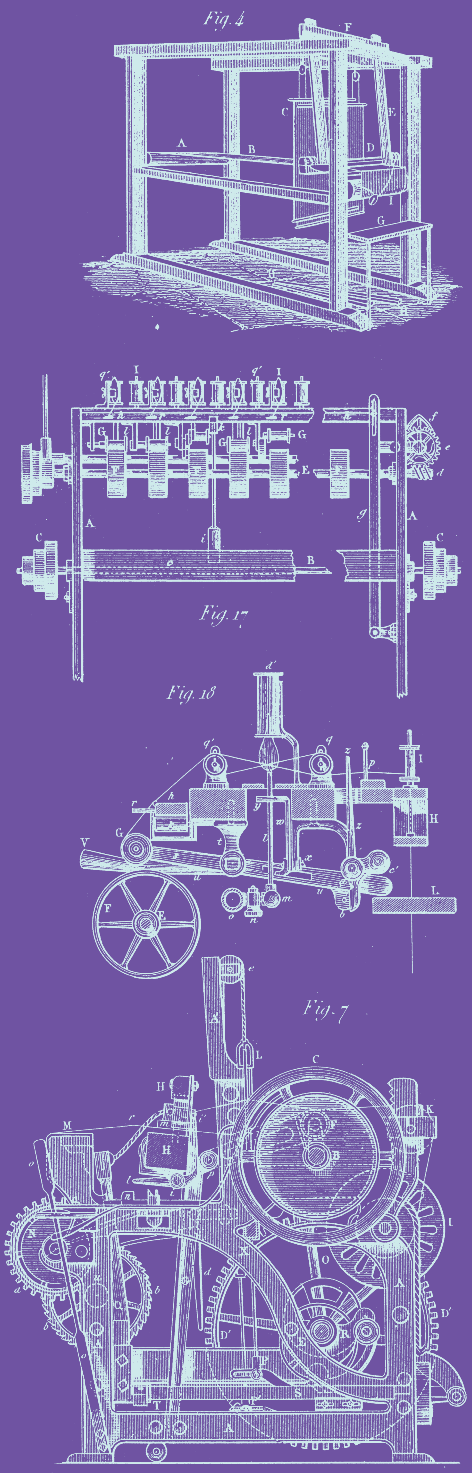
JUNE 2006 VOLUME 31 NUMBER 3

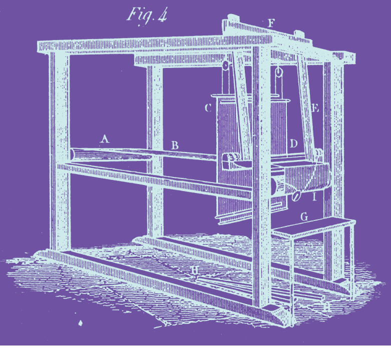
;login:

THE USENIX MAGAZINE



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



2nd Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI '06)

JULY 6–7, 2006, San Jose, CA, USA
http://www.usenix.org/sruti06

2006 Linux Kernel Developers Summit

JULY 16–18, 2006, Ottawa, Ontario, Canada
http://www.usenix.org/kernel06

15th USENIX Security Symposium (Security '06)

JULY 31–AUGUST 4, 2006, Vancouver, B.C., Canada
http://www.usenix.org/sec06

First Workshop on Hot Topics in Security (HotSec '06)

JULY 31, 2006, Vancouver, B.C., Canada
http://www.usenix.org/hotsec06

2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '06)

AUGUST 1, 2006, Vancouver, B.C., Canada
http://www.usenix.org/evt06

First Workshop on Security Metrics (MetriCon 1.0)

AUGUST 1, 2006, Vancouver, B.C., Canada
http://www.usenix.org/metricon06

3rd Workshop on Real, Large Distributed Systems (WORLDS '06)

NOVEMBER 5, 2006, Seattle, WA, USA
http://www.usenix.org/worlds06
Paper submissions due: July 7, 2006

7th Symposium on Operating Systems Design and Implementation (OSDI '06)

Sponsored by USENIX, in cooperation with ACM SIGOPS

NOVEMBER 6–8, 2006, Seattle, WA, USA
http://www.usenix.org/osdi06

Second Workshop on Hot Topics in System Dependability (HotDep '06)

