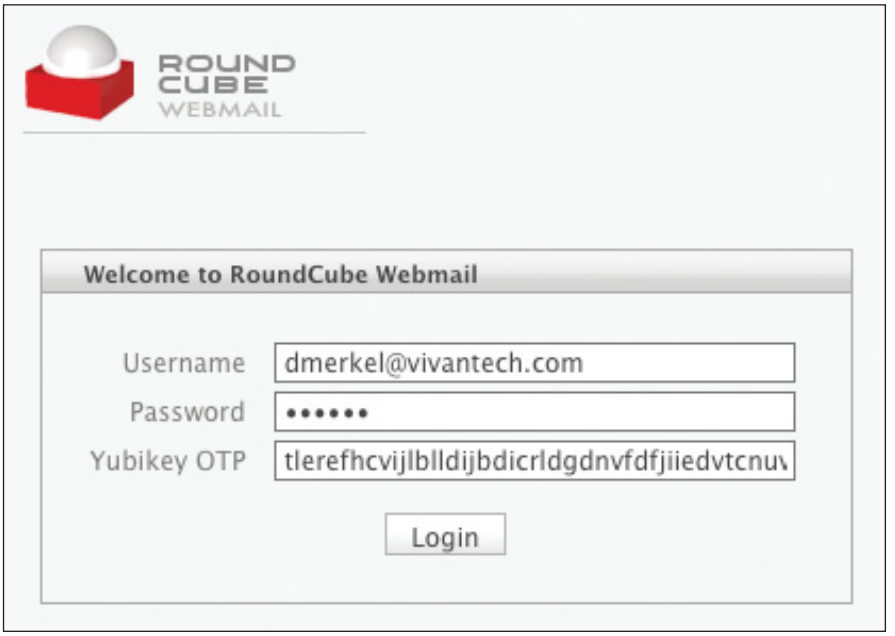


Figure 3. Modified RoundCube Login Form UI

How Do You Use It?

I use RoundCube to read my e-mail when I don't have access to my own system. RoundCube is an AJAX-centric Web-based e-mail client. You use it via your Web browser just as you might use Gmail or most other major on-line e-mail providers. Fortunately, RoundCube is open source and based on PHP, so it didn't take too much work to add Yubikey authentication.

Normally, RoundCube asks you to enter your e-mail address and password to log in. However, following a few modifications, the login screen now features a third field: Yubikey OTP (one-time password). Now, all you have to do is enter your e-mail and password as usual, position the cursor in the newly added text field, and put your finger on the Yubikey's button. After a second or so, the Yubikey magically spits out a 44-character sequence followed by a newline character. The newline character causes the form to be submitted. And, assuming that your Yubikey is indeed associated with your account, you will be logged in. Take a look at Figure 3, which shows the slightly modified login screen.

For obvious reasons, the Yubikey should not be used as the only method of authentication. If that were the case, someone getting a hold of your Yubikey then would be able to access your Yubikey-enabled accounts provided that person also knows your corresponding login. However, if you use the Yubikey to add another attribute to a multi-attribute authentication scheme, it can increase security significantly. Imagine if you will, people monitoring your network traffic without your consent. They may be able to glean your password by examining captured TCP packets, but the Yubikey password they capture will be of no use to them, because it can be used only once! After you use a Yubikey password to log in somewhere, it becomes useless. In the next section, I explain exactly how this one-time password scheme works.

More Details

Let's take a closer look at the character sequence the Yubikey transmits to the host machine. Here's an example of a sequence generated by my Yubikey:

```
tlerefhcvijlngibueiiuhkeibbcbecehvjiklltnbbl
```

The above is actually a one-time password that is secured using AES-128 encryption and ModHex encoding. Let's take a look at how the Yubikey constructs this string. For the purpose of this discussion, refer to Figure 4.

The device starts by creating a 16-byte sequence (Figure 4) where the individual bytes are allocated as follows:

- The first six bytes hold the key's secret unique ID, which is assigned when a Yubikey is programmed. This ID is

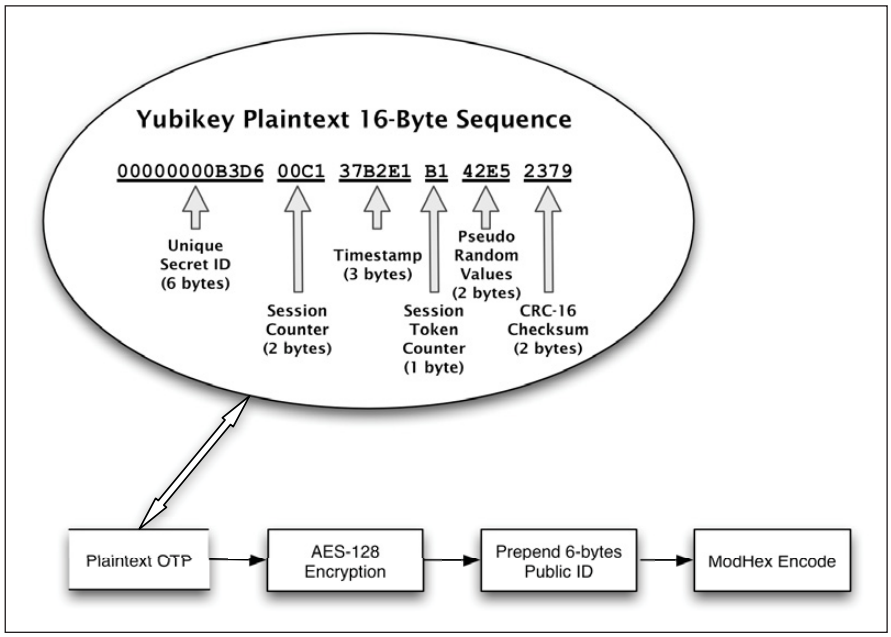


Figure 4. Yubikey Token Construction

known only to the entity that assigned it and cannot be retrieved from the Yubikey. Six bytes translates into $2^{(6*8)}$ = 281,474,976,710,656 unique combinations of bits, which is the number of Yubikey IDs that can be issued before Yubico has to think of a new scheme. Considering that this number exceeds the current world population by a factor of more than 42,000, Yubico is not likely to run out of unique IDs for some time, unless its business model is more successful than anyone could anticipate.

- The next two bytes in our sequence, bytes 7 and 8, are used to store a session counter in nonvolatile memory. The counter starts at zero and is incremented each time the device is plugged in. Two bytes for the session counter allows for $2^{(2*8)}$ = 65,536 sessions. In other words, you can plug in the Yubikey three times a day for almost 60 years before running out of session counters. Note that you can generate a significant number of OTPs during each session (see below).

- The following three bytes, bytes 9 through 11, are used as a timestamp, which is stored in volatile memory during each session. That means each time the device is plugged in, the timestamp starts at zero and continuously increases. Because it is incremented by an internal 8Hz clock, timestamp values will be exhausted after about 24 days. At that time, you need to unplug the Yubikey and plug it back in.

- Byte 12 in the sequence is a session counter that starts at zero and is incremented by one each time a token is generated. When it reaches that maximum value of 255, it wraps back to zero.

- Bytes 13 and 14 in the sequence are pseudo-random numbers provided by a free-running oscillator. These bytes are used to add additional entropy to the plain text before subjecting it to the cypher.

- The last two bytes, numbers 15 and 16, contain a checksum using the CRC-16 algorithm over all values of the token with the two checksum bytes set to zero. This checksum is used for data-integrity checking.

Each time the Yubikey is invoked, it generates the 16-byte sequence described above. However, if you look at the sample Yubikey output previously listed in this article, you will notice that it actually consists of 44 characters. That is because we still are missing three crucial steps before the Yubikey is ready to spit out the final token. First, the 16-byte token is encrypted using an AES-128 key that is unique to each Yubikey. Second, the Yubikey prepends the encrypted 16-byte token with a six-byte plain-text public ID. This public ID is completely different from the secret ID used to construct the 16-byte sequence. The public key does not change and can be used to associate a Yubikey token with an account. Finally, the whole 22-byte sequence (16 bytes encrypted plus six bytes public ID) will be encoded using the not-so-well-known ModHex algorithm.

Yubico chose this algorithm simply because it is limited to characters that are common to many different keyboard layouts. Because the Yubikey impersonates a keyboard, it tries to use characters that work with the various keyboard settings it might encounter in the wild. The disadvantage is that ModHex encoding is somewhat inefficient in that it requires two characters for each byte it encodes, which is why a 22-byte sequence turns into a 44-character sequence. However, as the Yubikey does all the typing, this does not translate into an inconvenience for users.

More about Encryption

Let's take a closer look at the encryption step of generating the token. In contrast to asymmetric algorithms used in public-key encryption schemes, such as PGP, AES is a symmetric algorithm. This means both the party encrypting the token and the party decrypting and validating it will need access to the AES-128 key! This sharing of the AES key happens when the device is programmed. Similar to the device's unique ID, the unique AES-128 key is generated and stored on the device by Yubico before it is shipped out. The company maintains a database where the unique public as well as secret IDs are associated with their corresponding AES keys. This way, Yubico is able to offer an authentication Web service.

Using a symmetric algorithm has the advantage that it is typically very fast. Also, you don't need to rely on third parties for key management or to vouch for identities.

If you want to be in charge of your own AES key, you have two options. First, you can request your AES key from Yubico. At the time of this writing,

Yubico will send you a CD containing the AES key, but the company also is working on a more convenient solution of retrieving the key on-line. Second, you can use Yubico's development kit to program the key yourself. This way, you can assign AES-128 keys, as well as public and secret IDs, according to your own naming conventions. If you supplement this approach by running your own authentication Web service, you eliminate any dependence on Yubico as a third party in your authentication procedure.

The Validation Algorithm: Order Matters

It's not surprising that the process of validating an OTP resembles reversing the steps necessary for constructing an OTP. A basic validation routine might look something like this. First, you ModHex decode the string. Next, you split the string into public ID and 16-byte token. Then, you use the public ID to look up the corresponding AES key. After using the AES key to decrypt, you have the original 16-byte token in plain text. Next, you would verify the CRC-16 checksum (the last two bytes). Then, you would compare the secret ID to the one you retrieved from the database using the public ID. Using the session counter and the session token counter, make sure that the current token was generated after the last successfully authenticated token. Although you don't know exactly when any two tokens were generated, you always can tell in which order they were generated. If the token passes all these tests, you can send a response signaling successful validation to the client. Otherwise, the token is rejected.

Optionally, you can harden the validation algorithm further. For example, you can try to calculate how many sessions or tokens have been skipped since the last successful validation and consider that information in your decision to validate or reject the token. You can use the session timestamp in a similar manner.

Yubico's Open-Source Approach

One thing I find really attractive about Yubico's business model is that it tries to provide all software in the form of open source. According to Yubico's statements, it plans to profit from the manufacture and sale of the devices, but intends to keep all software open source. For example, the source code for the aforementioned Web service is freely available as a reference implementation. Furthermore, Yubico offers client libraries needed for implementing Yubikey authentication in various applications and platforms. Currently, there are client libraries available in Java, C, C#/.NET, PAM, PHP, Ruby, Perl and Python. All these libraries and programs are set up as Google Code projects. Additionally, there are projects for libraries to decrypt OTPs in C and Java, as well as an Open ID server and a personalization tool to allow you to program your own Yubikey. Although all these software projects were initiated by Yubico, you already can see others contributing. Moreover, a number of independent open-source projects using the Yubikey technology have surfaced. Yubico's discussion forum is a good place to keep tabs on such projects and get support.

The Yubico Authentication Service

When you order a Yubikey, it comes ready to take advantage of Yubico's authentication Web service. Because Yubico maintains a database of all API keys, as well as public and secret IDs with which the Yubikeys have been programmed before shipment, Yubico has decided to offer an authentication Web service against those credentials. Developers then can use the Yubico authentication Web service to validate OTPs captured from the device. Yubico has a Web page where you can request an API key. Anyone can get an API key. The only requirement is that you

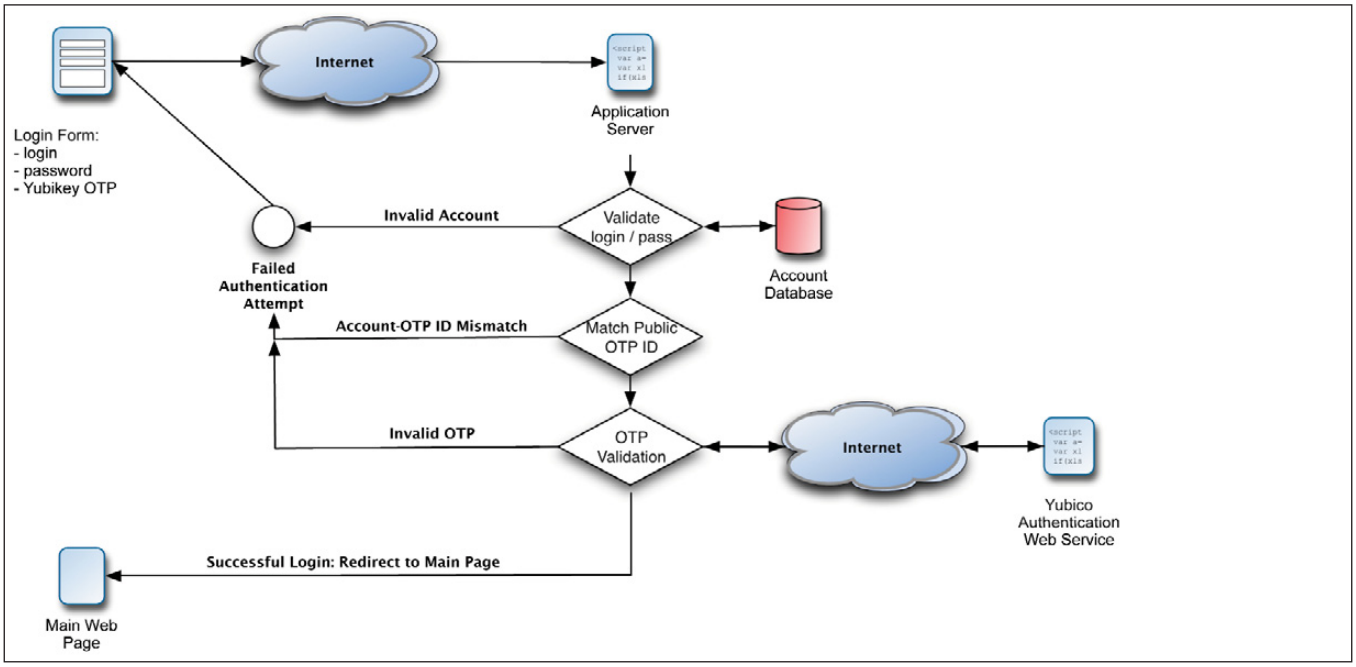


Figure 5. Yubikey OTP Validation Flow

Listing 1. Typo: Blog-Wide Yubikey Settings HTML

```
filename: app/views/admin/settings/index.html.erb

...
<!-- Yubikey authentication - start -->
<fieldset id="authentication" class="set" style="margin-top:10px;">
  <legend><%= _("Authentication")%></legend>
  <ul>
    <li>
      <label class="float"><%= _("Require Yubikey OTP")%>:</label>
      <input name="setting[yubikey_required]"
          id="yubikey_required" type="checkbox" value="1"
          <%= 'checked="checked"' if this_blog.yubikey_required%> />
      <input name="setting[yubikey_required]" type="hidden"
          value="0" />
    </li>
    <li>
      <label for="yubikey_api_id"
          class="float"><%= _("Yubico API ID")%>:</label>
      <input name="setting[yubikey_api_id]" id="yubikey_api_id"
          type="text" value="<%=h this_blog.yubikey_api_id %>"
          size="6" />
    </li>
    <li>
      <label for="yubikey_api_key"
          class="float"><%= _("Yubico API Key")%>:</label>
      <input name="setting[yubikey_api_key]"
          id="yubikey_api_key" type="text"
          value="<%=h this_blog.yubikey_api_key %>" size="50" />
    </li>
  </ul>
</fieldset>
<!-- Yubikey authentication - end -->
...
```

Listing 2. Typo: Adding Blog-Wide Yubikey Settings to Model

```
filename: app/model/blog.rb

...
  # Authentication
  setting :yubikey_required,        :boolean, false
  setting :yubikey_api_id,          :string, ''
  setting :yubikey_api_key,         :string, ''
...
```

have to submit a valid Yubikey OTP. This is merely a measure to avoid database bloat from too many bogus requests. The API key also comes with an ID number.

The purpose of the API key is to sign/verify requests to/from the Yubico authentication Web service using the HMAC-SHA1 hashing algorithm. This is done because support for SSL is often spurious in the various environments in which the Web service client libraries have to function. Note that it is not strictly necessary to use SSL, because the token already is encrypted! However, as an added precaution, SSL should be used as a transport layer whenever it is available. In the PHP client library, for example, all you have to do is add an s to http where the authentication server URL is specified.

Adding Yubikey Authentication to Typo

Now that we have a solid understanding of the underlying technology, let's add Yubikey authentication to an existing application. I use Typo to blog. Typo is developed using Ruby on Rails, and you can check out its latest codebase via the project's public Subversion repository. Whether or not you like the structure RoR imposes on the developer, it works to our advantage in this case, because it makes it easy to locate the files we need to modify. Take a look at Figure 5 for a basic outline of the validation routine we will be implementing.

To start, let's drop the Ruby Web services client library, yubico.rb, into the project's lib directory. After adding the corresponding require command to the config/environments.rb file, we can be assured that the library will be available throughout the application.

Two groups of settings are necessary to configure Yubikey authentication. First, there are the site-wide settings, namely the API key and corresponding ID necessary to submit authentication requests to the Web service. There also is a switch for enabling or disabling Yubikey authentication on a blog-wide level. Typo stores these blog-specific settings by serializing them and persisting them to the blogs.settings column. Lucky for us, that means we don't have to make any changes to the database. However, we do need to amend the UI and data model used to store these settings within the application. Listing 1 shows how to add these three Yubikey configuration options to the respective HTML template in the admin user interface. Similarly, Listing 2 shows how to add those same settings to the model. That's all it takes for Rails to render a form to input those settings and store them in the database for each blog. Figure 6 shows the final result.

Second, there are two user-specific settings: Yubikey ID and Yubikey Required. The former is necessary to associate a Typo account with a user's unique public Yubikey ID; whereas the latter allows users to enable Yubikey authentication selectively for their accounts only. Now, let's make both

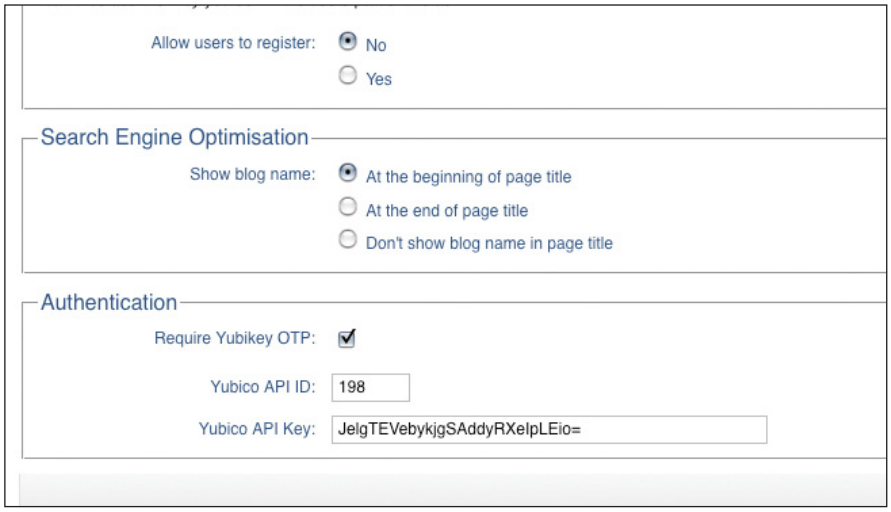


Figure 6. Typo: Blog-Wide Yubikey Settings UI

Listing 3. Typo: Account-Specific Yubikey Configuration Options HTML

```
filename: app/views/admin/users/_form.html.erb:

...
<li>
  <label class="float" for="user_notify_on_new_articles"><%=
      _("Send notification messages when new articles are posted")%>?
  </label>
  <%= check_box 'user', 'notify_on_new_articles' %>
</li>
<!-- new options for Yubikey authentication - start -->
<li>
  <label class="float" for="user_yubikey_required"><%=
      _("Yubikey Required")%>?
  </label>
  <%= check_box 'user', 'yubikey_required' %>
</li>
<li>
  <label class="float" for="user_yubikey_id"><%=
      _("Yubikey ID")%>:
  </label>
  <%= text_field 'user', 'yubikey_id' %>
</li>
<!-- new options for Yubikey authentication - end -->
</ul>
</fieldset>
<!--[eoform:user]-->
```

options available from the user's preference settings within the application's admin interface. To make the new options appear in the UI, I added a new section to the partial HTML template that renders the form for editing user options (Listing 3). Thanks to RoR's ActiveRecord support, we don't need to write any code to save these new options to the database; however, we do need to make sure that we add the correspondingly named fields to the user table to which all values on this screen are being persisted. In Rails, this is done by adding a database migration, which is nothing more than an abstract way of describing an incremental modification to the database. In our case, we

Listing 4. Typo: Yubikey Settings Database Migration

```
filename: db/migrate/071_add_yubikey_columns_to_user.rb:

class AddYubikeyColumnsToUser < ActiveRecord::Migration
  def self.up
    add_column :users, :yubikey_id, :string,
               :null => false, :default => ''
    add_column :users, :yubikey_required,
               :boolean, :null => false, :default => false
  end

  def self.down
    remove_column :users, :yubikey_id
    remove_column :users, :yubikey_required
  end
end
```

are adding the fields yubikey_id and yubikey_required to the user table by creating the migration shown in Listing 4. Now, all you need to do is run the rake utility from the command line and tell it to upgrade the database: `rake db:migrate`. The nice thing about Rails' migrations is that they are database-provider independent. The migration we created in Listing 4 can be used with any of the underlying databases that Typo supports. At the time of this writing, this includes MySQL, PostgreSQL and SQLite. Finally, you can admire the new settings in the account-specific options in Figure 7.