NOVEMBER 8, 2006, Seattle, WA, USA
http://www.usenix.org/hotdep06
Paper submissions due: July 15, 2006

ACM/IFIP/USENIX 7th International Middleware Conference

NOV. 27–DEC. 1, 2006, Melbourne, Australia
http://2006.middleware-conference.org

20th Large Installation System Administration Conference (LISA '06)

DECEMBER 3–8, 2006, Washington, D.C., USA
http://www.usenix.org/lisa06

5th USENIX Conference on File and Storage Technologies (FAST '07)

Sponsored by USENIX in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee, and IEEE TCOS

FEBRUARY 13–16, 2007, San Jose, CA, USA
http://www.usenix.org/fast07
Paper submissions due: September 4, 2006

4th Symposium on Networked Systems Design and Implementation (NSDI '07)

APRIL 11–13, 2007, Cambridge, MA, USA
http://www.usenix.org/nsdi07

For a complete list of all USENIX & USENIX co-sponsored events, see http://www.usenix.org/events

contents

OPINION

SYSADMIN

SECURITY

COLUMNS

BOOK REVIEWS

USENIX NOTES

;login:

VOL. 31, #3, JUNE 2006

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Lisa Camp de Avalos
Casey Henderson

TYPESETTER
Star Type
startype@comcast.net

USENIX ASSOCIATION
2560 Ninth Street,
Suite 215, Berkeley,
California 94710
Phone: (510) 528-8649
FAX: (510) 548-5738

http://www.usenix.org
http://www.sage.org

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bi-monthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

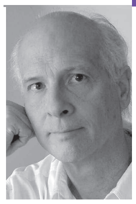
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $115 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

RIK FARROW

musings



rik@spirit.com

IT IS EASY TO BECOME A VICTIM OF future shock. I just read an ad in *New Scientist* for "gene silencers, suitable for *in vivo* work," by mail order. Once I had decided that the ad was real and not a joke, I next wondered whether any of my own genes deserved silencing via some mail-order sRNA sequences.

The ever-increasing scourge of Windows viruses, spyware, and rootkits provides another jolt of future shock. I've heard of people unplugging from the Internet rather than continue to deal with the plague of adware for porn sites, identity theft, and the requirement that they be clever enough to deploy at least two types of both anti-adware and antivirus software to be truly safe; some simply install some other OS, but I digress.

Sometimes Windows systems can become so infected with malware that the only way to secure them is to go through formatting and reinstallation. In *eWeek* (http://www.eweek.com/article2/0,1895,1945808,00.asp), an article quotes Mike Danseglio, program manager in the Security Solutions group at Microsoft, as saying that the only reliable solution is to rebuild from scratch. I'm pretty sure that most of you are not surprised to read this.

Securing Windows is not a simple problem. If it were, Microsoft would have laid this problem to rest years, and many billions of dollars, ago. Windows Vista, which makes some real progress in providing a more secure Windows environment by making it possible to use the system without being in the Administrator group and by running some device drivers unprivileged, will certainly help. But Vista has been delayed until at least January 2007. And even these changes will not address Windows' biggest issue, that of complexity. Real solutions to security issues will not be possible until Microsoft is willing to give up backward compatibility (see the December '05 opinion article by Dan Geer).

The Internet Is Broken

In a disturbing article in MIT's *Technology Review* (December 2005, http://www.technologyreview.com/InfoTech/wtr_16051,258,p1.html), David Talbot suggests that the "Internet is broken" and backs up this notion with support from David Clark, an early and key architect of the Internet.

Talbot writes that worms, spam, and phishing are evidence that the Internet needs replacing and that patching won't work. Besides confusing the Internet with end systems, Talbot does make some good points. The Internet was designed for just a few hundred systems, systems that were not mobile, and security was not even considered. Now, with the number of Internet-connected systems in the hundreds of millions, some of which are truly mobile systems (cell phones and PDAs are examples), the original Internet protocols seem a poor match with our current installed base.

Some of the architecture solutions suggested by Clark in Talbot's article make a lot of sense, whereas others just grate on my nerves. His first priority is giving "the medium a basic security architecture—the ability to authenticate who you are communicating [with] and prevent things like spam and viruses from ever reaching your PC." Whoa, there, Dr. Clark. Spam already comes from compromised systems, and certainly spam relay software will borrow the identity of the victim. Will we submit to iris scans in the future before we can send an email? And how in the world will a new Internet design defend vulnerable systems from exploitation?

There's more. The second point is to make the architecture "practical by devising protocols that allow Internet service providers to better route traffic and collaborate to offer advanced services without compromising their businesses." That part hints at creating a new, tiered Internet, one that permits ISPs to control traffic, giving those who pay for special services special access. Debate about the issue of content neutrality has arisen around the U.S. House bill known as the Barton Bill, after Representative Barton who wrote it, with Congress so far siding with neutrality. That is, large ISPs, such as AT&T, should not be able to filter out competing content, for example, paid music downloads from Google or Apple iTunes. And ISPs cannot add tariffs that make those offerings noncompetitive with the ISP's or their parent company's own offerings.

The whole idea of having telephone companies controlling the content their subscribers can receive strikes me as scary. Visions of *1984*, *V for Vendetta*, and good ol' Ma Bell running your communications media again just don't sit well with me. As if the telephone company has done a great job so far at controlling denial-of-service attacks and spam (e.g., those sales calls that used to occur at dinnertime), attacks (the random person who calls your phone number and starts screaming profanity at whoever answers, or the heavy breather who calls when your children are home alone). And then there's the telephonic version of phishing, where scammers call up elderly people and social-engineer them out of their savings. Sure, we trust the phone company to protect us and our ability to access information as we chose—just kidding.

But wait, Clark has two more points. I think the next one is actually very important: Allow future computing devices of any size to connect to the Internet. Right now, support for mobile IP, that is, the ability to maintain IP connections while you move from net to net, is extremely limited. Routing currently depends upon the network portion of your IP address, and if your device moves between networks, your IP address and your route must change. Changing your IP address plays havoc with protocols that embed the IP address in data, as well as killing any existing TCP connections. Most solutions focus on using a proxy that forwards your traffic to your current IP address, a clumsy solution that relies on some third party, the proxy service, as well as support for the applications you want to use, to work.

Mobile IP gets us right back into the territory of the telcos again. Imagine that we do somehow create a new Internet that supports real mobility.

Then, as you work, walk, or drive through cities with free WiFi, why use a costly cell phone, when you can use VoIP for free? There are already WiFi-enabled PDAs and cell phones, but not many. And most of these rely on Windows CE for their operating systems (what a scary idea). True, mobile IP will certainly impact telcos, but having this capability is really crucial to any new Internet design.

Finally, Clark suggests adding technology that makes the network easier to manage and more resilient. Like the third point, this is another strong argument for a reinvented Internet. I don't think Clark is talking about managing the Internet at subnet scales, but, rather, he is addressing the larger issues involved in managing the Internet, the network of networks. Back in the nineties, I would hear stories of how one large ISP would route its traffic over another ISP's network, preserving its own bandwidth, while taking advantage of a competitor. Today, these issues get resolved (more or less) through the careful configuration of BGP; still, they are not easy to solve. There are also issues such as slashdotting, DDoS, and other traffic-flow issues that really have no widely accepted management solution today.

Stupidity

Now, do I really believe that a new Internet will solve the security issues we see with today's Internet? Not at all. The real problems sit on people's desktops, and these involve insecure operating systems and applications. I believe that if it were possible to filter out all dangerous content, there would be a thriving market in doing so today. You have certainly observed that there is a huge market in selling incomplete and only partially effective solutions to viruses, spam, spyware, adware, rootkits, and other malware. I think you can compare the problem of blocking malware to the halting problem—in other words, it is an insolvable problem.