Additionally, there are projects for libraries to decrypt OTPs in C and Java, as well as an Open ID server and a personalization tool to allow you to program your own Yubikey.

Now that we have the setup all taken care of, we can focus on the actual authentication during login. First, let's add a Yubikey OTP input field to the login screen provided that Yubikey authentication is enabled for the whole blog. I have done this by modifying the partial template that renders the login form in Listing 5. Notice that we always have to show the Yubikey OTP field

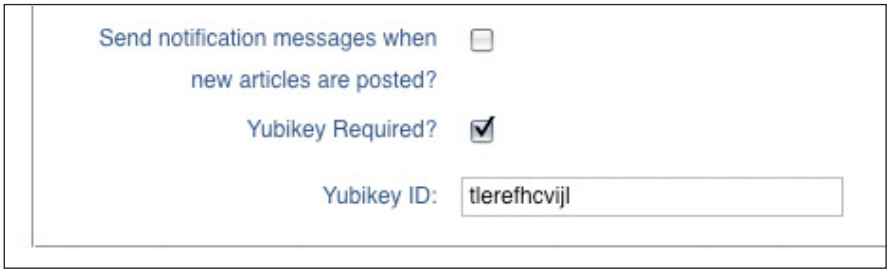


Figure 7. Typo: Account-Specific Yubikey Configuration Options UI

Listing 5. Typo: Modified Login Form HTML

```
filename: app/views/shared/_loginform.html.erb:

<% form_tag :action=> "login" do %>
<ul>
  <li>
    <label for="user_login"><%= _('Username')%>:</label>
    <input type="text" name="user_login" id="user_login" value=""/>
  </li>
  <li>
    <label for="user_password"><%= _('Password') %>:</label>
    <input type="password" name="user_password" id="user_password" />
  </li>
<!-- Yubikey authentication - start -->
<% if this_blog.yubikey_required %>
  <li>
    <label for="yubikey_otp"><%= _('Yubikey OTP') %>:</label>
    <input type="text" name="yubikey_otp" id="yubikey_otp" />
  </li>
<% end %>
<!-- Yubikey authentication - end -->
  <li class="r"><input type="submit" name="login"
      value= "<%= _('Login') %> &#187;"
      class="primary" id="submit" />
  </li>
</ul>
<p><%= link_to
      "&laquo; " + _('Back to ') + this_blog.blog_name,
      this_blog.base_url %></p>
<% end %>
```

during login, because until users supply their user names, we don't know whether Yubikey authentication is required for a particular user. Figure 8 shows the modified login screen.

When the login form is submitted, Rails routes it to the login method of the AccountsController class (Listing 6). This is where we add the logic to check whether we need to handle Yubikey authentication. After the existing code has verified the regular login and password, we now have an instantiated user object that can tell us whether Yubikey authentication is required for this user. If so, we invoke the static method authenticate_yubikey of the user object. Looking at Listing 7, we check that neither the Yubikey OTP from the login form nor the user's public Yubikey ID are blank. Moreover, by definition, the first 12 characters of the OTP have to match the public ID associated with the account. If everything is in order, we instantiate a Yubico object, which will handle the Web service authentication request for us. The method simply returns a boolean. True means the

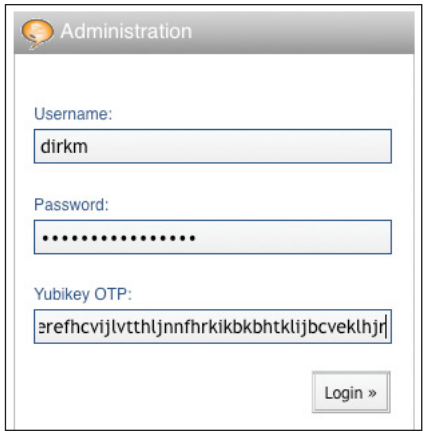


Figure 8. Typo: Modified Login Form UI

user was authenticated successfully. Conversely, false implies an invalid OTP or an attempt by an unauthorized user—possibly an attempt to hack into the account.

That's it! My Typo blog is now Yubikey-enabled. I will be submitting a patch to make these changes permanent by integrating them into the Typo codebase.

Implementation Variations

You might want to consider a few variations when implementing Yubikey authentication. First, you can choose to omit the user name, because the Yubikey token already includes a public ID that can be used to link to the user's account. This scheme works as long as you are not allowing users to associate a single Yubikey with multiple accounts.

Second, you can minimize modifications required to the UI of existing systems by including the Yubikey token in the password field. Because the OTP is of fixed length, it stands to reason that the remaining characters belong to the password. Also, as the Yubikey appends a newline character to the token, users would have to type their password first, followed by the OTP—rather than the other way around.

Third, you might want to consider making login a two-step process. First, prompt the user for the OTP and validate it. If the validation request is approved, prompt the user for the regular login and password. To see the advantage of this approach, consider the scenario in which user name, password and OTP

Listing 6. Typo: Yubikey Authentication Part 1

```
filename: app/controllers/accounts_controller.rb:

...
def login
  case request.method
    when :post
    self.current_user =
      User.authenticate(params[:user_login], params[:user_password])

    # check whether Yubikey authentication is required and perform
    # authentication
    if logged_in? &&
           (!this_blog.yubikey_required ||
            !self.current_user.yubikey_required ||
            self.current_user.authenticate_yubikey(
                this_blog,
                self.current_user.yubikey_id,
                params[:yubikey_otp]))
      session[:user_id] = self.current_user.id

      flash[:notice]  = _("Login successful")
      redirect_back_or_default :controller => "admin/dashboard",
                               :action => "index"
    else
      flash.now[:notice]  = _("Login unsuccessful")
      @login = params[:user_login]
    end
  end
end
...
```

Listing 7. Typo: Yubikey Authentication Part 2

```
filename: app/model/user.rb

...
  # Authenticate a user's Yubikey ID.
  #
  # Example:
  #   @user.authenticate_yubikey(this_blog, 'thcrefhcvijl',
  #   'thcrefhcvijldvlfugbhrghkibjigdbunhjlfnbtvfbc')
  #
  def authenticate_yubikey(this_blog,
                           yubikey_id = '', yubikey_otp = '')
    if (yubikey_id.empty? ||
        yubikey_otp.empty? ||
        !yubikey_otp[0, 12].eql?(yubikey_id))
      return false
    else
      begin
        yk = Yubico.new(this_blog.yubikey_api_id,
                        this_blog.yubikey_api_key)
        return yk.verify(yubikey_otp).eql?('OK')
      rescue
        return false
      end
    end
  end
...
```

are submitted simultaneously. If malicious parties are able to intercept the submission and prevent the OTP from being submitted to the validation server, they effectively have all three pieces of information they need to penetrate the system to which you are trying to authenticate. However, if you submit the OTP only during the first stop of the login process, malicious parties can intercept the token without gaining access to the system because they do not have the corresponding user name and password. To make you supply the user name and password, they need to let the OTP pass through and be validated, which also makes the OTP useless for subsequent uses. Thus, the attackers' task will be complicated significantly.

Yubikey in the Wild

On its Web site, Yubico maintains a growing list of applications and services that take advantage of the Yubikey. There is a plugin for WordPress, SSH integration, phpBB forum access and Windows login (commercial beta). As the above example of integrating the Yubikey into the Typo blog software's authentication routine shows, the process is fairly straightforward. Hopefully, this article inspires you to use this as a starting point to make your favorite piece of open-source software more secure by adding Yubikey authentication.■

Dirk Merkel is the CTO of Vivantech Inc. In his spare time, he likes to ruin perfectly good open-source projects by submitting unsolicited patches. He also writes about Web development. He lives in San Diego with his lovely wife and two wonderful daughters. Dirk can be reached at dmerkel@vivantech.com.

Resources

Yubico's Yubikey Page:
www.yubico.com/products/yubikey

Applications Supporting Yubikey:
yubico.com/products/apps

RoundCube Web-Based E-Mail Client:
www.roundcube.net

Typo Blogging Software:
www.typosphere.org

Did you know that RAM doesn't clear the moment it loses power? That it can persist for up to a few minutes if chilled? Learn about attack techniques that take advantage of these facts to uncover encryption keys and break disk encryption.

COLD BOOT ATTACK TOOLS for Linux

If you have used a computer for any reasonable length of time, you've learned about the difference between RAM storage and hard drive storage. Besides the fact that RAM is faster than hard drive storage, we also typically think that anything stored in RAM lasts only until the computer loses power, while data stored on a hard drive persists even when the computer is unplugged. Anyone who has lost power while working on a school assignment can attest to the temporary nature of RAM storage.

KYLE RANKIN

The Cold Boot Attack

It turns out that what we have learned about RAM isn't entirely true. On February 21, 2008, a paper titled "Lest We Remember: Cold Boot Attacks on Encryption Keys" was released. In this paper, the researchers describe their discoveries about RAM persistence and how they can be exploited. The researchers found that RAM isn't automatically erased when it no longer has power. Instead, RAM degrades over time, and even after a few seconds without power, you still can recover a significant amount of data. They also found that if you chill the RAM first, using liquid nitrogen or even a can of compressed air turned upside down, you can preserve the RAM state for more than 30 seconds up to minutes at a time—more than enough time to remove the RAM physically from a machine and place it in another computer.

By itself, although this discovery is surprising, what's most interesting are some of the implications if RAM contents can survive a reboot. It turns out that a number of common disk encryption tools for Windows, Mac and even Linux all store encryption keys in RAM. With this cold boot attack, if people lock their screens or even suspend their laptops, you could pull the power, grab the RAM contents and scrub it for any encryption keys. Essentially, you could compromise all of the common disk encryption techniques if you had a few minutes alone with a computer.

When I heard of this discovery, the first thing that came to my mind wasn't encryption, but forensics. I've written previously about forensics in *Linux Journal* [see "Introduction to Forensics" in the January 2008 issue], and in that article, I discuss the debate over how to respond initially when your server has been hacked. One school of thought favors instantly pulling the power on a compromised server. The idea is that you want to freeze the filesystem in place and don't want to risk that the attacker, or even the investigators for that matter, will destroy evidence. The other school of thought believes that pulling the power would destroy a lot of valuable data that exists only in RAM, so one should gather data from RAM first and then pull the power. With this cold boot attack, now you don't have to make that choice. If a server has been compromised, you can pull power first, and then reboot and grab the contents of RAM.

Cold Boot Attack Tools Released

In the paper, the researchers not only outlined the cold boot attack, they also described tools they had created to take advantage of this flaw. On July 16, 2008, the complete source code for these tools was released to the public at **citp.princeton.edu/memory/code**. In true UNIX style, each of the tools are small and single-purpose:

- RAM imaging tools: the first set of tools enables you to image a system's RAM. Although you potentially could boot off a rescue disk like Knoppix and then copy the memory, the rescue disk itself will overwrite a substantial amount of RAM. With the provided tools, you have a small executable that you can boot either from a USB disk or over the network via PXE. The USB executable dumps the entire contents of RAM to the USB disk and then powers off or reboots the host. The attacker then can take the USB disk to another computer and use a corresponding tool to dump the memory from the disk into a file. The PXE executable sets up the target for remote control, so the attacker then can dump the RAM over the network to the PXE server.
- Key-scanning tools: the second set of tools on the site can scan the RAM image you have created for encryption keys. The names of the tools are pretty self-explanatory. The aeskeyfind tool searches for AES keys, and the rsakeyfind tool searches for RSA keys.

Download and Build the Cold Boot Attack Tools

Since the source for all of these tools was released, you can download and use them yourself without too much setup. First, go to **citp.princeton.edu/memory/code**, and download the latest version of the bios_memimage tarball, or the efi_netboot tarball if you want to image a machine that boots with EFI. Then, unpack the tarball. For my examples in this article, I use the bios_memimage package.

The bios_memimage package contains a doc directory with good documentation on the project and how to build and use the source. The tools support both 32- and 64-bit environments. Although the 32-bit version technically will work on a 64-bit system, it can't address all the 64-bit environment's memory space, so you might not get a complete image. To build for a 32-bit environment, enter the bios_memimage directory and type `make`. To build for a 64-bit environment, enter the bios_memimage directory and type `make -f Makefile.64`.

Note: I noticed when I compiled the code on my environment, the build errored out with an undefined reference to __stack_chk_fail. This is due to GCC's new stack protection. As a workaround, edit the pxe/Makefile file and change the line that reads:

```
CFLAGS= -ffreestanding -Os -Wall -I../include -march=i386
```

to:

```
CFLAGS= -ffreestanding -Os -Wall -I../include
 ➥-march=i386 -fno-stack-protector
```

USB-Based Cold Boot Attacks

Once the code has compiled successfully, you are ready to install the tools. The procedure is different for the USB and PXE tools. For the USB tool, you need a USB drive that you are willing to erase and that is big enough to fit the RAM you want to dump. In the usb directory is a bootable image called scraper.bin. Connect your USB disk (in my example, /dev/sdb), and then use the dd tool as root to overwrite the beginning of the drive with the boot image:

```
$ sudo dd if=scraper.bin of=/dev/sdb
19+1 records in
19+1 records out
9792 bytes (9.8 kB) copied, 0.0101028 s, 969 kB/s
```

Now the disk is ready. Go to the machine you would like to image, connect the USB drive, and then force a CPU reset or pull and then restore the power quickly. Then, set the BIOS to boot from the USB key. This will vary depending on the computer. On some BIOSes, you will press F12 or some other key to see a list of boot options; others require you to enter the BIOS configuration to change the boot order. In any case, once

you boot from the USB key, the scraper tool immediately will start dumping the contents of RAM to the disk. Once it has completed, it will attempt an APM power-off or otherwise will reset the machine. Then you can unplug the USB drive and return to your machine.

You can use the provided usbdump tool under the directory of the same name to dump the RAM from the USB disk to your local drive. Simply specify the USB drive as an argument and then redirect the output to a file of your choice:

```
$ sudo ./usbdump /dev/sdb > memdump.img
recover segment0 [base: 0x0 size: 653312]
recover segment1 [base: 0x100000 size: 1062993920]
```

PXE-Based Cold Boot Attacks

The PXE-based scraper works somewhat differently from the USB-based scraper. First, if you don't already have a PXE server, you need to configure one. That process is out of the scope of this article, but I explained how to set up a PXE server in the article "PXE Magic" in the April 2008 issue of *Linux Journal*. Once you have a functional PXE server, copy the pxe/scraper binary to your tftp directory and change your pxelinux configuration so that it points to that file.

Next, connect the target system to the network (or if you set up the PXE server on a laptop, just connect the target system to the laptop via a crossover cable). Then, initiate a CPU reset or power off, and then immediately power on the target system. As with USB booting, different BIOSes have different ways to boot from PXE. On some BIOSes, you can press a function key, and others require that you change the boot order from the BIOS configuration.

Once the target machine gets a DHCP address and boots from the network, it will display a status message and then wait for the pxedump utility to connect. Unlike with the USB-based scraper, the PXE scraper doesn't automatically dump the memory over the network. Instead, you need to execute the pxedump binary found under the pxedump directory as follows:

```
$ ./pxedump target_machine_IP_address > memdump.img
```

Scan the Memory Dump

Once you have a dump from the target system's RAM, what can you do with it? Well, one of the primary things you can do is to scan the image for encryption keys. On the same page as the bios_memimage package, you will find tarballs for aeskeyfind and rsakeyfind utilities. To use these utilities, simply extract the source from the tarball and then run `make` within the source directory. Each source tree includes a README file that describes options with these utilities, but for basic scanning, just execute the aeskeyfind or rsakeyfind binary with the path to the memory dump as an argument. The tools will output any keys they find.

Unfortunately, there aren't a lot of other publicly available tools out yet that can reconstruct other useful information from a memory dump; however, you always can use the strings utility and grep to scan the image for keywords:

```
$ strings memdump.img | grep keyword
```

Cold Boot Attack Limitations

This attack can be very effective, particularly against laptops. That being said, there are a number of limitations to this attack. For one, the machine you attack must be powered on, suspended or hibernated, because the RAM will start to degrade once the machine is powered off. Second, some BIOSes and all systems with ECC RAM will scrub the RAM before it boots an OS. In those cases, you either would have to attempt to disable this scrubbing or chill the RAM and move it to a system that doesn't do any scrubbing.■

Kyle Rankin is a Senior Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *Knoppix Hacks* and *Ubuntu Hacks* for O'Reilly Media. He is currently the president of the North Bay Linux Users' Group.

Resources

Official Page for the Cold Boot Attack:
citp.princeton.edu/memory

Direct Link to the Research Paper:
citp.princeton.edu/pub/coldboot.pdf

Source Code for Cold Boot Attack Tools:
citp.princeton.edu/memory/code

Polywell Linux Solutions

More Choices, Excellent Service, Great Value!

Serving the Industry for More Than 20 Years

Netdisk 8000V
Quiet Performance NAS Storage

4TB $1,399
8TB $2,399
12TB $2,999

- Dual Gigabit LAN
- RAID-5, 0, 1, 10
- Hot Swap, Hot Spare
- Linux, Windows, Mac
- E-mail Notification
- Tower or Rackmount

The new generation of energy-efficient 45W AMD Athlon™ X2 dual-core processors with Cool'n'Quiet™ technology are designed to enable quiet performance.

AMD Athlon™ X2 64

Silent Eco Green PC

Based on the new 45W AMD® Athlon™ X2 dual-core processor is energy efficient, quiet and has plenty of power. starts at **$199**

Fanless Silent ITX PC

1G DDR2, 80GB Hard Drive starts at **$299**
AMD® processor, Low-profile Add-on Available

AMD Opteron™ 64

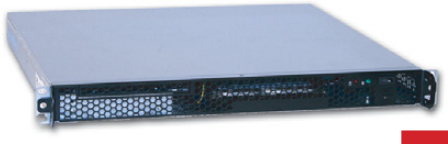
4U 24Bay Storage Server

AMD Opteron™ Processors
36TB RAID-6 128GB RAM, 4 x GigaLAN

AMD Phenom™ 64

Small 1U Server for Data Center ISP

AMD Phenom™ processor
4 to 8GB DDR2-800, 2 x 500GB RAID HD
Linux Server Starts at **$499**



Polywell OEM Services, Your Virtual Manufacturer

Prototype Development with Linux/FreeBSD Support
Small Scale to Mass Production Manufacturing
Fulfillment, Shipping and RMA Repairs

- 20 Years of Customer Satisfaction
- 5-Year Warranty, Industry's Longest
- First Class Customer Service

888.765.9686
linuxsales@polywell.com
www.polywell.com/us/Lx

Polywell Computers, Inc 1461 San Mateo Ave. South San Francisco, CA 94080 650.583.7222 Fax: 650.583.1974

POLYWELL

PAM

Securing Linux Boxes Everywhere

In a world without Windows, PAM guards the doors.

Federico Kereki

IF YOU ARE into British detective fiction and names like Sherlock Holmes, Sexton Blake, Mr. J. G. Reeder, Miss Marple, Hercule Poirot, Father Brown, Dr. John Evelyn Thorndyke and Lord Peter Wimsey mean anything to you, you also probably will recognize E. W. Hornung's (brother-in-law to Sir Arthur Conan Doyle, the creator of Sherlock Holmes) character: the white-glove thief, Raffles. In the "A Jubilee Present" short story, the thief is fascinated with an antique gold cup, displayed at the British Museum. Upon finding only one guard, Raffles questions him on the perceived lack of security and gets the confident answer, "You see, sir, it's early as yet; in a few minutes these here rooms will fill up; and there's safety in numbers, as they say." With Linux, rather than security by numbers (which eventually is no good for the poor guard; see Resources for a link to the complete story), security is managed by Pluggable Authentication Modules (PAM). In this article, we study PAM's features, configuration and usage.