In the *eWeek* article, based on a presentation made at the InfoSec World Conference, Danseglio also said, "Phishing is a major problem because there really is no patch for human stupidity." Hmmm, we do let stupid people have bank accounts, right? They drive cars, pay taxes, raise children, own and use weapons; but we can't trust them to use their computers properly? Somehow I think this argument is specious. If using your computer results in damage to your bank account, is it your fault? Or is it the fault of the software that cannot parse email headers, validate domain names, or at least offer clear warnings such as "This Web site does not appear to be affiliated in any way with [fill in your financial institution here]." Or is it the fault of the underlying software that made it easy to install the spyware that stole your identity? Stupidity? If cars were as unsafe to use as today's computers, most people would still be walking.

When I learned how to fly small airplanes, I also learned that these same airplanes are designed in ways that make them safer than they might otherwise be. Stability is a big concern. Modern warplanes are inherently unstable, requiring clever fly-by-wire systems to make it possible for even well-trained pilots to fly them. Cessnas, in contrast, are designed so that they are stable, difficult to stall, and easy to land. Aircraft designers do this so that their product will be widely accepted and safer to use. Too bad our operating systems vendors haven't figured this out yet.

The Architecture Is Broken

I do believe that Microsoft, Sun, and the Open Source developers are working with a serious handicap. They are building and patching operating systems designed for hardware that is obsolete—hardware that was designed for another era. Our hardware architecture resembles that of ’60s mainframes, designed to support an operating system running a time-sharing system. We don’t run time-sharing systems anymore, and we haven’t for years. Most computers today have a single user, but the operating system designers have not come close to appreciating this fact. Remember that authors of the UNIX system quickly morphed the original, single-user version into a multiuser system, and every UNIX or Linux system today shares that legacy.

The single most dangerous and commonly exploited application today is the Web browser. Web browsers are purposely designed to execute remote code in the context of the single user of the system. No security system based on time-sharing notions, the Orange Book, Multi-Level Security (MLS), SELinux, or AppArmour is going to protect a user against code that that user has elected to execute. Today, reading HTML-formatted email and browsing the Web are the most insecure activities you can engage in. And the operating systems, and the hardware they rely upon, really don’t make the Web, and by extension the Internet, a very safe environment.

Time-sharing systems needed a method for isolating processes being executed by different users. Memory management does this and is itself controlled by software running at the highest privilege level, sometimes called ring 0. In today’s operating systems, all of the operating system—an enormous, complex program requiring megabytes of memory just for the code —runs at ring 0. A single error here compromises the entire system—and if this isn’t a bad way to design a system, I don’t know what is. But the hardware was designed for just such a system.

I would certainly like to think that the current environment is ripe for new designs and new ways of thinking about operating systems and security. But system architecture is not going to change easily, and neither are the operating systems that have been designed for these architectures.

The Lineup

But that’s enough bellyaching. In this issue of *;login:*, we start off with an opinion piece by Mark Burgess. Mark explains the meaning of autonomic computing, what it means today, and where it is going, in what I hope will initiate a series of articles about this topic.

In the Sysadmin section, Kurt Chan has satisfied my long quest for someone who can authoritatively explain the differences among different types of disk drives. I’ve heard people say that SCSI is dead and that SATA will supplant the more expensive SCSI drives. Chan explains that the problem with this analysis is that it doesn’t slice the problem correctly. While SCSI drives will be replaced by SAS (SCSI over Asynchronous Serial), the real divisions between drives have to do with how they will be used, not just the interfaces used to connect them. And no, SATA will not replace SAS. If you don’t believe me, just read Chan’s article.

Kirk McKusick has a different perspective about drive types, and he has contributed a short article that adds another way of looking at the drive types. Kirk sees the world from the filesystem and device-driver writer’s perspective, and this is relevant in its own way.

Next, Tom Haynes discusses the configuration and uses of ZFS, Sun’s Zone File System, coming to an Open Source system near you soon and already available in Solaris 10. Stefan Büttcher delves into a different filesystem aspect, indexing. Büttcher, whose paper about Wumpus was presented during last December’s FAST workshop, explains the design decisions behind

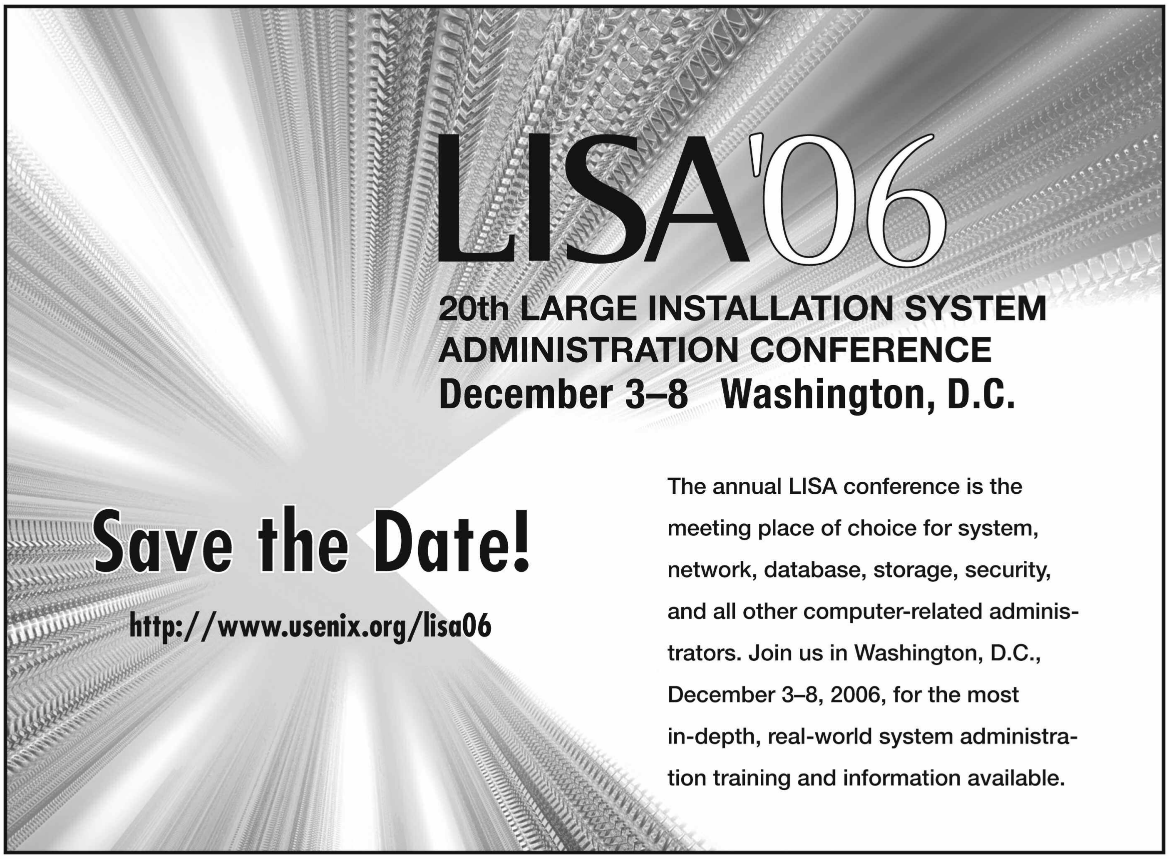
Wumpus, while explaining important issues about filesystem indexing on multiuser systems and systems using networked file systems.

In the security section, we start off with an article by Pablo Neira Ayuso, one of the key Netfilter developers. Neira explains the architecture supporting Netfilter's Connection Tracking subsystem, the foundation for stateful filtering in Linux kernels. Then Markos Gogoulos and Diomidis Spinellis report on their research into using live CDs for penetration testing.

This issue, as has become the custom, ends with articles by our regular columnists and book reviews.

I have, sadly, become accustomed to complaining about security. I recently wrote an article for a newsletter in which I pointed out that the proliferation of security vendors clearly demonstrates our collective failure to produce secure systems. Somehow, I don't think I will notice, or even believe, a little ad found in a science magazine that advertises that the solution to desktop security, and server security, can be obtained via mail order.

The solution would be a lot easier if only we were willing to stop using the software we rely upon today and start over. But Word has become the opium of computer users, and breaking the habit is not going to be easy. Perhaps a solution like ODF (Open Document Format) will be the methadone that eases us away from the addiction.



MARK BURGESS

autonomic computing—the music of the cubes

Mark Burgess is professor of network and system administration at Oslo University College, Norway. He is the author of cfengine and many books and research papers on system administration.