Let's start at the beginning and consider how an application authenticates a user. Without a common, basic mechanism, each application would need to be programmed with particular authentication logic, such as checking the /etc/passwd for a valid user and password. But, what if you have several different applications that need authentication? Do you include the same specific logic in all of them? And, what if your security requirements vary? Would you then have to modify and recompile all those applications? This wouldn't be a practical method and surely would become a vulnerability. How would you be sure that all applications were duly updated and correctly implemented your new specifications?

The PAM Project provides a solution by adding an extra layer. Programs that need authentication use a standard library or API (Application Programming Interface), and system administrators can configure what checks will be done by that library separately. (Checks are implemented via independent modules; you even can program your own modules.) This way, you can change your security checks dynamically, and all utilities will follow your new rules automatically. In other words, you can modify the authentication mechanism used by any PAM-aware application, without ever touching the application itself. For programmers, this also is a good thing, because they need not be concerned with the mechanisms that will be used. Simply by using the PAM libraries, whenever the application is run, the appropriate checks will be made (Figure 1).

The PAM library breaks down authentication in four areas or groups (Table 1). Note that all applications won't always require the four previous actions. For example, the passwd command will require only the last group. (Quick tip: how can you learn whether an application uses PAM? Use `ldd` to print the shared libraries required by the program, and check for libpam.so; see Listing 1 for an example.)

Configuring PAM

For each service (such as login or SSH), you must define which checks will be done for each group. That list of actions is called a stack. Depending on the results of the actions in each stack, users will succeed or fail, and whatever they attempted to do will be allowed or rejected. You can specify each action in the stack for each service using a specific file at /etc/pam.d (the more current method) or by editing the single, catch-all file /etc/pam.conf (the older method); in this article, we use the former method.

Each stack is built out of modules, executed sequentially in the given order. For each module, you can specify whether it's necessary (failure automatically denies access), sufficient (success automatically grants access) or optative (allows for alternative checks). Table 2 shows the actual control flags. The file for each service consists of a list of rules, each on its own line. (Longer lines can be split by ending with a \, but this is seldom required.) Lines that start with a hash character (#) are considered to be comments and, thus, are ignored. Each rule contains three fields: the context area (Table 1), the control flag (Table 2) and the module that will be run, along with possible (optional) extra parameters. Thus, the specification for the PAM checks for login would be found in the /etc/pam.d/login file.

The control flag field actually can be more complicated, but I won't cover all the details here. See Resources if you are interested. Also, you can use include, as in `auth include common-account`, which means to include rules from other files.

There is a special, catchall service called other, that is used for services

NOTE:

Remember that playing with configuration files can be dangerous to your health! A particularly nasty thing to do is remove all configuration files accidentally, because then you won't be able to log back in again. Make sure to back up all files before you start experimenting and have a live CD available just in case.

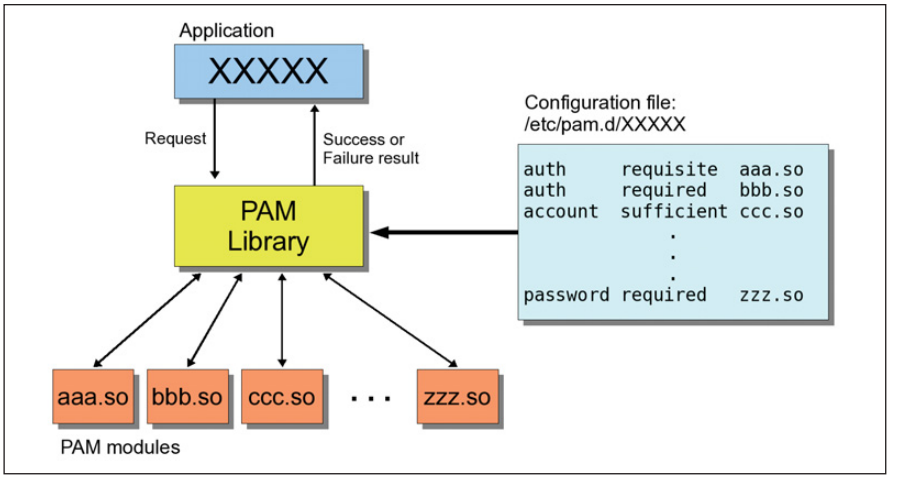


Figure 1. Whenever an application does an authentication request, the PAM library executes whatever modules are specified in the configuration file and decides whether to approve (success) or reject (failure) the request.

Listing 1. To learn whether a program uses PAM, use ldd and look for the libpam.so library. You need to provide the full path to the program; use whereis if you don't know it.

```
$ whereis login
login: /bin/login /etc/login.defs /usr/share/man/man3/login.3.gz
 ➥/usr/share/man/man1/login.1.gz
$ ldd /bin/login
        linux-gate.so.1 =>  (0xffffe000)
        libpam_misc.so.0 => /lib/libpam_misc.so.0 (0xb7eff000)
        libpam.so.0 => /lib/libpam.so.0 (0xb7ef3000)
        libaudit.so.0 => /lib/libaudit.so.0 (0xb7edf000)
        libc.so.6 => /lib/libc.so.6 (0xb7dac000)
        libdl.so.2 => /lib/libdl.so.2 (0xb7da8000)
        /lib/ld-linux.so.2 (0xb7f25000)
```

Table 1. PAM has four groups of checks, organized as stacks. The groups that will be used depend on what the user requires.

auth	Related to user identification, such as when a user needs to enter a password. This is usually the first set of checks.
account	Has to do with user account management, including checking whether a password has expired or whether there are time-access restrictions. Once users have been identified by the authentication modules, the account modules will determine whether they can be granted access.
session	Deals with connection management, with actions such as logging entries or activities, or doing some cleanup actions after the session ends.
password	Includes functions such as updating users' passwords.

Table 2. For each stack, modules are executed in sequence, depending on their control flags. You must specify whether the corresponding check is mandatory, optional and so on.

required	This module must end successfully. If it doesn't, the overall result will be failure. If all modules are labeled as required, any single failure will deny authentication, although the other modules in the stack will be tried anyway.
requisite	Works like required, but in case of failure, returns immediately, without going through the rest of the stack.
sufficient	If this module ends successfully, other modules will be skipped, and the overall result will be successful.
optional	If this module fails, the overall result will depend upon the other modules. If there are no required or sufficient modules, at least one optional module should end successfully to allow authentication.

Listing 2. A safe "other" definition forbids all generic access in absence of specific rules. The pam_deny.so module always returns failure, so all access attempts will be rejected, and pam_warn.so sends a warning to the sysadmin.

```
#
# default; deny all accesses
#
auth     required        pam_deny.so
auth     required        pam_warn.so
account required        pam_deny.so
password         required         pam_deny.so
password         required         pam_warn.so
session required         pam_deny.so
```

Listing 3. A PAM definition, equivalent to the standard UNIX security rules. Note: on some distributions, you might need to use pam_unix.so instead.

```
#
# standard UNIX minimalistic rules
#
auth     required         pam_unix2.so
account required         pam_unix2.so
password         required         pam_unix2.so
session required         pam_unix2.so
```

Listing 4. The /etc/pam.d/sshd specifies security rules for SSH connections. The pam_access.so module was added to the standard configuration to provide further checks.

```
auth     required         pam_unix2.so
auth     required         pam_nologin.so
account required         pam_unix2.so
account required         pam_access.so
session required         pam_limits.so
session required         pam_unix2.so
session optional         pam_umask.so
password         requisite        pam_pwcheck.so cracklib
password         required         pam_unix2.so use_authtok
```

Listing 5. The /etc/security/access.conf is used by pam_access.so to decide which users are allowed to log in and from which IPs. In this case, everybody from the local network can log in, but only remoteKereki is allowed external access.

```
+ : ALL : 192.168.
+ : remoteKereki : ALL
- : ALL : ALL
```

without specific rules. A good start from a security point of view would be creating /etc/pam.d/other, as shown in Listing 2. All attempts are denied, and a warning is sent to the administrator. If you want to be more forgiving, substitute pam_unix2.so for pam_deny.so, and then the standard Linux authentication method will be used, although a warning will still be sent (Listing 3). If you don't care about security, substitute pam_permit.so instead, which allows entry to everybody, but don't say I didn't warn you.

Finally, give the files in /etc/pam.d a quick once-over. If you find configuration files for applications you don't use, simply rename the files, so PAM will fall back to your "other" configuration. Should you discover later that you really needed the application, change the configuration file back to its original name, and everything will be okay again.

Secure Remote Access

To get a handle on all this, let's consider an actual application. I wanted to be able to access my machine remotely with SSH, but I didn't want to allow any other users (Listing 4). So, I configured my /etc/pam.d/sshd file. See the Modules, Modules Everywhere sidebar for more details on these and other modules. Here are some of the modules I used:

- pam_unix2.so: provides traditional password, rights, session and password-changing methods, in the classic UNIX way.
- pam_nologin.so: disallows login if the file /etc/nologin exists.
- pam_access.so: implements extra rules for access control (more later in this article on how I used this).
- pam_limits.so: enforces limits for users or groups according to the file /etc/security/limits.conf.
- pam_umask.so: sets the file mode creation mask for the current environment (do `info umask` for more information).
- pam_pwcheck: enforces password-strength checks (more details on further uses of this module later in this article).

If you check your own /etc/pam.d/sshd file, it probably will look like this, except for the pam_access module, which is the interesting part. This module implements added security controls based on the /etc/security/access.conf file. I edited it in order to specify who could access my machine (Listing 5). The first line means that anybody

Listing 6. The password section of the /etc/pam.d/passwd file that enforces good practices for new passwords.

```
#
# retry=3 allows three tries for a new password
# minlen=10 requires at least ten characters
# ucredit=-1 requires at least one uppercase character
# lcredit=0 accepts any number of lowercase characters
# dcredit=-2 requires at least two digits
# ocredit=-1 requires at least one non-alphabetic symbol
#
password required pam_cracklib.so retry=3 minlen=10 \
     ucredit=-1 lcredit=0 dcredit=-2 ocredit=-1
#
# As pam_cracklib only checks passwords, but doesn't store
# them, we require the standard pam_unix module for this.
# The use_authtok parameter ensures pam_unix won't ask for a
# password by itself, but rather will use the one provided by
# pam_cracklib.
#
password  required pam_unix.so use_authtok nullok
```

(ALL) can log in to my machine from within the internal network at home. The second line allows the remoteKereki user to access my machine from anywhere in the world, and the final line is a catchall that disables access to anybody not included specifically in these lines. I created the remoteKereki user with minimum rights to allow myself entry to the machine, and then I execute `su` and work as myself or even as root, if needed. If people guess the correct password for remoteKereki, it won't help them much, because attackers still will have to guess the password for the other, more useful, users. As it is, it provides an extra barrier before intruders can do serious damage.

I had to modify /etc/ssh/sshd_config by adding a line `UsePAM yes`, so sshd would use the PAM configuration. I had to restart SSH with `/etc/init.d/sshd restart` so the configuration would be used. For even more secure connections, you also could change the SSH standard port (22) to a different value, forbid root remote logins and limit retries to hinder brute-force attacks, but those topics are beyond the scope of this article. Do `man ssh_config` for more details.

Requiring Good Passwords

Left on their own, most users will (trustingly and unknowingly) use easily guessable and never-changed passwords, simplifying the job for intruders. With PAM, you can enforce several good practices for password management by using the password stack and the pam_pwcheck.so module. This module does several checks on the strength of your password:

- Is the new password too short?
- Is the new password too similar to the old one?
- Is the new password merely the old password, reversed or rotated (for example, safe123 and 123safe)?
- Is the new password the same as the old one, with only case changes (such as sEcReT and SEcrET)?

Modules, Modules Everywhere

Your system's security depends on the modules you use. Modules are stored in /lib/security or /lib64/security (for 64-bit systems), but some distributions do not follow this standard. For example, you might find the modules in /usr/lib/security. You can write your own modules if you want (see Resources), but for starters, you probably will be able to manage with the standard ones. The following is a list of the more common modules. For more information, use the `man` command. Also note that there is no standard list of modules, and each distribution may include more modules or variations on the modules below.

pam_access: allows or refuses access, based on IPs, login names, host or domain names and so forth. By default, access rules are specified in /etc/security/access.conf. Whenever a user logs in, the access rules are scanned in order for the first match, and permission is granted or denied accordingly. See also pam_time for further restrictions.

pam_cracklib and **pam_pwcheck:** provide password strength-checking and disallow repeated, too simple and easily guessed possibilities. Users are prompted for a password, and if it passes the predefined rules and is considered strong, users are prompted again as a check.

pam_deny: simply denies access. It can be used to block users as a default rule. See also pam_permit.

pam_echo: displays a (configurable) text message to the user. See also pam_motd.

pam_env: allows setting or unsetting environment variables. The default rules are taken from /etc/security/pam_env.conf.

pam_exec: calls an external command.

pam_lastlog: displays the date and time of the last login.

pam_limits: sets limits on the system resources that a user might require. The default limits are taken from /etc/security/limits.conf.

pam_listfile: allows or denies services based on a file. For example, you could limit FTP access to users in the file /etc/ftpusers_ok by including the line `auth required pam_listfile.so item=user sense=allow file=/etc/ftpusers_ok onerr=fail` in the /etc/pam.d/ftpd file. See also pam_nologin.

pam_mail: informs users whether they have mail.

pam_mkhomedir: creates a user home directory, if it doesn't exist on the local machine. This allows you to use central authentication (NIS or LDAP, for example) and create user directories only when needed.

pam_motd: displays the "message of the day" file to users. See also pam_echo.

pam_nologin: disallows logins when /etc/nologin exists.

pam_permit: allows entry without checks—quite unsafe! See also pam_deny.

pam_rootok: allows access for the root user without further checks. This typically is used in /etc/pam.d/su to let root act as another user without entering a password. The file should contain the following lines (regarding the second line, see pam_wheel):

```
auth  sufficient   pam_rootok.so
auth  required     pam_wheel.so
auth  required     pam_unix.so
```

pam_succeed_if: tests for account characteristics, such as belonging to a certain group, having a certain UID and so on.

pam_time: restricts access to services depending on the day of the week and time of the day. The default rules are taken from /etc/security/time.conf. Note, however, that only the login time is enforced. There's no way to force the user to log out afterward.

pam_umask: sets the file mode creation mask.

pam_unix or **pam_unix2:** classic UNIX-style authentication, based on the /etc/passwd and /etc/shadow files. See also pam_userdb.

pam_userdb: authenticates against a database. See also pam_unix.

pam_warn: logs the service, terminal, user and more data to the system log. The module can be used anywhere, because it won't affect the authentication process.

pam_wheel: allows root access only to members of group wheel. This frequently is used for `su`, so only selected users can use it. See the pam_rootok entry for an example.

- Was the new password already used before? (Old passwords are stored in the /etc/security/opasswd file.)

You can add several parameters to the module (do `man pam_pwcheck` for complete documentation) for extra rules, such as:

- minlen=aNumber: specifies the minimum length (by default, five characters) for the new password. If you set it to zero, all password lengths are accepted.
- cracklib=pathToDictionaries: allows use of the cracklib library for password checks. If the new password is in a dictionary, a simple brute-force attack quickly will guess it.
- tries=aNumber: sets how many attempts to allow, if previous attempts were rejected because they were too easy.
- remember=aNumber: defines how many previous passwords will be remembered.

Another module provides similar functionality, pam_cracklib.so, but it has some different parameters. For example, you might specify how many characters must differ between your old and new password and whether you want to include digits, uppercase, lowercase and nonalphabetic characters. Do `man pam_cracklib` for more information.

Conclusion

There might be security in numbers (as the poor British Museum guard thought when he tried to deter Raffles from stealing the cup), but for Linux, PAM is the way to go. Without even resorting to rolling out your own modules, you can add plenty of flexibility to your security by setting up a few configuration files and rest assured that those rules will be obeyed globally.■

Federico Kereki is a Uruguayan Systems Engineer, with more than 20 years' experience teaching at universities, doing development and consulting work, and writing articles and course material. He has been using Linux for many years now, having installed it at several different companies. He is particularly interested in the better security and performance of Linux boxes.

Resources

"A Jubilee Present" by E. W. Hornung: **hornung.thefreelibrary.com/Raffles-Further-Adventures-Of-The-Amateur-Cracksman/2-1**

Official PAM Documentation: **www.kernel.org/pub/linux/libs/pam**

Configuration File Details: **www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/sag-configuration-file.html**

Commonly Available PAM Modules: **www.kernel.org/pub/linux/libs/pam/Linux-PAM-html/sag-module-reference.html**

TESTING THE LOCKS: Validating Security in a Linux Environment

Is your security worth its salt? Try this assessment strategy to find out.

JERAMIAH BOWLING

Many of you think you have a secure environment. You follow best practices. You check your logs regularly. Then, something gets through and although it may not wreak havoc, you wonder how it happened. A lot of shops practice passive security by putting security measures in place and assuming they work based on logs, dashboards or other output. This practice is inadequate for today's security landscape. Administrators must take an active approach to security to combat threats effectively. Active security can be as simple as verifying a password policy or as complex as running a full-blown penetration test. Whatever approach you choose, it always is a good idea to test the locks periodically with a security assessment to make sure they work. The locks are items such as the operating systems, network, applications and most important, security policies that exist in your environment. With regular security assessments, you can gain confidence that your security measures are keeping the bad guys out.

This article covers a security assessment in four parts. The sections are organized in reverse order of what an actual attack might look like. By the fourth section, I bring everything together and explain how such an attack might occur. I recommended that before proceeding with any of the following tests, you get the approval of upper management or the owner of the network and/or systems you will be testing. To minimize further any risk to a production network/system, the following tests should be performed after production hours if possible.

To assist in this assessment, I use a prebuilt VMware virtual machine (VM) with the BackTrack distribution on it (available from **remote-exploit.org**). BackTrack is a comprehensive security auditing and testing platform with many tools preconfigured and ready to use upon the first boot. All the scripts and applications presented here should be run as root. Only the custom script in the first section should be run locally on a target machine. All other tools should be run from the BackTrack VM.

1. Think Globally, Act Locally

Let's begin by checking the locks at the local level. The included script (Listing 1) profiles basic settings to identify and weed out common misconfigurations. It is by no means a catchall to validate all of your security measures. The script has been tested on Red Hat- and Debian-based systems, and as such, output may vary from system to system. You also may need to customize the script for your own systems to ensure functionality. All output is placed in the /tmp/seccheck/hostname directory, where hostname is your locally defined hostname.

Rather than go line by line, let's look at the output of the script. The first prompt identifies the base distribution and checks for needed patches and then outputs this information to /tmp/seccheck/hostname/patchcheck.txt. After patch-checking, the main output file is created as /tmp/seccheck/hostname/secoutput.txt. The first section of this file lists the local services that run at startup. With this information, you can view and disable any unnecessary services. This section is followed by a listing of failed authentication messages along with the results of the lastb command (Figure 1). From these two sections, you quickly can determine whether the machine has been accessed by an unauthorized user.

Next, the script checks whether the inetd dæmon is in use. Most modern distributions no longer use the inetd super server,

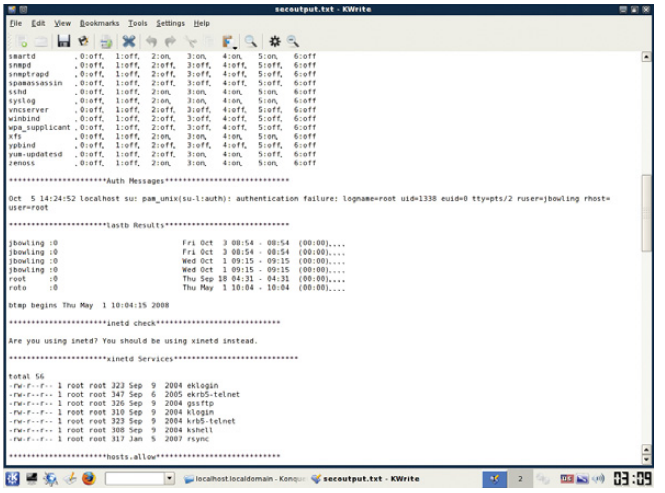


Figure 1. Some Output from the Local Script

but some legacy systems still do. If possible, you should convert those servers/services to xinetd. xinetd-enabled services are listed in the section that follows. Both super-servers can provide host-based access control to specific services using TCP Wrappers. The access controls for TCP Wrappers are stored in

Listing 1. This script checks some common misconfigurations.

```
#!/bin/bash

mycompname=$(hostname)
mydir=/tmp/seccheck/$mycompname
myoutput=$mydir/secoutput.txt
mkdir -p $mydir

sl()
{
  SECTION=$1
  echo >> $myoutput
  echo **********$SECTION********** >> $myoutput
  echo >> $myoutput
}

echo ^^^^^^^^^^ START OF OUTPUT ^^^^^^^^^^ > $myoutput

echo -n Is this a Red Hat \(r\) or a Debian based system \(d\)?
read REPLY

case "$REPLY" in
  'r')
    yum list updates > $mydir/patchcheck.txt
    sl "Service Runlevels";chkconfig --list >> $myoutput
    sl "Auth Messages";cat /var/log/secure|grep failure >> $myoutput
    ;;
  'd')
    apt-get update
    apt-get -qs upgrade > $mydir/patchcheck.txt
    sl "Startup Services";ls -l /etc/rc2.d >> $myoutput
    sl "Auth Messages";cat /var/log/auth.log|grep failure >> $myoutput
    ;;
esac

sl "lastb Results";lastb >> $myoutput
sl "inetd check"; file -f /etc/inetd.conf && \
    echo "Are you using inetd? You should be using xinetd instead." \
        >> $myoutput
sl "xinetd Services";ls -l /etc/xinetd.d >> $myoutput
sl "hosts.allow";cat /etc/hosts.allow |grep -v "#" >> $myoutput
sl "hosts.deny";cat /etc/hosts.deny |grep -v "#"  >> $myoutput
sl "iptables output";iptables --list >> $myoutput
sl "SUID Files";find / -perm -4000 -print >> $myoutput
sl "SGID Folders";find / -perm -2000 -print >> $myoutput
sl "SUDoers";cat /etc/sudoers|grep "="|grep -v "#" >> $myoutput

echo -n "Do you want to capture Password Files"
echo -n " for an offline Password Check (y or n?)"?
read REPLY2

if [ $REPLY2 = "y" ]; then
     cp /etc/passwd /tmp/seccheck/$mycompname
     cp /etc/shadow /tmp/seccheck/$mycompname
     echo Your Password and Shadow folders have been copied to
/tmp/secheck/$mycompname
else exit
fi

echo vvvvvvvvvv END OF OUTPUT vvvvvvvvvv >> $myoutput
```

the hosts.allow and hosts.deny files. The contents of these files are output after the xinetd section. If you use TCP Wrappers, there should be an entry in your hosts.deny that reads `ALL:ALL` to deny hosts that aren't allowed access explicitly. Local firewall (if used) rules are listed next.

Next, the script lists any SUID/SGID files and directories found on the machine. These files should be identified and their access verified, as they often are taken advantage of by rootkits. After that, the script concatenates a listing of the /etc/sudoers file. Users and groups found in the sudoers file can run as a super user (root) or any other user defined in the file. You should take stock of these users and verify they need sudo access.

Other good utilities/commands that could be added to this script, but have been omitted due to space considerations, are ps, top, mount, route, history, find / -perm 777 and testparm (Samba). If you use SELinux, you can run the `getsebool -a` command for confirmation of policy enforcement.

At the end of the script, you are prompted to copy the machine's local password and shadow files to the /tmp/seccheck directory, so you can transport them to the VM and perform a brute-force crack using John the Ripper later. After the script has completed, copy or burn the /tmp/seccheck directory to removable media for analysis on the BackTrack VM. Boot the VM, and log in with root and use "toor" as the password. After logging in, type `startx` to launch KDE. Copy the seccheck folder containing the password and shadow files from the removable media to the VM.

With the files local to the VM, let's run a brute-force password crack to test our password policies. Brute-forcing can be time consuming. You can speed the operation with the use of word lists, some of which are available from the john Web site. To start the crack with a basic brute-force, open a terminal on the VM and run the following command:

```
/usr/local/john/unshadow /pathtopasswdfile/passwd
➥/pathtoshadowfile/shadow > password.txt
```

This command combines the two files into the password.txt into a traditional UNIX-style password file. Next, run the following command from the terminal in your VM:

```
john password.txt
```

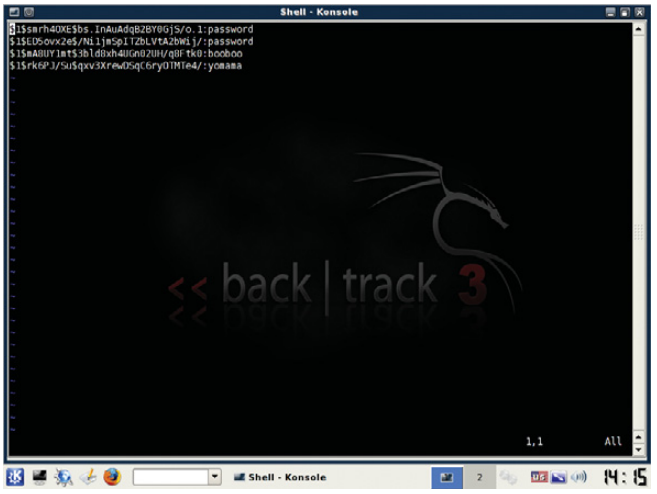


Figure 2. Hashes and Their Plain-Text Passwords Cracked by john on BackTrack

john will output its results to the terminal and also write to /usr/local/john/john.pot (Figure 2). One really nice feature of john is the ability to restart a terminated crack. If you need to terminate john for any reason, use Ctrl-C to end it. To resume it, type:

```
john --restore
```

Within a few minutes, you should see any simple passwords displayed. More complex passwords will take longer, based on various factors, such as complexity, system performance and the use of word lists.

Regardless of when you run john, you should review the secoutput.txt file thoroughly, document its findings and remedy any that fall short of our defined security policies.

2. Communication Is the Key

The second set of locks to validate is on your network. Any comprehensive security assessment must include validation of your network's correct operation. There is no better way to validate this than by simple observation. The first tool to use for this is the Wireshark network protocol analyzer. Wireshark puts your network card in promiscuous mode and captures any traffic broadcast on your local network segment. It may be necessary to take samples on different parts of your network or use span ports to get a good representation of normal traffic.

To start the program, open a terminal inside the VM and type `wireshark`. Once open, click on the Capture menu and then on Interfaces. On the Interface options window, click Start next to eth0 to start the capture (Figure 3). If you use something other than the BackTrack VM to run Wireshark, you might select a different interface. Click on the Capture menu again, and then click Stop to end the capture. When finished, save the capture to a file. I recommend that you take captures of no less than five minutes at random times during the day. The capture files will be big (longer capture = bigger file) if you have a busy network, but in my experience, five minutes is enough for most small-to-medium networks. Scan the capture files to identify unusual traffic, and validate any network-level policies you may have in place. For example, many networked printers, by default, broadcast NetBIOS for discovery on Windows networks, but you may not allow NetBIOS traffic on your network. Captures also can help find rogue-user PCs or VMs running without approval. Many people are surprised the first time they run a capture. The shortcoming of captures is the time required to analyze them. That is where our second network tool, Snort, comes in.

Snort is many things, but traditionally it's used as an intrusion-detection system (IDS). An IDS patterns network traffic against a database of known attack signatures to alert administrators to potential intrusions. Unlike Wireshark, Snort aggregates and analyzes the data it collects providing a thousand-foot view of the network. When using Snort, you should be aware of two things: IDSes are sensitive to false positives, and they do not alert on normal traffic. Snort is useful as an assessment tool, because it can tell you whether there are any major problems on your

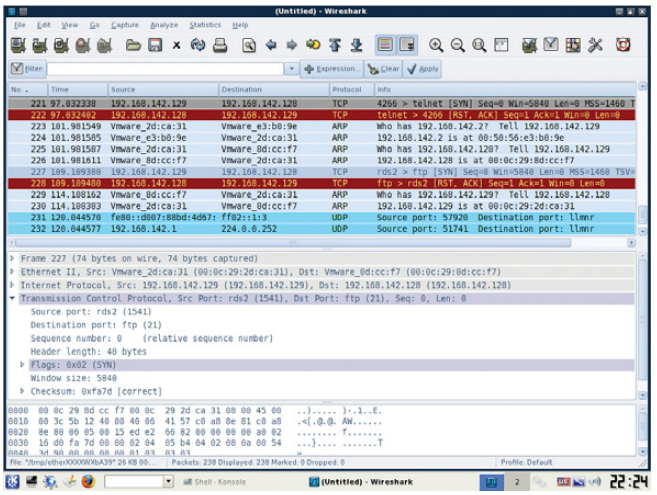


Figure 3. Wireshark analyzes all the way to the packet level.

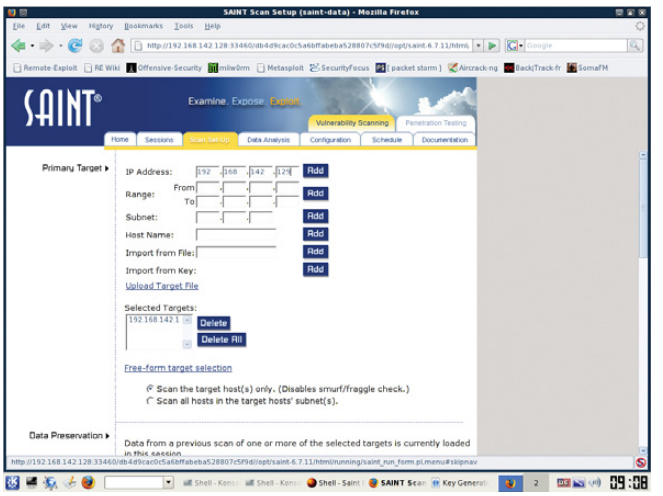


Figure 5. Adding Hosts into SAINT

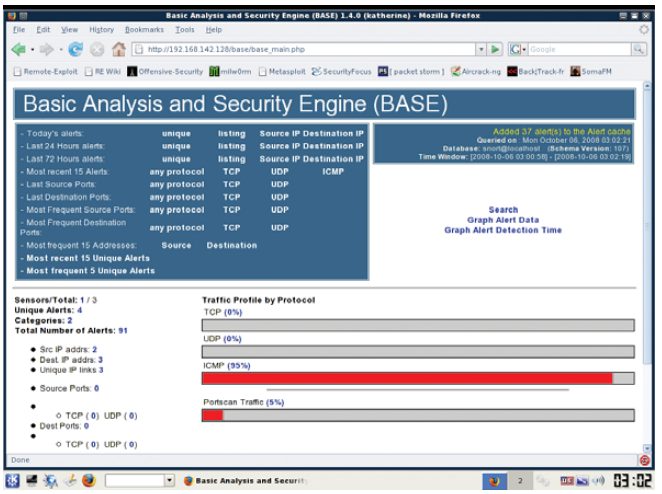


Figure 4. BASE makes Snort so much easier.

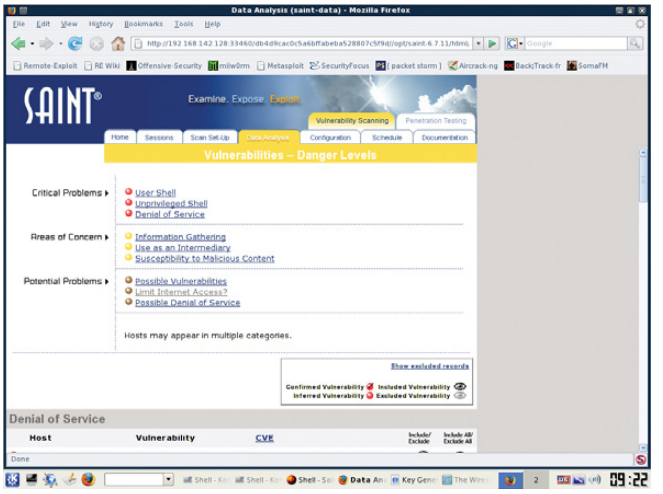


Figure 6. Results from a SAINT Scan

network in a short amount of time.

The BackTrack team conveniently has packaged Snort with the BASE Web front end in the distribution. From the KDE menu, select Services→Snort→Setup and Initialize Snort. You will be prompted by the setup script to enter root and Snort user passwords for MySQL in order to create the needed tables. At the end of the script, open a Web browser and enter http://youripaddress/base/base_db_setup.ph, and on the page that loads, click on the Create Base AG button. Now, click on the Main Page link (Figure 4) to access alert information. Unlike Wireshark, Snort should be run over a longer period of time (more than 24 hours in most cases) to provide a good sampling of network data.

3. Finding the Chink in the Armor: Vulnerability and Application Scanners

The third set of locks to test is found in the operating systems and applications on your network or, more specifically, in the vulnerabilities that exist on them. A reasonable approach to finding these vulnerabilities is to perform one or more broad vulnerability scans across the network, followed by any application-specific scans for our critical apps. Let's use the Security Administrator's Integrated Network Tool (SAINT) as our primary scanner.

SAINT normally allows only two IP addresses for scanning for 15 days, but BackTrack users can use up to ten IP addresses for up to a year by using the registration page found under the KDE menu: BackTrack→Vulnerability Identification→SAINT Exploit→SAINT Exploit License. From this Web page, click the Get License button at the bottom of the page and provide the necessary information on the registration page. Proceed with registration, and generate a key for use with the scanner. Once the key has been entered on the VM, launch SAINT from the same KDE folder as the License link, but click on the SAINT link instead. This launches the Web front end. Click the Scan Set-Up tab. Enter the IP addresses or range you want to scan (Figure 5). Under the Scanning Level section, check off Exhaustive and Full Port Scan. In the Firewall section, select No Firewall Support. You can play with any of these options to tailor the scans to your needs. Click Scan Now at the bottom of the page when finished. The results are displayed when the scan is finished (Figure

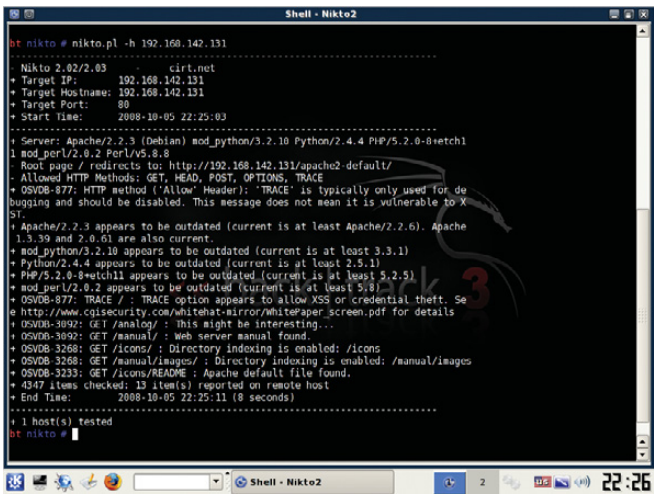


Figure 7. Nikto Scanning a Web Server

6). You should review and document the scan results, and wherever possible, remediate discovered vulnerabilities.

This broad scan with SAINT should be followed up with more specific scans against your most valuable (and therefore juicier targets) machines. As an example, let's scan a Web server using another tool found on BackTrack, Nikto. Nikto is a mature, simple scanning tool and an excellent resource for locking down a Web server. Assuming you have a Web server in your environment, launch a Nikto shell from the VM under the KDE menu BackTrack→Penetration→All→Nikto2, and from the resulting shell, type:

```
nikto.pl -h yourwebserveripaddresshere
```

On the Web, Articles Talk!

Every couple weeks over at LinuxJournal.com, our Gadget Guy Shawn Powers posts a video. They are fun, silly, quirky and sometimes even useful. So, whether he's reviewing a new product or showing how to use some Linux software, be sure to swing over to the Web site and check out the latest video: **www.linuxjournal.com/video**.

We'll see you there, or more precisely, vice versa!

As you can see, the output is straightforward and can be redirected to a file easily for later analysis (Figure 7). As with SAINT, you should follow up this scan by documenting the results and fixing any discovered issues.

4. Casing the Joint

The last lock to test is, in many cases, the first entrance into your network, the perimeter. Let's test it by placing our VM outside the network and then performing a network map against our publicly facing IP address(es) to verify that only allowed services are allowed in or out of the network. We use the time-tested Nmap application for this role.

Although Nmap is on the BackTrack VM, you need to update to the latest version to use the handy new topology tab of the zenmap front-end GUI. Download Nmap from the project's site, and install on the VM with the usual `./configure`, `make`, `make install` sequence. Type the command `zenmap` from a terminal to bring up the GUI. Enter a host, host range or network as the target, select Regular Scan from the Profile drop-down list and click on Scan. This performs a cursory

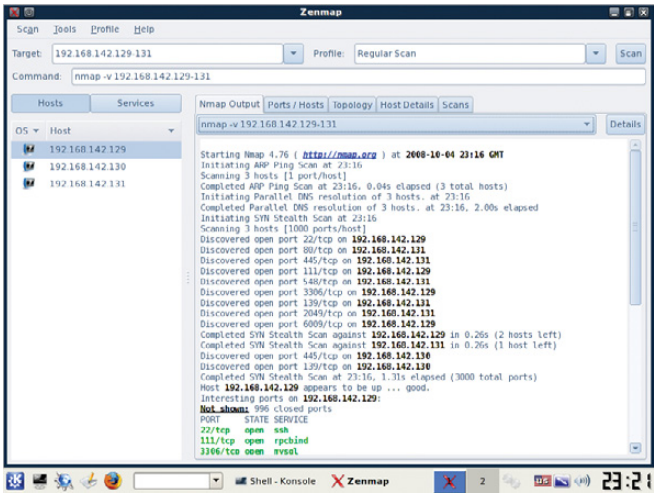


Figure 8. Nmap Results from a Regular Scan

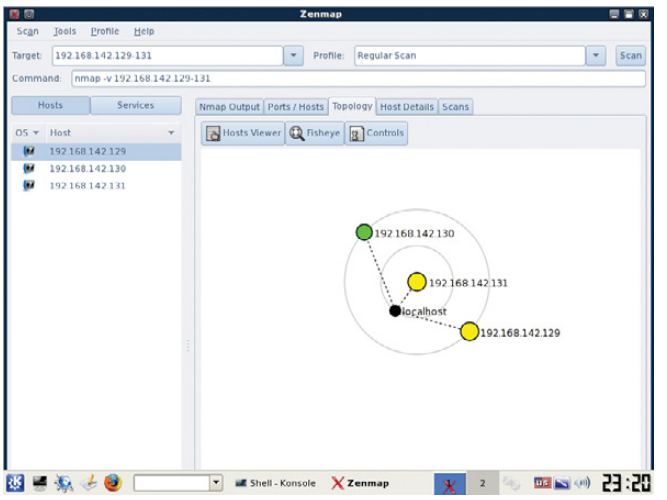


Figure 9. The Topology Tab of zenmap Visualizes a Map

scan of the host/networks and identifies open ports and other available information about the host, such as OS and app versions (Figures 8 and 9). Be patient; this process may take a while. Use Nmap's results to verify that only allowed hosts and services are accessible from the outside.

Let the Battle Begin

After running Nmap, we can start to envision how an attack against our network might take place. Assume we can glean our network's external IPs from public DNS or whois records. With this information, we run a network map against those IP addresses and identify host OS and application versions. With map results in hand, we scan said hosts for vulnerabilities as discussed in section 3 of this article. If we are lucky, we find one and run an exploit against it to take control of the box. If all we wanted was to own the box, mission accomplished. But, if we wanted to own other hosts or the network, we might begin a new map from the inside or sniff with a tool like Wireshark from the owned box. If we passively sniff traffic instead of map, we are less likely to set off any IDS alarms. At that point, we notice SSH traffic to a particular machine, so we attempt to gain a remote shell against it. Hopefully, there aren't any glaring openings in our local configuration, as we checked for in section 1, or we might lose another box or boxes.

Although this is not a standard blueprint for attack by any means, it is a possible avenue for attack. There are too many methods, techniques, hacks, cracks and attacks to document at length here. By performing regular assessments like the one shown in this article, we can lower the risk of attack, but not eliminate it. Unfortunately, it is a lot harder to play defense than offense. The bad guys do not focus on one aspect of security (or insecurity), and all they need is a single opening in the network, the OS or the application to be successful. Hopefully, after sampling the tools here, you can test your own locks and get the peace of mind that your network, your systems and your security measures work.■

Jeramiah Bowling has been a systems administrator and network engineer for more than ten years. He works for a regional accounting and auditing firm in Hunt Valley, Maryland, and holds numerous industry certifications including the CISSP. Your comments are welcome at jb50c@yahoo.com.

Resources

BackTrack: **www.remote-exploit.org/backtrack.html**

John the Ripper and Word Lists: **www.openwall.com/john** and **www.openwall.com/wordlists**

Wireshark: **www.wireshark.org**

Snort: **www.snort.org**

SAINT: **www.saintcorporation.com/products/vulnerability_scan/saint/saint_scanner.html**

Nikto: **www.cirt.net/nikto2**

Nmap: **nmap.org**

MinorFs

The MinorFs user-space filesystem works with AppArmor to provide a flexible form of discretionary access control.

ROB MEIJER

MinorFs is a set of cooperating user-space filesystems that work with AppArmor to provide a flexible form of discretionary access control that operates at the process level. This type of process-level authority restriction is roughly equivalent to that seen in object-oriented programming, providing least-authority restrictions by parameter passing without requiring the administrative overhead of policy controls seen in mechanisms like SELinux. Least authority also is known as least privilege or POLA (Principle Of Least Authority).

In Linux, access to filesystem data is managed by two different access-control mechanisms. First, there is the basic and familiar UNIX discretionary access-control system. The DoD document "Trusted Computer System Evaluation Criteria" (aka the "Orange Book") defines discretionary access control as "a means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control)".

Linux also provides access control through the Linux Security Module (LSM) interface. LSM provides hooks for additional access-control mechanisms, such as mandatory access controls, while leaving the base UNIX discretionary access-control mechanisms untouched. The Orange Book defines mandatory access controls as "a means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity".