Mark.Burgess@iu.hio.no

AUTONOMIC COMPUTING IS A PHRASE invented by IBM to sell mostly existing technologies for automation in the marketplace. IBM has led a corporate procession away from centralized management technologies toward self-maintainance. They were among the first companies to see light at the end of the system administration tunnel.

Over the past few years, "network management" (i.e., system administration) research has been taking giant strides in a random walk of hunting and gathering, taking and improving upon ideas about automation, some of which have been known for thirty years or more. Researchers have been putting these ideas into some kind of context or practice, and the activity has been substantial. Some of those ideas have been developed and used in the USENIX community through research in configuration management and policy-based automation. Some of this development genuinely comes from the big corporations, including IBM, HP, Motorola, and now the academic EMANICS Network of Excellence in Europe. We're all good friends, ignoring the hype and working on the real issues, while the marketing departments justify the work with colorful banners.

So, what does autonomic mean? The name gives us a clue: it is formed from two Greek roots: *autos,* meaning "self," and *nomos,* meaning "the law." In other words, it is about self-governance—or, as it is sometimes paradoxically expressed, self-management. As such, it brings together three ideas: automation, decentralization, and autonomy of decision. If you like, it is a self-help program for computers.

While the corporate autonomic computing campaign has been, for end users, more of a triumph of XML style over substance, the underlying idea has been taken very seriously by multitudinous corporate and academic researchers, anxious to see their work realized in the marketplace.

But if we have known about these ideas for a long time, why the big song and dance now? To understand the whys and wherefores one needs to take a step back from the servers to look at corporate politics and produce, because the bandwagon is a commercial development, not a research development.

The traditional view of system management, in many large organizations, has been dominated by the monopolistic telecom dream of world domination. Everyone has to do as the service providers say; don't they? So the telecoms have believed they could simply create the usual management position to monitor systems and issue instructions to make magic happen. Everything would then be "managed" and hence hunky-dory. The IETF and DMTF, formed mainly from the genes of telco leviathans, gave birth to TMN, SNMP, NETCONF, etc., based upon this belief in the power of central authority.

But the thing that none of them could ignore is that there are, in reality, now more than five computers in the world. In fact, there's one in every pocket and on every wrist, not to mention desk, car, aircraft, etc. ad nauseum. Like it or not, computer *ownership* is now utterly decentralized, and information privacy rests on the lips of every global citizen. There is a feeling that one should not surrender to authority. Neither these personal computing devices nor their owners are looking to open their hearts and minds to a just and angry trunk provider. The spirit of the times is rather to be found in the rustling of the leaf nodes.

Today the computing industry is still having a hard time letting go of this central-command mentality. Companies such as IBM and HP are studying "control theory," or self-regulation, as it is known from electrical engineering, in the same breath as they speak of autonomics. No one quite believes it all yet—their corporate consciences are whispering to them that making machines run themselves is going just a little too far. Who will be pushing the buttons?

It is somehow reminiscent of firewalls and intrusion detection. Do you centralize security or try to avoid bottlenecks?

> Let's isolate systems to keep them safe!
>
> What do you mean, you need to talk to them?
>
> All right, we'll introduce a manager in the network to take care of security: a firewall will keep us safe!
>
> What do you mean, the host opened up a back door through a wireless LAN?

Immediately people think management means a centralized point of responsibility. Does it? Of course it doesn't. Try explaining the schools and shoals and swarms of fish as they swim in perfect coordination, or colonies of ants. Do birds fly around just dying for promotion into a magnificent managerial role? Let me lead the flocking way! No, they get along fine without having to invent the leash.

> Don't you need the management privilege to violate private boundaries to get that all-important deep knowledge and perspective?

Ants do not rely on satellite communications or radar to navigate. They manage vastly complex tasks from a low-cost local perspective, by interacting cheaply with immediate and local information through smell, not through gigabytes of collected data. Hurrah for the war against tera! We can be economical if we don't try to overmanage or micromanage.

Managers think they can command excellence, but systems never achieve crystalline perfection. They are not quite ecological slime, but they are getting close! Have you seen the pictures of the Internet lately? Maintaining software and systems is something like trying to solve hundreds of Rubik's cubes—just when you think you've fixed a single face, you push all the other faces out of synch.