These two constructs are combined restrictively, which means if either one denies access, access is denied. Well known users of the LSM interface are Security-Enhanced Linux (SELinux), used in Debian and Red Hat, and AppArmor used in SUSE and Ubuntu.

Although the UNIX discretionary access control for filesystem access has remained at the same (simple user level) granularity for decades, mandatory access control has become more fine-grained (process level). This granularity, however, comes at relatively large administrative costs. SELinux, for example, is known among many administrators for the large amount of overhead that comes with maintaining profiles.

Object Orientation Provides the Model

When designing and writing object-oriented (OO) programs, avoiding global variables, using data hiding, passing references between objects and using established design patterns (like proxies and factories) are concepts we are used to and comfortable with, and most of us have come to appreciate the many advantages these techniques offer. What many of us fail to realize when working with these concepts, however, is the fact that part of what we are doing can be considered access control.

If we look at the OO paradigms from an access-control viewpoint, it is easy to see that the model used by OO programs is both discretionary *and* suitable for the highest granularity. Therefore, you could say that OO programs internally use an extremely fine-grained form of discretionary access control. We must note, however, that this form of access control is actually older than the whole concept of object-oriented programming. The access-control mechanism used implicitly by OO programmers is, in fact, to a large extent equivalent to the access-control mechanisms in use in so-called capability-based systems. Capabilities, often called keys, are an unforgeable authority token that can be passed between programs. In capability-based systems, having a capability gives you the right to use the referenced object within the boundaries specified by the rights associated with the capability. With capabilities, there is no need to check other access-control mechanisms (for example, ACLs); the capability itself contains all the necessary information.

So, why not use this same form of discretionary access control at a slightly coarser level of granularity for access to files and directories by processes? MinorFs aims to do just that, with a lot of help from AppArmor.

First, let's look at how classes, objects and member data, as used in OO design and programming, compare to programs, processes and filesystem data. There are clear indications that we could be dealing with the same set of abstractions at a different granularity level.

You could look at a program the same way you look at a class. A process is an instance of a program (the disk image), the same way that an object is an instance of a class. Most objects have state, in the same way that most processes have state. You could say the same abstractions are there both at the object level of granularity and at the process level of granularity.

Next, we need to map the persistent on-disk directory structures to the same OO model that we just used to model programs and processes. A couple hurdles need to be overcome to accomplish this. First, there is process persistence, which is to say that processes are "not" persistent, so how do they fit the model?

Second, there is pass by reference. If an object wants to share part of its private state with another object that it knows, the object can pass either a copy of or a reference to a part of its internal state. Processes, however, to a great extent are confined to passing copies, not references.

Delegation

One of the most important differences between SELinux and AppArmor is that SELinux is label-based while AppArmor is path-based. There are two heavily discussed issues with path-based security: one is temporary files (that could be solved by using the MinorViewFs temp provisions), and the second is hard links. The perceived hard link problem is that one entity with access to a file could create a hard link that would delegate access to this file. There are many legitimate uses of delegation, and for this reason, advocates of capability-based security advise always to allow delegation. To use delegation effectively, delegate only least authority. In this context, least authority means always delegating the smallest and, if possible, most attenuated subgraph that still could get the job done.

Process Persistence

Programs are persistent; directories and files are persistent, but processes are not. This mismatch makes it impossible to add any persistent on-disk data storage to a process identified by a process ID, because when the process ends, the process ID is no longer valid. The base solution to allowing the OO-like abstractions at the process level of granularity for persistent on-disk storage is to define processes as an incarnation of a so-called pseudo-persistent process. So now, the program still will be equivalent to the class; the pseudo-persistent process is the persistent equivalent to the object, and the on-disk persistent directories and files are equivalent to member data fields. Using this new concept of a pseudo-persistent process gives us the ability to lift the disk data access-control features of AppArmor to a granularity level beyond what is possible with mandatory access control—that is, to the granularity of the pseudo-persistent process, but we don't have the burden of central or human administration, without the administrative overhead that mandatory access control embodies.

Pass by Reference

Where objects in OO languages can pass by reference, most IPC on Linux does not allow pass by reference between processes. One insightful exception to this that early UNIX engineers made was creating the ability of passing file handles over UNIX sockets. You could say that file handles used like this are fully pass by reference. In capability systems, such a reference is called a protected capability or an object capability.

Currently, directory file handles cannot be used as protected capabilities. To overcome this problem, there is a concept from capability-system history that is quite useful. The concept is to use a sparse key string as representation of the reference. That is, we create a relatively long sparse key string that both designates a resource and authorizes access to the resource. This string is called a sparse capability or unprotected capability. This type of capability is somewhat inferior to the protected type of which the UNIX file handle is an example. When combined with protection by AppArmor, it still has many properties that make its usage roughly equivalent to the usage of references in object-oriented languages.

AppArmor

AppArmor is the purely permissive mandatory access-control system used in SUSE and Ubuntu Linux. MinorFs uses AppArmor as its foundation, and in this way, it extends AppArmor so it can be used in a discretionary, even capability-based manner. Although MinorFs might be used separately from AppArmor, its usability is relatively limited. The main reason for MinorFs' limited usability without AppArmor is that by default, processes can access data (like the environment variables or command-line arguments) of other processes by way of the /proc/$PID directories, which (according to MinorFs' philosophy) should be considered private to the process.

This means without AppArmor, processes will, in some cases, be able to steal each other's capabilities through the proc filesystem. Although AppArmor fixes the vulnerabilities posed by the default proc filesystem access rights, MinorFs extends AppArmor. The access-control mechanism provided by MinorFs extends the static least-privilege approach that AppArmor offers with a dynamic least-authority approach. That is, it adds abilities to delegate decomposed and/or attenuated permissions.

The prime property of capability-based security that AppArmor helps us enforce is that processes should not have access to what would be equivalent to global variables. The temp and home directories in UNIX systems in many ways can be considered global variables if we look at them at the process level of granularity.

The way an AppArmor profile works is that it defines a list of permissions that are available for a specific application. For convenience, AppArmor also provides the ability to include sets of permissions with a single include directive.

When designing a system that will use MinorFs, you always should design your separation of privileges setup first. Don't allow your application to become a monolith.

Using AppArmor and MinorFs, you can build privilege-separated applications according to OO or capability paradigms, but even smaller steps can be quite useful. On installation, MinorFs creates a hard link to /bin/bash named /bin/minorbash that has the following AppArmor profile:

```
#include <tunables/global>

/bin/minorbash {
   #include <abstractions/base>
   #include <abstractions/bash>
   #include <minorfs/systemreadonly>
   #include <minorfs/full>
}
```

This profile basically gives a large set of read-only permissions but no write permissions to the version of bash named minorbash and to all programs started by it. This means, you simply can run programs with diminished access rights by starting them from a shell script that uses minorbash instead of bash.

MinorFs

Now, for MinorFs itself. MinorFs currently consists of two user-space filesystems. These filesystems are relatively simple Perl scripts implemented using the FUSE Perl module. Each filesystem has its own distinct task. FUSE (Filesystem in USErspace) is a kernel module that allows nonprivileged users to create their own filesystems.

MinorCapFs

MinorCapFs is at the core of MinorFs. Some time ago, the Linux directory and file-access API was extended with a set of new calls—openat(), mkdirat() and so on—that take an additional first argument, a file descriptor, which specifies from where relative paths should be resolved (these calls are to be standardized in a future version of POSIX). Given the fact that file handles in Linux can be communicated between processes and used as capabilities, it seemed like a good idea to look at the new directory handle calls and create or extend an LSM module so that directory handles could be passed as directory capabilities. The main goal was to use a directory handle as a capability to a directory that wouldn't disclose anything about parent directories.

After discussing my ideas with the AppArmor people, it was concluded that I should try to do as much as possible in user space, so I started designing MinorCapFs. The goals of MinorCapFs are to allow (unattenuated) decomposition, delegation and composition of subgraphs. MinorFs defines a sparse capability for each directory tree subgraph.

In order for you or your program to decompose the directory graph, each file and directory is given an extended attribute named cap. This extended attribute holds the full MinorCapFs path containing the sparse capability for the directory subgraph. Using any form of interprocess communication at your disposal, this path can be shared with any process or even with other users on the same system. The receiving user or process can create a symbolic link in another directory subgraph—for example, in order to make the delegation permanent.

Figure 1 shows how you could use the attr command to fetch the cap attribute, and how this attribute can be used as a short strong path or sparse capability to a directory or file. Normally, you should not use the command line for this but instead do the same thing from your program code. The getxattr function can be used to do the same thing that the attr command does in the example above.

Composition is almost as important as decomposition. Where the usage of extended attributes for decomposition may be strange and new, composition uses a construct that we probably are all much more comfortable with, the construct of using symbolic links. Next to decomposition, MinorCapFs provides the ability to create symbolic links in

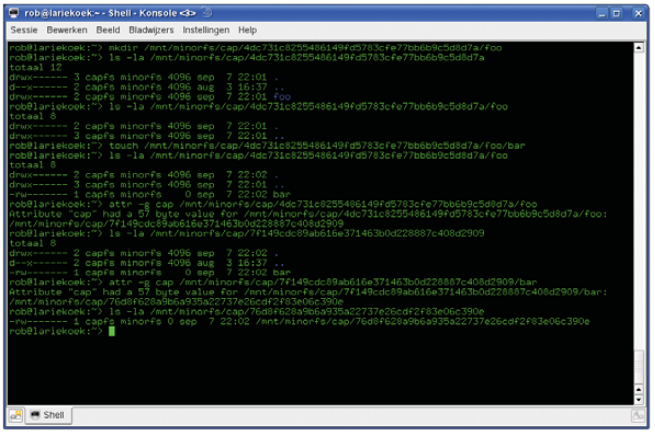


Figure 1. MinorCapFs Extended Attributes

the same way that the filesystems we are used to do. Thus, MinorCapFs combines two basic functionalities for doing simple unattenuated decomposition of directory tree graphs and for doing composition of directory graphs from subgraphs.

You could say that MinorCapFs provides the simplest bare-level form of unattenuated capability-based access control. But, what holds the top-level capability? And, how are subgraphs delegated to individual processes? That's where a second filesystem comes in.

MinorViewFs

As MinorCapFs provides for tree graph decomposition and composition constructs, something has to pass sparse capabilities to processes in order for any process to become able to use MinorCapFs.

To see how we need to solve this, let's take a step back and look at the parallelisms we are trying to exploit. We are trying to make processes into a coarser-grained form of object that, just like objects in any OO language, have private data members. There are two ways to look at the process as such. First, there is the traditional view of nonpersistent processes where all state held by the process disappears when the system reboots or ends for any other reason. You could look at this form of delegation as a better alternative to the troublesome usage of temp directories. Temporary files, by default, would become private to the process until the process delegates them explicitly to other processes.

It is important to note that the temp provision of MinorViewFs is not a reference-counting garbage-collection system. Delegated subgraphs instantly will become invalid at

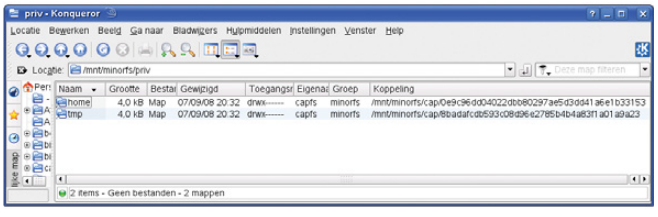


Figure 2. MinorViewFs Links

Working with SharePoint?

SPTechCon
The SharePoint Technology Conference

January 27-29, 2009

Hyatt Regency San Francisco Airport

Burlingame, CA

- SPTechCon features a heavy slate of classes to teach how to take full advantage of SharePoint, from business intelligence tools to reporting and much more.
- Learn best practices for managing a SharePoint environment and integrating it with other systems to unleash the full power of the software ... and your company!
- SPTechCon offers a deep dive into the architecture, and provides practical classes on such SharePoint-centric features as Web parts, lists and pages.
- Learn how to create applications for SharePoint that solve real business problems, and also see what kind of third-party applications have already been created to run on top of SharePoint.

REGISTER by Dec.19
Early Bird Rate
SAVE $300!

Go Behind the SPTechCon Portal
blog.sptechcon.com

PRODUCED BY
BZ Media
SD Times

For more information, and to download the course catalog, go to www.sptechcon.com

the time the owning nonpersistent process dies.

MinorViewFs delegates subgraphs to individual processes by means of two symbolic links under /mnt/minorfs/priv (Figure 2). Each process reading these symbolic links will have a completely different set of subgraph sparse capabilities delegated to it. The second symbolic link /mnt/minorfs/priv/tmp points to the temporary subgraph described above.

Pseudo-Persistent Processes According to MinorViewFs

Although delegation of temporary subgraphs to processes is relatively simple, the concept of the same process being an incarnation of some pseudo-persistent process needs a bit more thought.

MinorViewFs looks at pseudo-persistent processes on a so-called n-th claim basis. What it basically boils down to is that if a program is instantiated while two earlier instantiated versions of the program already are running, the new process will claim the third slot. If the system is rebooted, you also will need to restart the first and second instantiation of the program.

Although appropriate for dæmon-like programs, this, indeed, may be inconvenient for programs like editors and other user-driven programs. To work around these problems, and also to work around the problem posed by scripts and Java programs all being instances of the same program, MinorViewFs uses some simple tricks to determine program, or more specifically, program-invocation-based identity.

So how does MinorViewFs determine a program-invocation identity? First, there is the process parent chain. The process parent chain, including both programs and libraries loaded by those programs, contributes to a unique identity for the invocation. If the parent chain is insufficient as an invocation identity, the system administrator could add a config file under /etc/minorfs/.

Here is an example of a config file for the E language interpreter:

```
<codefile path="/usr/local/e/e.jar" cmdline="true" slots="256">
    <env>DISPLAY</env>
</codefile>
```

The example config adds the command line to the identifying properties of the program invocation. So, using optional config files, MinorViewFs is able to create and re-create a uniquely identifying set of data that allows it to re-delegate a subgraph to a new incarnation of the same program.

The E language named above takes this concept one step further; it allows large subsystems within an E program to be taken together and be serialized and synchronized to disk storage automatically. What's more, the E language is an object-capability language; thus, combining AppArmor and MinorFs with the E language allows you to combine both least authority and private storage all the way down to the object level of granularity. Although E is a bit of an esoteric language, it is a mature and complete language that is worth considering when doing high-integrity projects.

When a process is started and accesses the /mnt/minorfs/priv/home symbolic link, this symbolic link will point to the same MinorCapFs subgraph as the previous time the program was invoked into the same slot.

Next to being useful to new programs designed with privilege separation and least authority in mind, MinorViewFs also can be used with legacy programs like the SSH client. This does, however, involve the usage of the admin tool 2rulethemall that helps the user bypass the basic process-based access-control mechanism with a per-user password. You can put your unprotected SSH private key in the SSH client's private persistent storage space. Again, no program not run by root other than MinorViewFs, SSH or 2rulethemall would be able to access the private key.

Conclusion

MinorFs brings an extreme (capability-based) form of discretionary access control to your AppArmorized Linux system. It uses a form of access control that embraces delegation as a beneficial thing for security. Although MinorFs still is being developed, and is incomplete, it already should provide a useful and intuitive way to create privilege-separated programs that use filesystem access. It provides a way to protect serialized data stored on disk for persistent processes, and a way to protect process private data. And, it's an alternative to the troublesome usage of temp directories.

Upcoming versions of MinorFs will include a third filesystem, MinorCtkrFs that will implement attenuation in a generic way based on the so-called Caretaker pattern. This MinorCtkrFs should add different kinds of read-only capabilities to files and directories, as well as revocable read/write and read-only capabilities.■

Rob Meijer is a computer forensic and security software development professional from the Netherlands. He started his career as a UNIX system administrator, switching one decade ago to software development. In his spare time, he is working on several least-authority-related open-source projects, including MinorFs.

Resources

Trusted Computer System Evaluation Criteria: **www.radium.ncsc.mil/tpep/library/rainbow/5200.28-STD.html**

MinorFs: **minorfs.polacanthus.net**

LSM: **en.wikipedia.org/wiki/Linux_Security_Modules**

AppArmor: **en.opensuse.org/AppArmor**

FUSE: **fuse.sourceforge.net**

Fuse.pm: **search.cpan.org/~dpavlin/Fuse-0.09/Fuse.pm**

Boost: **www.boost.org**

E Language: **www.erights.org**

Robust Composition: **www.erights.org/talks/thesis**

Detecting Botnets

A simple solution combining Darknet and IDS.

GRZEGORZ LANDECKI

We've all heard the stories about botnets and some emerging, professional tools to manage them in a business-like style, but many engineers probably have not had an opportunity to play with them or even research them completely.

Botnets and computer zombies are increasing dramatically. The ShadowServer Foundation continues to gather interesting statistics on this trend, showing how many botnets were found in the last two years (Figure 1).

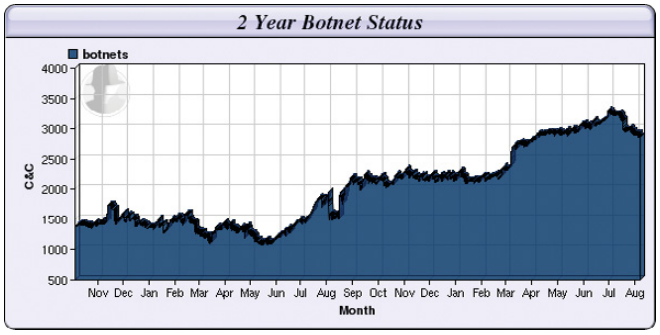


Figure 1. Known Botnets in the Past Two Years

The questions are simple. How can we be sure that no zombie computers exist on our network? Are patching, antivirus, anti-rootkit and antispam protections sufficient? Is something else is necessary? Can we really trust one leading security IT vendor? Would it be better to implement two? Should we exercise some other techniques?

Unfortunately, there are no easy answers to those questions. In March 2008, a security company called Damballa was the source of news that a new Kraken botnet existed in the wild and was far more resource-reaching than the Storm one. Damballa reported seeing approximately 400,000 compromised computers (victims)—some of them from at least 50 Fortune 500 companies. It's an interesting example, because many security (mostly antivirus) vendors responded quickly that they already had protection in place and that the threat was old, so no need to worry. Was this really a threat, and how did Damballa get these numbers?

To simplify the story, Damballa discovered (probably during a security audit) a new malware with hard-coded addresses (URLs) of control centers (CCs—computers that manage tasks for zombie machines and all infected computers report to them). Damballa also found that some of those hard-coded addresses were not registered in a DNS service (the botnet probably was tested at that time, and the authors were preparing to launch it later). Damballa registered those domains as its own and ended up controlling quite a large botnet for research. Now, Damballa could identify IP addresses of zombie computers that started to report to its CC, and it discovered a number of devices sitting inside large corporate networks. Damballa could play with the bots and discover their potential power for malicious activity.

Much discussion has ensued about Damballa's ethical behavior. It hasn't contacted any security company about the methods of infection it discovered. It hasn't published any details of the exploits used to any bugtrack, nor has it contacted any vendors to alert them of the issue. Damballa wanted all the credit itself.

I don't approve of those things, but as a security technologist, having the opportunity to research such botnets is really tempting, and I can understand (but still not agree with) those decisions. Having an army of zombies under the control of a security organization is much better than having them in the wild. On the other hand, Damballa allowed malware to spread undetected just to justify its research.

But, that's not the point. The real point is Damballa proved that undetected botnets could exist, even in highly secured environments, in companies that have dedicated resources to fighting malware.

So, if large corporations that have committed a small fortune to protect system and network resources can be vulnerable, who's safe? Apparently, having state-of-the-art antivirus and malware protection isn't enough. What can you do about it, and how should you protect your IT systems and fight undetectable malware?

One solution is something called Darknet.

The idea of Darknet isn't new. It evolved from honeypots—a solution that's undervalued and underestimated, although it's

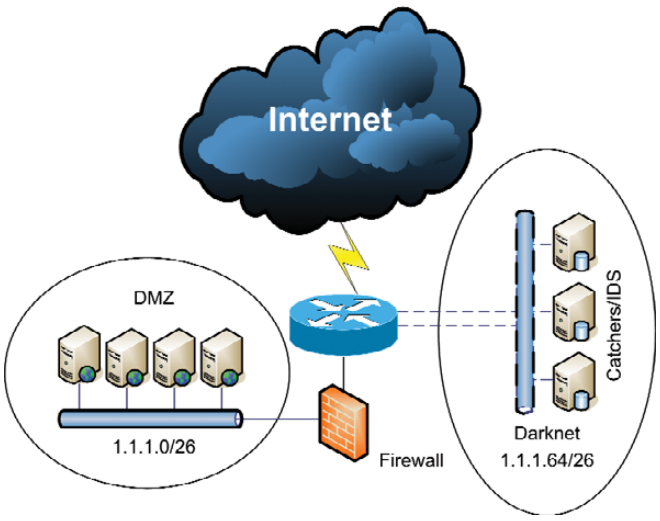


Figure 2. Darknet sits quietly waiting and listening.

really easy to implement. The term Darknet refers to a private or public chunk of a network that is empty of any servers/services. In fact, there is at least one silent host on this network, catching and inspecting all packets. We can call it a silent honeypot. The idea is simple. We don't expect any traffic on this network, so any packet found here is not legitimate and needs to be analyzed.

As shown in Figure 2, the network has been divided into two parts with a /26 mask. The Darknet part consists of silent "traffic catchers" or Network Intrusion Detection Systems (NIDS).

There are plenty of sophisticated commercial Network Intrusion Detection Systems, but if you don't want to pay a lot of money, you can use some of the open-source and free solutions, such as Snort, Argus or even the fully functional Darknet solution from Team Cymru (see Resources). These tools allow you to gather detailed packets for analysis of new or zero-day exploits in the wild.

Figure 3 from the Team Cymru Web site shows how Darknet detected a worm just minutes after its release.

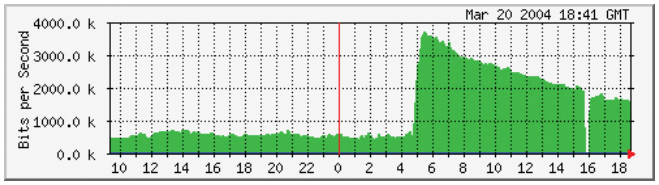


Figure 3. Notice the unusual spike in traffic.

In this example, Darknet has a public address space, which means it will catch all the traffic from outside the network. So, we will have all the information about what threats are currently in the wild, and we will be alerted about new traffic patterns and potential zero-day exploits. But, how can we detect botnets inside our network? To answer that question, we need to look deeper into malware behavior.

About 90% of malware these days behaves in specific and common ways, so from the network traffic perspective, we can say that typical malware has some distinct characteristics:

1. It will assure its survival. It's not exactly network-related, but it will copy itself to the Start folder or add itself to startup scripts or the registry (Windows).

2. It will try to replicate and spread (infect other computers in its neighborhood) by searching for e-mail addresses and sending messages from a user's mailbox (mail channel); creating files on Windows shared folders, network drives and P2P shares (let's call that the P2P channel); or direct infections—using zero-day exploits on unpatched systems.

3. It will try to contact the control center (CC) to download other malware and to get instructions—usually from Web sites (Web channel) or Internet Relay Chat (IRC channel). Often these CCs are located on computers using dynamic IP addresses (dynamic DNS) or located in countries known to be sources of malicious software (China, Russia and so forth) or on suspicious networks (such as the so-called Russian Business Network).

4. It will be used for malicious purposes—typically spam (mail channel), data leakage/spyware/identity theft/phishing, DDOS, ransomware, often via the Web channel also.

As we can see, malware often uses the most popular channels to spread and operate—mainly Web, mail, P2P and IRC channels.

Knowing this information, we can create a Darknet inside our network and place some traffic catchers or IDS systems there to analyze and gather all suspicious data.

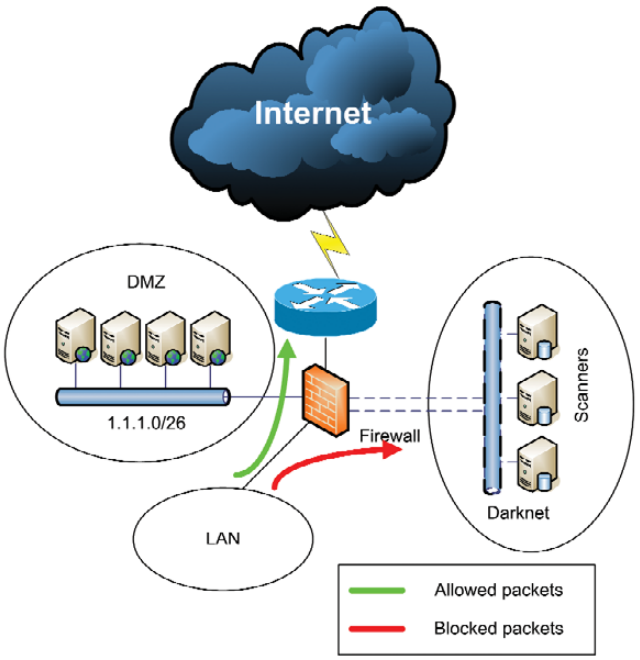


Figure 4. Suspicious packets are examined instead of simply discarded.

The method shown in Figure 4 can be explained in one sentence: "All outgoing traffic that is not legitimate (violates a company's policy) or traffic that is suspicious will be forwarded for analyses."

One question remains. How do we decide what traffic is malicious or unwanted? The ultimate solution would be to forward all packets with an "evil bit" set in a funny way (RFC 3514). Unfortunately, this is a little more complicated.

Let's consider an example. If we have a company with internal mail and a name server (DNS/WINS), we can redirect all outgoing traffic (other than from these servers) to ports TCP 25 (SMTP), TCP/UDP 53 (DNS), TCP 6667-6669 (IRC) and all known P2P software (like Limeware) to Darknet hosts for analysis. As computers inside the network don't really send traffic directly to mail servers or connect to the IRC, we can block these channels to avoid spreading malware. If the nature

of a company's business is focused on a local area or country, we also can redirect all WWW port TCP 80 requests to suspicious domains (such as .cn or .ru), dynamic DNS domains and so on.

To accomplish this task, we can set up basic iptables rules on a Linux firewall, as in this example (we are redirecting all requests coming from an internal eth0 interface destined for TCP 6669 IRC port to internal host 1.1.1.1):

```
iptables -A PREROUTING -t nat -i eth0 -p tcp --dport
 ➥6669 -j DNAT --to 1.1.1.1:6669
iptables -A FORWARD -p tcp -i eth0 -d 1.1.1.1 --dport
 ➥6669 -j ACCEPT
```

We also will need to configure the internal server with address 1.1.1.1 to catch all the traffic. There are two ways to do that: we can record all the packets going to this server, or we can install some services (WWW, IRC, SMTP, POP3, DNS) and then monitor them for connections and integrity.

Let's focus on a simple packet-capture machine. More sophisticated solutions (such as the ones from antivirus companies) usually have a dozen machines (most likely VMware images) with different operating systems, open shares, Web servers, P2P clients, mail agents, instant-messaging clients and so on.

After the attack/infection, system changes will be compared to the input state (VMware snapshot) to analyze malware behavior and to ease the remediation process.

Such labs can be very complex, but to achieve basic functionality (traffic monitoring and threat alerting), it is enough to have one computer with your favorite Linux distribution.

Traffic Monitoring

One of the many tools for sniffing traffic and gathering statistics is ntop. You can download it from **www.ntop.org** or use a package manager on your system to install it. There already are cooked packages for popular Linux distributions, such as

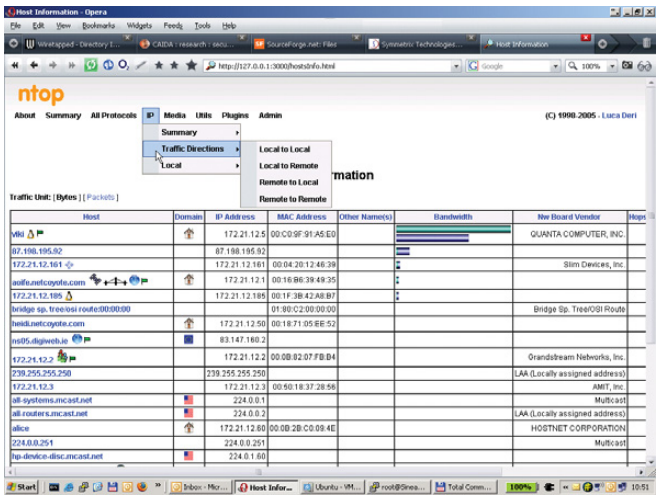


Figure 5. ntop breaks down the flagged traffic to help identify the source of illegal traffic.

Figure 6. ntop offers a wide variety of graphed information.

Red Hat, Debian/Ubuntu and SUSE. Before using it, you have to set up an admin password by running the following:

```
sudo ntop --set-admin-password
```

And start it with:

```
sudo /etc/init.d/ntop start
```

Now you can go to your IP address (http://127.0.0.1:300) and look for some statistics. This is a very powerful tool that provides a lot of information. You can sort by packets, ports, hosts and so on. Network usage graphs also are helpful in determining the amount of traffic getting into your system.

Remember, no packets should be legitimate in Darknet, so this tool provides great statistical data as to what hosts/networks are responsible for illegal traffic.

Figure 5 shows ntop's graphic interface and its ability to detect host operating systems, vendor and other details in Host view.

Figure 6 presents standard ntop graph capabilities, thanks to built-in support for RRDTool.

Threat Alerting

To get alerts regarding what exploits are used (if any) on your network, you need a network IDS system. The best one that's publicly available is Snort. You can get it from **www.snort.org**, and it also is available on many systems as a binary package.

One thing you need to configure in /etc/snort/snort.conf is setting your $HOME_NETWORK variable to match IP addresses and netmask to your configuration. Snort is an intrusion detection system based on a pattern database.

If traffic matches, it will write an alert to a log file (by

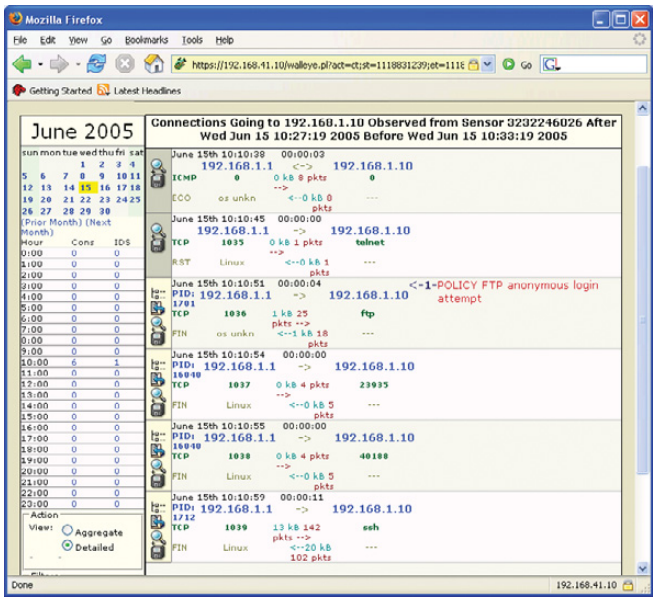


Figure 7. The honeypot GUI shows recorded incidents.

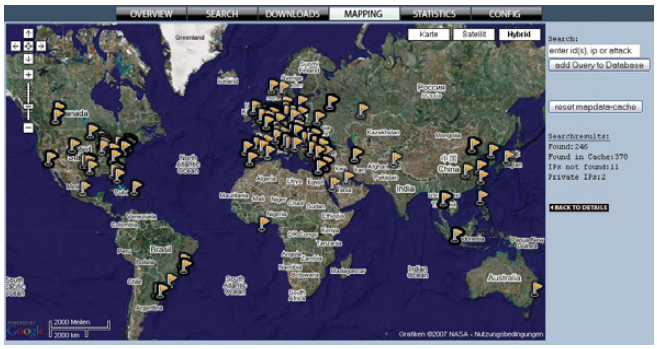


Figure 8. By mapping IP addresses, we can see geographic trends.

default in /var/log/snort) and record the packets for later analysis (you can reply to them using the `tcpdump -r` command or examine them using tools like Wireshark).

With powerful yet not complicated rules, you can write your own signatures or edit existing ones to record traffic that matches your custom criteria. Additionally, you can consider installing Snort support tools, such as IDScenter (see Resources).

There also is a Honeynet project, based on Snort and Sebek technologies. It provides a cut-down Linux system, based on Fedora and custom-built tools with a GUI for incident management (Figure 7).

If you want to go further, there also are projects, such as HIHAT (Highly Interactive Honeypot Analyses Toolkit), that transform popular PHP applications, such as PHPNuke or osCommerce, to fully functional logging, reporting and alerting tools.

You easily can detect commands and SQL injections, cross-site scripting and map involved IPs to geographic locations, as shown in Figure 8.

Advertiser Index

Advertiser	Page #	Advertiser	Page #
1&1 Internet, Inc.	1	O'Reilly Emerging Technology Conference	77
www.oneandone.com		etech	
Aberdeen, LLC	7	Polywell Computers, Inc.	59
www.aberdeeninc.com		www.polywell.com	
ASA Computers, Inc.	93	The Portland Group	53
www.asacomputers.com		www.pgroup.com	
Cari.net	45	Rackspace Managed Hosting	C3
www.cari.net		www.rackspace.com	
Coraid, Inc.	33	Servers Direct	9
www.coraid.com		www.serversdirect.com	
Emac, Inc.	58	Silicon Mechanics	11, 31
www.emacinc.com		www.siliconmechanics.com	
EmperorLinux	41	SPTechCon	75
www.emperorlinux.com		www.sptechcon.com	
Genstor Systems, Inc.	71	SXSW Festivals and Conferences	83
www.genstor.com		www.sxsw.com	
Linode.com	63	Technologic Systems	13
www.linode.com		www.embeddedx86.com	
Logic Supply, Inc.	65	Ubiquiti Networks, Inc.	C2
www.logicsupply.com		www.ubnt.com	
Microway, Inc.	C4	USENIX Association	89
www.microway.com		www.usenix.com/events	
Mikro Tik	3	ZT Group International	5
www.routerboard.com		www.ztgroup.com	

Results

This simple configuration of putting a server on an internal Darknet allows us to detect and receive alerts on the following:

1. Actively spreading malware.
2. Covert channels and possible data leakage.
3. Suspicious activities (deliberate or not), such as abuse of a company's policy and network reconnaissance attempts (for example, port scanning).
4. Provide audit trails and record evidence for later investigation.
5. Provide general network usage statistics for base-lining.

Not All Traffic Is Malicious

Although you decided to block IRC access from inside the network, it might not be that clear for other employees in your company. If Mary from another department tries to connect to her favorite IRC channel at lunchtime, you'll probably catch it, but that doesn't mean there is a malware on Mary's workstation trying to contact the control center. However, a number of the same type of connections from one or multiple computers often is a good indication that something is going wrong.

In my work every day, I see some strange behavior. People always are trying to install illegitimate software, sometimes without even knowing it. Sometimes an employee's children try continuously installing Limewire on a company laptop given to them for playing a game or browsing the Internet.

With a little bit of information, you should be able to gather some statistics and distinguish real threats from normal misuse or other isolated incidents.

Securing information systems is a very hard task. Today we are in ongoing war against attackers—fighting the battles of time and money. Time is crucial in securing all environments when there is a threat in the wild, but first you need to know about it. If you know your enemies, their intentions and weapons, it is much easier to react and mitigate attacks. That's what Darknet and honeypots are all about.■

Grzegorz Landecki, CCNP, CISSP, is a security technologist at Cyber Security Team in Dublin, Ireland, responsible for protecting a major US company's 85K+, globally located computers.

Resources

ShadowServer Foundation: **www.shadowserver.org**

Damballa: **www.damballa.com**

Snort IDS: **www.snort.org**

Argus: **www.qosient.com/argus/flow.htm**

Team Cymru Project: **www.team-cymru.org/Services/darknets.html**

Setting an Evil Bit RFC3514: **rfc.net/rfc3514.html**

Snort IDS: **www.engagesecurity.com/products/idscenter**

Honeywall Project: **https://projects.honeynet.org/honeywall**

HIHAT Project: **hihat.sourceforge.net**

CAIDA Network Telescope Research: **www.caida.org/research/security/telescope**

University of Michigan—The Internet Motion Sensor: A Distributed Blackhole Monitoring System: **www.isoc.org/isoc/conferences/ndss/05/proceedings/papers/ims-ndss05.pdf**

Tracking Global Threats with the Internet Motion Sensor: **www.nanog.org/mtg-0410/pdf/bailey.pdf**

Commercial Example of the Darknet Implementation: **https://tms.symantec.com/Default.aspx**

The Honeynet Project: **www.honeynet.org**

TECH TIP Using ps to Monitor Processes

In a previous tech tip, we saw how to use kill to monitor processes. Another option is to use ps. With both methods, you can check $? for success/failure. However, note that `kill -0` may return failure even if the process actually exists. This happens when the current user has no permission to the process in question, for example: `kill -0 1`.

To check for a process silently (with no output), use:

```
kill -0 PID 2>/dev/null
ps -p PID >/dev/null
```

—JANOS GYERIK

SXSW '09 interactive

march 13–17 • austin, tx

SXSW INTERACTIVE FESTIVAL: CONNECT, DISCOVER, INSPIRE

Attracting digital creatives and new media entrepreneurs, the 15th annual South by Southwest (SXSW) Interactive Festival gives you both practical how-to information as well as unparalleled career inspiration. Attend this legendary gathering of the tribes to renew your link to the cutting edge.

Tony Hsieh to Deliver Opening Remarks on Saturday, March 14

At Zappos.com, Tony Hsieh has fostered a culture where extraordinary customer service is the norm. On Saturday, March 14, hear him talk about how good deeds can help you leverage the power of your audience to massively extend your brand.

Scheduled 2009 panels include:

Appfrica: How Web Applications Are Helping Emerging Markets Grow • Being a UX Team of One • CSS3: What's Now, What's New and What's Not? • Gestural UI: iPhone Taught Us Flick and Pinch. What's Next? • Get Me Rewrite! Developing APIs and the Changing Face of News • How To Roll Your Own API • The Invisible Web and Ubiquitous Computing • Make it So (Sexy): Lustful Design in Mainstream Science Fiction • Making Web Widgets Accessible: Tools and Techniques • More Secrets of JavaScript Libraries • OpenID, OAuth, Data Portability and the Enterprise • Post Standards: Creating Open Source Specs • Version Control: No More Save As... Many, many, more to be announced!

ScreenBurn Panels & Evening Events
March 13-17, 2009

NEW!

ScreenBurn Arcade
Friday, March 13 • 2-6pm
Saturday, March 14 • 12-6pm
ScreenBurn Arcade EXTRA DAY!
Sunday, March 15 • 12-6pm

REGISTER TO ATTEND SXSW INTERACTIVE 2009

Register before **January 16** to receive the next discounted rate and get the best choice of available hotels: **sxsw.com/attend**

Attend SXSW Film and SXSW Interactive at a bargain rate by purchasing a **Gold Badge**

SOUTH BY SOUTHWEST INTERACTIVE FESTIVAL
March 13-17, 2009 | Austin, Texas | **sxsw.com**

Miller Lite · FUZE · Microsoft Silverlight · The Austin Chronicle austinchronicle.com

MythVideo: Managing Your Videos

Managing your videos has gotten a little easier with MythVideo, but it helps knowing a few expert tricks.

MICHAEL J. HAMMEL

MythVideo is a video management plugin for the open-source personal video recorder (PVR) system known as MythTV. Its primary purpose is to help organize digital videos that are saved on a MythTV back-end server for display on front-end client systems. The most common use of MythVideo is to create a personal digital archive of videos ripped from DVDs.

In this article, I explain how to configure both your hardware and the MythVideo software so you can make the best use of your computers and disk space, while still providing a comfortable user experience with uninterrupted playback of your digital videos. First, I walk through the process of using and configuring MythVideo and then cover some tips on improving both the process and the end result.

It is assumed that you have MythTV and its associated software installed. MythVideo doesn't require support for live TV, so I don't cover configuration of live TV components in this article.

MythTV Overview

The MythTV system has a client/server architecture that utilizes plugins to extend its feature set. The server side is known as the back end, and it is generally responsible for providing the hardware required for live TV recording and the storing of audio and video content for use within the MythTV system. It also provides database features used by both MythTV and its plugins.

The client side is known as the front end, and it primarily is used for playback of content that is stored on the back end. This can include viewing videos or listening to music, but it also includes browsing photos and the Web, making Internet phone calls, displaying the weather forecast and even ordering movies from Netflix. Front ends and back ends are separate pieces of software that communicate over a network, but they also can run on the same computer.

MythVideo is a plugin that runs on a front-end client and communicates with the back-end server to manage videos. It provides administrative tools for adding new videos to the system or for editing video information, along with tools for selecting videos for playback. Videos are stored on the back end but must be made available over a network using NFS in order to be played by the front end.

MythVideo User Interface

The MythTV display is divided into pages. There are three sets of pages specific to using MythVideo: the video selection pages, the video manager pages and the video settings pages. The video selection pages (Videos on the main menu) is where you browse your video collection, select a video and play it. There are three ways to view your collection: browsing one at a time, as a pageable gallery and as a list. Each method allows you to view the video title, summary information (running time, directory, plot summary and so forth) and artwork.

Browse Mode sorts all your videos alphabetically, and although the information it displays is detailed and easy to read, it can take some time to browse a large collection. Use paging keys (by default, this is the Page Down key on a

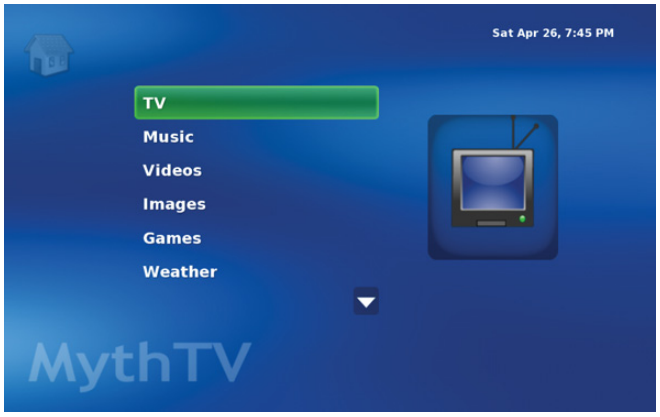


Figure 1. MythTV Utilizing the MythCenter Theme

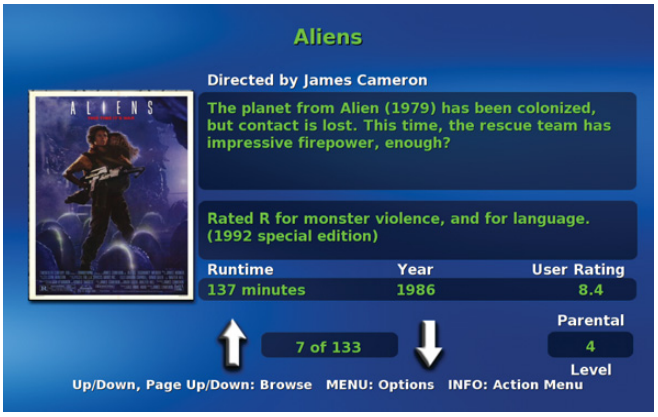


Figure 2. Browse Mode

A Word about MythTV Themes

Many themes are available for MythTV, and each can be configured in a variety of ways. The MythMediaCenter theme was used while writing this article, and the theme was configured (see Setup→Appearance) to use the Classic menu theme. Screenshots in the article reflect this specific setup.

Despite the difference in themes and configurations, the underlying functionality related to MythVideo remains the same. All themes offer the same set of video browsing options and the same administrative interfaces. The only difference between themes is where you find the menu option that takes you to each of these features. If you have problems finding a particular page described in this article, feel free to drop me an e-mail, and I'll try to help you out.

Figure 3. Gallery (Upper Left) and List (Lower Right) Modes

Figure 4. Video Settings Main Page

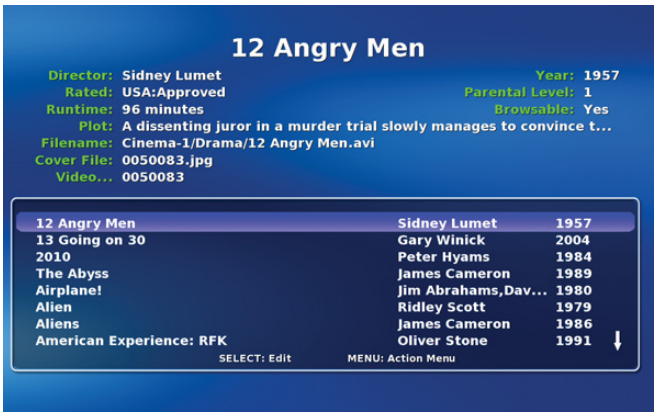


Figure 5. Video Manager Main Page

keyboard) to page through the list a little faster.

List Mode displays two small windows. The left side is the current folder and the right is the contents of that folder. If you have all your videos in one folder, List Mode is only a slight improvement over Browse Mode. However, if you arrange your videos in topical folders (by genre, for example), List Mode makes finding a video much easier than Browse Mode.

But, if you've arranged your videos in genre-oriented folders, which is the recommended manner for this article, the Gallery Mode probably is easier to use than either Browse or List modes. This is because the Gallery Mode lets you see a user-defined set of thumbnail poster art for the videos in the current folder. This mode does run a little more slowly than list mode, however, as MythVideo needs to cache the rows-by-column set of thumbnails for the current folder at least once.

Video Settings

MythVideo can be configured on two sets of pages. The first is found under Setup→Video Settings. These pages allow global configuration of items like the MythVideo storage directory (under General Settings), how the gallery will lay out thumbnails (also under General Settings), which tools to use for playback (under Player Settings) and ripping options (under Rip Settings).

The Video Settings are global in scope, which means they apply to all videos unless a video has its own configuration. Setting video-specific configuration is done with the Video

Keyboards vs. Remote Controls

If you're just getting started with MythTV, use a keyboard. The default keyboard mappings are easy to learn and modestly well documented on the MythTV Wiki. However, moving to a TV remote control (using LIRC and an infrared receiver) is an advanced topic that only experienced users will want to tackle, partly because setting up LIRC is not easy but also because, once set up, you still need to teach LIRC about your specific remote and how it interacts with MythTV.

Manager (Videos→Video Manager). This section of MythVideo allows you to acquire metadata for videos, set a video-specific player, choose how to play videos in sequence (one after another), and choose poster art to display while browsing videos.

Familiarize yourself with the Video Manager, as it will become important when cleaning up artwork for your videos, not to mention when dealing with videos that don't play well with the internal video player.

The MythTV internal video player does a good job with most videos, and I recommend it over external players (at least for use with MythTV). But, I've found it to have a problem with some videos ripped with MEncoder, though this may be due to a bad DVD reader and not to MEncoder. Still, the way around this (until I can replace the faulty hardware) is to choose an external player, such as MPlayer or Xine. And, using the Video Manager is the best way of dealing with this problem should it occur.

Day-to-Day Usage

The first step in using MythVideo is to rip your DVDs. There are a number of tools for doing this, including a MythTV DVD ripper, but I've found AcidRip to be the easiest to use for beginners (advanced users will want to move on to DVD::RIP or try using the command-line utilities MEncoder and Transcode). You'll want the smallest files you can get, without significant loss of quality, using the AVI file format with the audio and video ripped to MPEG-3 and MPEG-2, respectively. Other formats might produce better quality or smaller files, but if you're just getting started, start with these settings. Fortunately, these selections are the default with AcidRip, so the only thing you need to do is play with the file size in order to find the smallest size (see the General tab File Size field) with the best video quality (see the Video tab bits/px and Bitrate fields).

Once you have a ripped file, you need to store it in MythVideo's storage directory (see the Video Settings section discussed previously). I have internal disk space of about 150GB on an IDE drive and 500GB on an external USB drive.

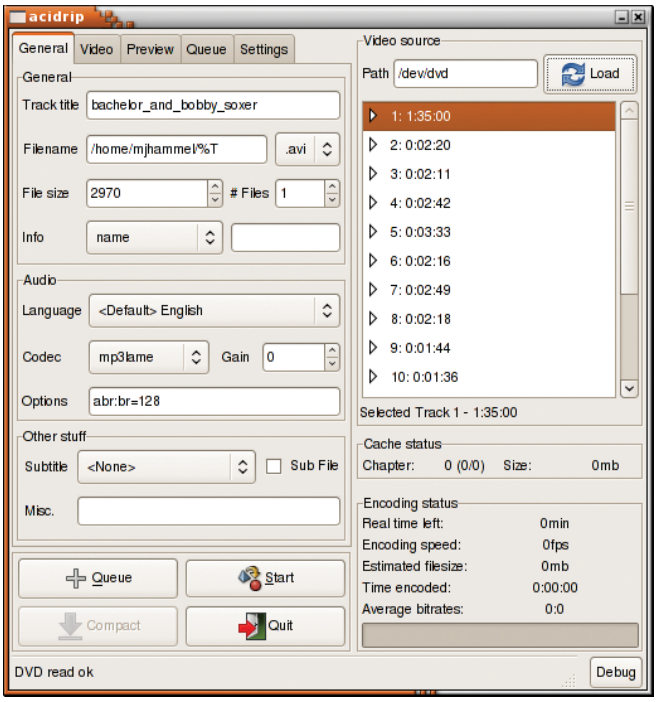


Figure 6. AcidRip

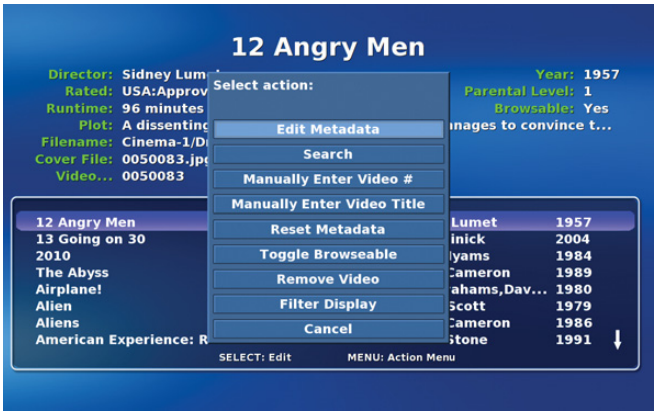


Figure 7. Video Manager Menu

I use the internal drive for TV recordings and the external drive for videos. I mount the external drive under /store and set this in the Video Settings pages.

The videos are ripped by AcidRip and then copied to the external drive manually. This is so that I can rip them to temporary storage first and verify they work under MPlayer or Xine before installing to MythVideo's directories. I do this to save wear and tear on the external drives, some of which have less than stellar reliability.

Once you copy a video into the MythVideo storage area, you need to grab its metadata using the Video Manager. If you're using a remote control with MythTV, note that this step is easier to do with a keyboard, though you can use the

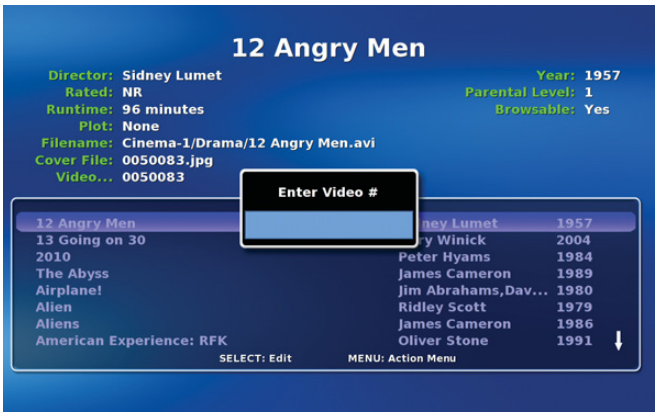


Figure 8. Video Manager Manual Search

built-in keyboard with your remote control. I don't recommend this if you have lots of new videos to add or if you add videos often.

To update the database, choose Videos→Video Manager. This takes you to a page where you can select a video to edit. Your collection is listed alphabetically by video title with the director and year also listed. New additions to the MythVideo storage directories show up with the filename, followed by Unknown for the director and a question mark for the year.

Page through the videos, if necessary, until you find the new entry. With the entry highlighted, press M for the menu, then select Search. If all goes well, MythVideo will find the video on the IMDb database and fill in the metadata for you.

If MythVideo locates the video in the IMDb database, you'll need to find the video manually with your Web browser. The URL for the video will be suffixed with an ID, something like tt0362227. Drop the leading alphabetic characters and note just the numeric portion of this ID. In the Video Manager, in the menu, choose Manually Enter Video Number, type in the number and then press Enter. MythVideo will fetch the appropriate information based on the video ID.

Storage Tips

Now that you know the basics, there are a few tricks to make this all work a little better. First, you'll want large storage drives for your videos. Even when ripped to the relatively small AVI files, a collection of 100 videos each ripped to 2GB in size will take up 200GB of disk space. And, if you're like me, you've probably purchased much more than 100 DVDs.

Next, you'll want to separate your videos from your live TV recordings. My internal IDE is a 7200RPM drive, and my external USB 500GB drive is only 5200RPM. The latter is fast enough for playback but not ideal for video recording. That's another reason I rip to temporary storage (on a fast IDE drive) before copying to the external USB drive.

External drives are easier to install than their internal counterparts. However, you'll need to make each drive a different directory under the main MythVideo storage directory. I created a directory called /store/movies/Cinema-1 for my first external drive, then mounted the external drive to that directory. The /etc/fstab entry looks like this:

```
# MythTV drives
/dev/sdc1   /store/movies/Cinema-1   ext3   defaults   0 0
```

If you have multiple drives, you may need to write a program to identify what drives are allocated to which device files at bootup time, because it's possible that the drives may not be recognized in the same order each time. This is a problem when dealing with external USB drives and a reason I'm currently using only one very large drive.

A minor problem with USB drives is that they spin down when not in use. This means the first time you browse your video collection to that drive, there may be a modestly long pause while the drive spins up. Fortunately, this is, at most, an inconvenience and will not affect playback of the video.

I've had good luck with my Western Digital 500GB USB drive, but I've had poor luck with Maxtor drives—two of three drives have failed inside of the first week (the other is working fine, however). At the time of this writing, the Seagate FreeAgent drives were having problems related to power-saving mode under Linux. Workarounds are available, but until Seagate resolves the problem, you probably should avoid those drives.

Another tip is to place your DVD readers on separate machines, if available. This will allow you to rip your videos to NFS mountpoints without affecting performance off your MythTV back end. I export /store/rip from my back end to all my systems and rip to that directory from various places, including my laptop. Again, /store/rip is on the internal IDE drive, so it doesn't adversely affect playback of saved videos from the external drive. My exports file, /etc/exports, looks like this:

```
/store 192.168.1.0/255.255.255.0(rw,sync,no_root_squash)
/store/movies/Cinema-1 192.168.1.0/255.255.255.0(rw,sync,no_root_squash)
/music 192.168.1.0/255.255.255.0(rw,sync,no_root_squash)
```

Note that my back-end server is behind a firewall with no direct access from the outside world. I'm not streaming any videos across the Internet, which is fairly pointless, as the throughput would be quite bad from my home. The videos are accessible only from within my home network.

Administrative Tips

Now, let's look at naming your ripped videos. AcidRip pulls the name of the video from the disk but generally uses all lowercase letters and replaces spaces with underscores. You always should change this to be the same as the title of the video as it is listed on IMDb.com. Because the metadata lookup will use that name, you'll have a far greater chance of having the automated lookup succeed if you simply use the correct title for the video's filename when you rip the video.

You'll also want to categorize your videos. The primary reason

for this is that you won't want to scroll through 100s of videos in any mode (Browse, List or Gallery) using MythVideo.

If you create top-level directories with the category names and then copy the videos into those directories instead of the top-level MythVideo directory, browsing the files in any of the available modes will be a bit easier. Ideally, MythVideo would allow you to categorize the files without creating directories manually, but because it doesn't do that yet, this is the next best way to handle the issue. As an added bonus, you can add an image file called folder.png (or folder.jpg) to each category directory and that image file will be used as an icon in the Gallery display.

My directory structure looks like this:

- /store/movies: top-level storage directory configured for MythVideo.
- /store/movies/Cinema-X: mountpoints for each external drive, with X replaced by a number.
- /store/movies/Cinema-X/category: video categories, with category being one of the following: Action, Comedy, Drama, Romance, War, Classics, Documentary, Fantasy, SciFi and Westerns.

Note that each external drive, when mounted, also includes a lost+found directory. MythVideo is smart enough to ignore this directory, as should you when managing your videos.

A Word about Artwork

The artwork retrieved for your videos for display while browsing the collection is not always ideal. Some videos end up with rather obscure poster art. If this bothers you, the simple solution is to scan the cover of your DVD case and save it to your posters directory. This directory is configured under General Settings in the Video Settings page. After you scan the case cover art, save the file in this directory using the same filename as the original poster file retrieved from IMDb. The filename for the poster of each video is listed in the Video Manager page. Alternatively, you can save it using a different name and then manually edit the metadata from the Video manager.

The size of your scan doesn't matter, although you might want to make it roughly the same size as the original poster art to reduce the time MythTV spends resizing the image. Resizing occurs all the time and is based on the settings for the number of rows and columns to display or whether you're in List or Browse mode. So, there is no really ideal size. The file format for poster art should be JPEG.■

Michael J. Hammel is a Principal Software Engineer for Colorado Engineering, Inc. (CEI) in Colorado Springs, Colorado, with more than 20 years of software development and management experience. He has written more than 100 articles for numerous on-line and print magazines and is the author of three books on The GIMP, the premier open-source graphics editing package.

Resources

MythVideo: **www.mythtv.org/wiki/index.php/MythVideo**

AcidRip: **untrepid.com/acidrip**

DVD::RIP: **www.exit1.org/dvdrip**

Transcode: **www.transcoding.org/cgi-bin/transcode**

Mencoder/MPlayer: **www.mplayerhq.hu/design7/news.html**

Xine: **xinehq.de**

VLC: **www.videolan.org/vlc**

IMDb: **imdb.com**

LIRC: **www.lirc.org**

USENIX Upcoming Conferences

1st Workshop on the Theory and Practice of Provenance (TaPP '09)

Co-located with FAST '09

FEBRUARY 23, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/tapp09

7th USENIX Conference on File and Storage Technologies (FAST '09)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

FEBRUARY 24–27, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/fast09

2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)

Sponsored by ACM SIGPLAN and SIGOPS in cooperation with USENIX

MARCH 11–13, 2009, WASHINGTON, D.C., USA
http://www.cs.purdue.edu/VEE09/

First USENIX Workshop on Hot Topics in Parallelism (HotPar '09)

MARCH 30–31, 2009, BERKELEY, CA
http://www.usenix.org/hotpar09

8th International Workshop on Peer-to-Peer Systems (IPTPS '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/iptps09
Submissions due: January 9, 2009

2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/leet09
Submissions due: January 16, 2009

6th USENIX Symposium on Networked Systems Design and Implementation (NSDI '09)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 22–24, 2009, BOSTON, MA, USA
http://www.usenix.org/nsdi09

12th Workshop on Hot Topics in Operating Systems (HotOS XII)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)S

MAY 18–20, 2009, MONTE VERITÀ, SWITZERLAND
http://www.usenix.org/hotos09
Paper submissions due: January 13, 2009

2009 USENIX Annual Technical Conference

JUNE 14–19, 2009, SAN DIEGO, CA, USA
http://www.usenix.org/usenix09
Paper submissions due: January 9, 2009

18th USENIX Security Symposium

AUGUST 12–14, 2009, MONTREAL, CANADA
http://www.usenix.org/sec09
Paper submissions due: February 4, 2009

23rd Large Installation System Administration Conference (LISA '09)

Sponsored by USENIX and SAGE

NOVEMBER 1–6, 2009, BALTIMORE, MD, USA
http://www.usenix.org/lisa09
Paper submissions due: April 30, 2009

Using Capistrano

"We will encourage you to develop the three great virtues of a programmer: laziness, impatience, and hubris."—Larry Wall, *Programming Perl* DAN FROST

For most programmers, deployment is an area that could do with a touch of laziness. Deploying to a cluster—or even one machine—can be repetitive and tiring. Enter Capistrano, a Ruby deployment tool that makes the task of deploying an application to servers easier by running defined tasks for you on the remote servers.

The Ruby programmers' toolbox contains so many tools for eliminating most of their work, it's fair to say that Ruby programmers are probably some of the laziest. If having all the boring jobs done for you isn't enough, Ruby programmers even contrive to have most of their tools built in one language—Ruby. No bash-make-PHP-Perl combinations. It's all Ruby.

Think of Capistrano as a build system that specializes in running commands remotely on any number of machines. If you have to connect to a half-dozen machines to push updates, or have no quick-and-easy way of rolling back the entire cluster if (or when?) something goes wrong, you need to be a little more lazy.

Capistrano groups tasks in recipes, and the default recipe, which we'll look at in a moment, is very geared toward Rails, running migrations and restarting the Rails server. However, Capistrano's core is not Rails-specific. You can build your own recipes for all your dullest tasks, and you can tweak the Rails recipe to work with whichever language or framework you're using.

The Ruby programmers' toolbox contains so many tools for eliminating most of their work, it's fair to say that Ruby programmers are probably some of the laziest.

Let's take a look at what Capistrano does for Rails deployment, how to build your own tasks and how to push your own application out to 20 servers with just one command.

Capistrano and Rails

Like Rails, Capistrano increasingly is deployed with flavours of Linux and is installed by default in Leopard, so you might not even have to install it. If you do need to, installing Capistrano is as easy as any Ruby gem. Simply run:

```
sudo gem install -y capistrano
```

Capistrano has two main commands: cap, which is used for viewing and running the tasks, and capify, which is used to set up a Rails project for use with Capistrano. Assuming you have a Rails project, grab a copy of it, and run capify at the project root:

```
cd path/to/project capify .
```

This creates just two files: Capfile and config/deploy.rb. Capfile is to Capistrano as Makefile is to make and Rakefile is to rake. Capistrano expects a Capfile to be present and to contain the tasks or to include a Ruby file that does.

In this case, the Capfile just includes config/deploy.rb, so the latter is the one of interest. The deploy file contains a bunch of settings you need to take care of before running cap, starting with:

```
set :application, "set your application name here"
set :repository,  "set your repository location here"
```

If you aren't used to Ruby's syntax, this all will look deceptively like simple configuration. However, because you don't have to use brackets when calling functions in Ruby, each line actually is a call to the set() function in Capistrano's core:

```
set(:application, "your-app-name")
```

Set the :application variable to a name without spaces—this will be used to create a deployment directory later. Set :repository to your versioning repository's URL (in this example, we use SVN).

If you have a user name and password for SVN, set them with the lines:

```
set :scm_username, "svn-username"
set :scm_password, "svn-password"
```

Then, uncomment and set the deployment directory. If the deploy_to doesn't exist on your deployment server, Capistrano creates it:

```
set :deploy_to, "/path/to/doc/root/www/#{application}"
```

Here, we're using the application variable we set previously to set part of the deploy_to variable. This is all standard Ruby syntax, available in all Capistrano scripts, making this way of working extremely powerful and a little less cumbersome than a hodge-podge of obscure syntaxes.

Finally, we need to set the servers that will host the deployment. You can add as many servers as you like, and

the server name just has to be something that SSH understands—for example:

```
role :app, "app-server-1", "app-server-2", "app-server-3"
role :web, "192.168.1.123"
role :db,  "db-server-1", :primary => true
```

If you're just testing out Capistrano, it's worth setting the deployment location as your working machine; that way, you can learn without moving between machines:

```
role :app, "me@my-local-ip"
```

Now we're ready to ask Capistrano to set up the deployment location using the command:

```
cap deploy:setup
```

When you run this, Capistrano starts showing you what it's doing. This helps when debugging Capfiles, and it reassures you that you're doing the right thing. Whenever you connect to another server, you'll be prompted for the password, as usual, after which Capistrano will run a bunch of other commands.

After deploy:setup, the deployment directory now contains some extra directories that will allow cap to push new versions, do rollbacks and so on:

```
myapp/
      releases/ shared/log shared/pids shared/system
```

Next, we get on and deploy the application. Capistrano will check out the source, put it into releases and create a symlink to it called current:

```
cap deploy:cold
```

After this has run, take another look in the deployment location:

```
# current@ -> /www/captest/myapp/releases/20080614144520
```

This a "cold" deployment, meaning tasks that are one-time tasks are run. To deploy the application in the future, you simply use the deploy task:

```
cap deploy
```

When you've run either deploy:cold or deploy, have a look in the deployment directory and find where your source code fits into Capistrano's way of deploying things.

The deploy task replaces logging in to the server, getting the source, setting up any databases and restarting the servers. Run it a few times, and get used to that lazy feeling!

Finding More

To deploy our application, we used only `deploy:setup`, `deploy:cold` and `deploy`. The recipe has a lot more in it. To see all the available tasks, run:

```
cap -Tv
```

Much like `rake -T`, this lists all the tasks with their documentation. If you've run `deploy` a few times, play with either of the `rollback` or `rollback_code` tasks.

Each time you roll back, Capistrano simply points the symlink to the previous deployment's directory. Rollbacks can be run repeatedly until you find the stable version you want:

```
cap deploy:rollback_code
```

Your Own Tasks

Once you get Capistrano working on a Rails project, it's easy to see how it could help make your life really lazy. The same kind of tasks that wrap around Rails-specific commands can contain pretty much any command.

When you run Capistrano tasks, like deploy, you'll see various SSH commands and responses scroll by. If you have several servers, the responses will come back from multiple servers as Capistrano runs your tasks across as many machines as you need.

The deploy task replaces logging in to the server, getting the source, setting up any databases and restarting the servers.

The potential uses of this are huge—checking disk space, copying live data from clusters and running maintenance tasks—so how can we build our own tasks?

Tasks in Capistrano are defined with the following syntax:

```
desc "Short description here..."
task :name_of_function, :roles => :servers do
        # tasks is in here...
end
```

Ruby's elegant syntax often makes things confusingly simple, so let's pick it apart. The first line provides some documentation that is output when you run the following on the command line (still from the root of your project):

```
cap -Tv
```

Ruby can cope without brackets when calling functions, so the second line actually is a call to Capistrano's task function.

The first argument is the new task's name (name_of_function). The second is the set of machines on which the task will be run; this can be either :servers, :app, :db or any other collection of servers you have.

The last part, starting at do, is an anonymous function, which means that everything between do and end is executed when your task is run. You may have come across anonymous

functions in JavaScript.

A very simple task would be to run `df -h` on the remote servers to check on disk space. This isn't going to change anything on your servers, so you should feel safe running it:

```
desc "Check disk space"
task :diskspace, :roles => :servers do
        run 'df -h'
end
```

The run function simply runs the command on the remote servers. You can replace this with sudo, which also does what it sounds like—runs remote commands under sudo:

```
desc "Who hasn't been cleaning out their home directories?"
task :home_disk_usage, :roles => :servers do
        sudo 'du -sh /home/*'
end
```

If you have capified a project as we did on the Rails project in the previous section, you even can add your own custom tasks to the standard Rails recipe and change the behaviour of the Rails recipe itself. This lets you get Capistrano working just as you need it to work, and it's is good for those commands you never can remember how to run!

If you have capified a project as we did on the Rails project in the previous section, you even can add your own custom tasks to the standard Rails recipe and change the behaviour of the Rails recipe itself.

To add your own tasks to a capified Rails project, add them to config/deploy.rb using the task syntax described above. Once you have added a task, run `cap -Tv` to check whether your task was found, and then run the task as you would any other.

Tasks can call each other just like functions can, so complex tasks can be broken down into simple tasks that will keep your custom Capistrano recipes "DRY". Tasks can call each other using the normal Rails function call:

```
task :home_disk_usage, :roles => :servers do
        vhosts_disk_usage
        run "ls /home/"
end
```

You'll probably want your customised tasks to know the location in the filesystem where your project is being deployed. This is a matter of using the configuration variables we set right at the beginning, which can be done using the Ruby syntax:

```
run "tar czf ~/snapshot.tgz #{release_path}"
```

If you need additional variables, you can set them using the same syntax as before:

```
set :foo, "bar"
```

Alternatively, you can prompt the user for the variables by using the set function, but with a slightly different usage:

```
set(:deploy_version) do
        Capistrano::CLI.ui.ask "What version is this? "
end
```

The variables are used in the same way, no matter which method is used to set them.

All this Ruby should start falling into place, and by this point, you'll start thinking of Capistrano as a Ruby framework rather than a standalone application or script. If Ruby is new to you, keep going—it'll start dropping into place soon.

Finally, it's nice to keep things neat as well as DRY. All of the Rails recipes are found in the deploy namespace, which you'll notice when you run `cap -Tv`. Namespaces allow you to group tasks together, and this can be done by wrapping the tasks in the namespace command:

```
namespace :our_tasks do
  desc "The default task"
  task :default do
    restart
end

  desc "Empty logs"
  task :empty_logs do
        # ...
  end
end
```

When you run `cap -Tv`, you'll see these neatly grouped:

```
cap our_tasks             # The default task
cap our_tasks:empty_logs # Empty logs
```

Customising the Rails Recipe

Making new Capistrano tasks is straightforward, but the Rails recipe we used earlier probably contains 90% or more of what you need. In this case, it's best to customise the recipe rather than create one from scratch. We can do this by overriding specific tasks to customise the corresponding behaviour of the recipe.

I discovered this when trying Capistrano on our internal makefiles, which is where I do most of our code file management, database versioning and installation configuration loads. We use these for pretty much everything that isn't committing or editing files, so the idea that we also could deploy really quickly using Capistrano was just too tempting.

If you've read this far but are thinking, "cool, but we're not about to migrate to Rails", customisation will make sense for you because you can override the tasks that try to do Rails-specific things.

First, try `capify` on a non-Rails project, but make sure you have a config/ directory where capify can put its deploy.rb file. Once capify has run, you can start trying the various `cap deploy` tasks we did above, but it all goes wrong when Capistrano starts whining about the Rails server not being present and about a Rakefile not being present.

This is because one of the tasks, `deploy:restart`, tries to restart the Rails server. Another of the tasks tries to run `rake db:migrate`. Your project probably will support neither of these, so you should override it by adding the following to config/deploy.rb:

```
desc "Do nothing"
deploy.task :restart, :roles => :app do
  # ...do what you like here...
end
```

Intuitively, this is overriding the restart task in the deploy namespace, and everything inside the task (everything from do to end) can be edited as normal. You might want to restart your Apache server instead of the Rails server:

```
desc "Do nothing"
deploy.task :restart, :roles => :app do
        sudo '/etc/init.d/restart'
end
```

When you run `cap deploy:cold`, the Rails migrations are run to create the database. We override this to run our equivalent, which is:

```
deploy.task :migrate, :roles => :app do
        run "make data"
end
```

Conclusion

Capistrano provides a really simple way of deploying an application. It also can be used for anything involving remote servers: monitoring, arbitrary tasks, creating ad hoc backups and so on.

Thanks to Ruby's elegance, Capistrano can be extended in pretty much every way. The Rails recipe can be honed for non-Rails applications, and adding whole new recipes involves very little Ruby knowledge.

Finally, to make things even quicker, use SSH identities so you don't even have to log in to the remote servers. If you want to keep your identities somewhere nonstandard, simply add the following to your deploy.rb file:

```
ssh_options[:keys] = "/path/to/identity_file"
```

This way, you can deploy your app using `cap deploy` and nothing else—now you really can master laziness.■

Dan Frost is Technical Director of 3ev, a Web app development company in Brighton, UK. Alongside his work as a developer and technical architect in PHP, Java and all the usual stuff, he writes articles on Cloud computing, Rails and Web 2.0 technologies.

KYLE RANKIN

BILL CHILDERS

Small Laptops vs. Large Laptops

Is portability or performance king when it comes to laptops? Read below to find two Linux geeks' opposing viewpoints on the matter.

Ever since its inception, the Linux space has been full of contention. From the initial Minix vs. Linux debates to GNOME vs. KDE to distribution holy wars, it seems for any Linux question, people with strong opinions are willing to join the flame fest. In this column, we throw a little fuel on the fire with an article dedicated to promoting two conflicting points of view. This month, Bill Childers and Kyle Rankin tackle an issue near and dear to their hearts—small laptops vs. large laptops.

KYLE: I have always been a fan of small laptops. When I look back, I was probably first inspired by Penny's computer book on *Inspector Gadget*. My very first laptop was a Toshiba Libretto 50CT—a 75MHz mini-laptop about the size of a VHS tape (those of you who remember 75MHz computers should also remember what a VHS tape is, and for the rest of you, there's always Wikipedia). Ever since the Libretto, all of my laptops have had 10.6" screens or smaller, and that is my personal standard for a small laptop. I just don't understand the current trend of 15"–17" Sport Utility Laptops (SULs). Some of these SULs are almost of the size of those luggable computers of yesteryear—so big you have to get a special bag to carry them, and so big that most vendors hesitate to refer to them as *lap*tops and call them notebook computers instead. For me, a true laptop should be extremely portable and should have excellent battery life.

BILL: I used to like small laptops, but then I got better. I had an HP200LX palmtop for a long time—it was the only portable PC I could afford. That thing had an 80186 running at 8MHz and ran on two AA batteries. It had CGA graphics and was the epitome of cool. Then I stepped into the modern era and started getting systems that would let me do actual work. A system with a 15" or 17" screen isn't a luggable unless you're a little girly-man. It's a system that's capable of doing anything from standard office tasks to CAD work to playing the latest and greatest 3-D games—all the power of a desktop PC, except I can hang out on the couch. Or in my hammock. What's wrong with that?

KYLE: I wouldn't call what Bill has a "laptop" until he has someone else's lap beside him. I heard he has a Mac cube too. It's pretty sad when your desktop is smaller than your laptop.

BILL: Hey, have you seen me lately? It fits on my ever-increasing lap. Let's see you do any kind of graphics on that single-lung Yugo of a computer. Yeah, that's what I thought. It's also nice to have the added heat-generating capacity of the larger laptop in the winter months. Just put a kid by each exhaust fan and no more complaining about being cold. And, no jokes about *Star Wars* or "exhaust ports", please. It's not the *Death Star*.

KYLE: That's no laptop, it's a space station. Sure, he may be able to play video games made in the 21st century, but you should see him deathmatch with me in *Quake III*. Anyway, when his laptop's battery runs out a few seconds after booting, he hits the escape latch, and my laptop pops out like a pod full of droids from the *Death Star*. One advantage to my small laptop is I don't need a suitcase to carry it around. I use a nice, small vinyl case made for a portable DVD player. Okay, so it looks like a man purse, but it's small all the same.

BILL: I don't need a suitcase. It fits in a backpack. Okay, the backpack has an aluminum frame, but that's just for decoration. Hmm, yours cost the same as mine, yet mine can do twice as much work as yours. Who got the better value? And, I get a workout when carrying it as a bonus. Besides, when a server falls on my bag, the aluminum frame lets my computer just shrug it off like an NFL lineman. What happened when a server fell on your laptop, Kyle?

KYLE: Wow. That was below the belt. Too soon, man, too soon. You don't have to worry about servers hurting your laptop, because when they fall near it, the laptop's gravitational pull causes the server to orbit it. You can get an inexpensive tiny laptop too. So what if its specs are the same as your BlackBerry? It can run a Web browser. Don't get me wrong, I can see some advantages to having Bill's laptop on my lap, but right

now, I'd like to keep my sperm count where it is.

BILL: Hey, that's not an issue, I've had my kids. Plus, I have 4GB of RAM in my system. I may not use all 4GB, but it's nice to know I have it on tap should I decide that I need it. How much memory can Kyle shoehorn into his dinky box?

KYLE: He needs all 4GB so he can start his mail client. As a mutt user, I guess I just don't need as much RAM, but that's for a different Point/Counterpoint column.

BILL: Hey, Gmail doesn't take any more RAM than Firefox. Besides, I start my mail client only when I need to write a long message or a *Linux Journal* article. Like you said above, I have a BlackBerry for all other e-mail duty.

KYLE: For me, battery life is the key. I can sit for most of a workday on a single charge. When Bill wants to work from a coffee shop, he definitely needs his power cord. When he wants to work outside, he has to fire up his diesel generator.

BILL: Diesel generator? Hardly. My Precision M90 laptop can run for a little more than an hour on battery. While that's not your "all day" runtime, it's plenty of time for me to knock out the work I need to do before hunting for a power outlet.

KYLE: This is ultimately what it comes down to for me: when I have work to do, I don't want to hunt for an outlet, and when I work on an airplane, I like that I can fully open my laptop on the seat-back tray, even if the person in front of me leans all the way back. Today, you can get a dual-core processor even in mini-notebooks. When you combine that with a solid-state drive, you don't even have to sacrifice performance to go ultraportable. I want a laptop that fits on my lap, lasts most of the day, yet still has plenty of power for everything I do. These days, a number of laptops fit the bill—even if they don't fit Bill.

BILL: "Even if they don't fit Bill?" Wow, man, you said I hit below the belt, yet you bust out a fat joke. My main thing is, I need a system that doesn't feel like I'd break it if I looked at it wrong. It's got to have the horsepower to do anything I throw at it and be something I can haul around comfortably. Opening it on an airplane is obviously a non-starter, but I've gotten to the point where the last thing on my mind when on an airplane is *doing work*. Heck, I'm management now. I just fire up my BlackBerry's media player and put my feet up in business class while the nice flight attendants bring me drinks. You can sit back in coach and "work", Morlock. The bottom line for our readers is they need to make the decision that works best for them.■

Kyle Rankin is a Senior Systems Administrator in the San Francisco Bay Area and the author of a number of books, including *Knoppix Hacks* and *Ubuntu Hacks* for O'Reilly Media. He is currently the president of the North Bay Linux Users' Group.

Bill Childers is an IT Manager in Silicon Valley, where he lives with his wife and two children. He enjoys Linux far too much, and he probably should get more sun from time to time. In his spare time, he does work with the Gilroy Garlic Festival, but he does not smell like garlic.

EOF

The Power of Definitions

We need to do for the Net what the Free Software Definition did for software.

DOC SEARLS

As a concept, freedom is usually defined two ways, one negative and one positive. *Freedom from* is the negative. *Freedom to* is the positive. Countless social and political causes grow around the need for *freedom from*—slavery, oppression, poverty, taxation—anything that limits our *freedom to* act, move, associate, choose.

The freedoms described by the Free Software Definition (**www.gnu.org/philosophy/free-sw.html**) are all positive:

- *The freedom to run the program, for any purpose (freedom 0).*
- *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
- *The freedom to redistribute copies so you can help your neighbor (freedom 2).*
- *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.*

These freedoms are also personal: "Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software"; and "a program is free software if users have all of these freedoms."

Freedom is a profoundly human value. We are, more than any other species, devoted to originality, and we savor values that express it: intelligence, talent, choice, craft. Other animals make things too. Birds build nests, ants build hills, beavers build dams, bees build hives. But it is the nature of each to build these things the same ways as others within the species. Every human is different. What we value most in people is what makes them different from other people and what they do that's different. Freedom maximizes the scope of those differences and of our originality.

Software is one among countless other original human creations, but with an essential difference: it has no physical substance. Even the ephemeral creations we call music and speech are waves compressed within air. Software is something else. It is code. At a deeper level, it is binary math: ones and zeros.

Humans make sense of things through their bodies. Good is "up" and "light", while bad is "down" and "dark", because we are upright-walking diurnal animals. If we had the bodies of raccoons, we might say the opposite. Our worlds are full of metaphorical understandings grounded in our physical structures. When we say, "He picked my face out of a crowd", we use the metaphor *seeing is touching*. When we say we "grasp a concept", we use the metaphor *understanding is grasping*. What we do with our bodies shapes what we know in our minds and how we talk about it.

Yet software isn't physical. We need help understanding it, or we'll mess up by understanding it with misleading metaphors (for example, that it's a packaged good, like cereal). This is why we need to start with deep insights into software's nature, and into connections between that nature and our own. The Free Software Definition provides those. So does the companion concept of copyleft (**www.gnu.org/copyleft/copyleft.html**), which protects the liberties inherent in free software. This is why Richard M. Stallman calls free software a "social movement", while positioning open source as a "development methodology" (**www.gnu.org/philosophy/open-source-misses-the-point.html**).

Today we live in a networked world not only filled with free software and open-source code, but also increasingly organized and defined by it. This has caused problems of perception that are similar to those that required the Free Software Definition 25 years ago.

The Internet, for example, has become a form of infrastructure, yet it lacks the physical qualities that have defined familiar forms of infrastructure in the past. Although it embodies qualities that are similar to real estate ("sites" and "domains" with "addresses") and transport systems ("pipes" and "highways"), its supportive capacities are categorically limitless. This is why restricting our understanding of the Net to real estate and transport metaphors is a mistake.

Ask ten people to tell you what the Net is, and you'll get ten different answers. The same won't happen if you ask them what a road or a water system is. Or a phone or cable TV system. An irony in that last case is that telephony and television are now forms of data. In February 2009, here in the US, analog broadcast television will go the way of the steam locomotive. All TV broadcasting will be digital. Yet it will still be represented in familiar analog-like ways, with "channels" from "networks" and so on. Lost is the fact that these things are coming to homes by digital signaling using Internet protocols.

Where I live in California, burying service underground is a huge chore. The ground is rocky, and underground service culverts need to be eight feet deep, so there's room to keep electrical, cable TV and telephone services separate, just like they are on the poles above the ground. Yet the old analog phone wiring and coaxial TV cabling are no longer required. Being just data, telephony and television can be carried on fiber-optic cabling. And that cabling can run right next to high-voltage electrical wiring, as fiber-optic signaling is unaffected by proximity to electric current. The smart thing to do, then, is to trench the dimensions required for electric service, and run the rest over fiber-optic cabling alongside it.

But we're not ready for that, mostly because we still see the Net as a grace of telephone or cable company carriage—not as something that's essentially free and open. Yes, capital outlays are required, but the upsides of making those outlays are incalculably large, for everybody.

So our problem with the Net is very similar to the problem we had with software up to a quarter century ago: it's seen as essentially proprietary. We think of it as something owned and/or controlled by a big company and delivered as a "service" that we "access". Although that's how most of us "get" the Net today, that understanding is at odds with the Net's free and open nature, and with our own as sources of value for the Net.

What we need now is a definition of the Net that is as deep and useful as the Free Software Definition's is for software. Without that definition, the Net will continue to be defined mostly by government, and by phone and cable companies.■

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

GPU Computing

More GFLOPS, Fewer Watts!

High Density Tesla Clusters

WhisperStation™ –PSC with Tesla C1060 GPUs
Computing Clusters with Tesla S1070 - 4 GPU Servers

GPU Performance:

- 1 TFLOPS per GPU
- 4 GB DDR3 per GPU
- 102 GB/Sec Bandwidth

CUDA SDK

- Standard C language
- Standard numerical library- FFT and BLAS

36 GPUs + 36 CPUs
Computing Cluster in 24 U

FireStream™ Workstations & Servers

WhisperStation™ – FS with FireStream 9250 Stream Processors

AMD
Solution Provider
PLATINUM

AMD FireStream 9250

Stream Processor Performance

- 1 TFLOPS
- 1 GB DDR3
- Single Wide Card /150 Watts
- Double Precision Floating Point Hardware

Stream SDK

- Brook+ Open Source Compiler
- ACML and CAL Math Libraries

Microway®
Technology you can count on℠

508-746-7341
microway.com

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