So is autonomics important? Yes, it is, and especially one aspect of it that truly is new. It is not automation nor yet distribution. I would say that *autonomy* is the key challenge to the future of computer management or system administration. We have to unlearn what we are used to and ask: What is a system? Where are its boundaries? Who has political control over it? How can be get users to behave nicely when their private devices come together?

My own work, around cfengine, has always placed autonomy as an important principle, more out of a sense of belief in local adaptability than of an awareness of the onset of the future. But it seems clear that it was a lucky guess as far as the proliferation of personal computing is concerned.

You might not really want to speak of *management* when it's just one mobile phone we're talking about, but *maintenance* is a reasonable word. In an autonomous world, the process of fitting in with our neighbors will borrow more from symbiosis and swarming than from command and strike force. In the end the swarms of unfinished Rubik's cubes might fall into similar patterns simply autonomically. And that curiously imperfect state of consistency might be the best we can hope for.

Thanks to USENIX & SAGE Supporting Members

Addison-Wesley Professional/
Prentice Hall Professional

Ajava Systems, Inc.

AMD

Asian Development Bank

Cambridge Computer Services, Inc.

EAGLE Software, Inc.

Electronic Frontier Foundation

Eli Research

FOTO SEARCH Stock Footage and Stock Photography

GroundWork Open Source Solutions

Hewlett-Packard

IBM

Infosys

Intel

Interhack

The Measurement Factory

Microsoft Research

MSB Associates

NetApp

Oracle

OSDL

Raytheon

Ripe NCC

Sendmail, Inc.

Splunk

Sun Microsystems, Inc.

Taos

Tellme Networks

UUNET Technologies, Inc.

It is with the generous financial support of our supporting members that USENIX is able to fulfill its mission to:

- Foster technical excellence and innovation
- Support and disseminate research with a practical bias
- Provide a neutral forum for discussion of technical issues
- Encourage computing outreach into the community at large

We encourage your organization to become a supporting member. Send email to Catherine Allman, Sales Director, sales@usenix.org, or phone her at 510-528-8649 extension 32. For more information about memberships, see http://www.usenix.org/membership/classes.html.

KURT CHAN

a comparison of disk drives for enterprise computing

Kurt Chan is a Technical Director at Network Appliance, responsible for storage subsystems.

kurtc@netapp.com

FOR END USERS, THE FIVE MOST externally visible characteristics of a disk drive are capacity, price, interface type (e.g., SCSI, ATA, Fibre Channel, SATA), performance (e.g., access time, I/Os per second, sustained transfer rate), and reliability (e.g., MTBF or unrecoverable read error rate). When evaluating a drive for a particular application, these attributes carry varying weight. We'll examine how these attributes are related in real disk drive implementations, what applications are best suited to specific drive types, and what the future holds for disk storage in the enterprise.

Disk Drive Economics

The disk drive business has undergone heavy consolidation over the past decade, and even the survivors operate on relatively thin margins compared with those who integrate drives into enterprise systems. Here's a chart of some disk drive manufacturer gross margins for 2005, along with some major storage integrators [1]:

Disk Drive Manufacturer	*Gross Margin*
Maxtor	11.1%
WD	18.4%
Seagate	25.1%

Disk Drive Integrator	*Gross Margin*
Dell	17.8%
EMC	53.7%
NetApp	61.3%

Disk Drive Integrator	*Units (2004)*
Dell	16.1%
EMC	0.6%
NetApp	0.5%

Source: *IDC Worldwide Disk Storage Systems Market Forecast and Analysis*, 2002-9

Note that although EMC and NetApp have superior gross margins, Dell accounted for almost 15 times the unit shipments of both companies put together—16.1% market share versus 1.1%. This is because the volumes of the consumer and desktop markets dwarf the volume associated with the enterprise storage market. Furthermore, overall enterprise HDD revenue has remained relatively flat over the past 3–4 years, and cost/GB enterprise disk pricing has dropped about fourfold in the past four years. This means that, to maintain revenue, drive vendors must offer higher and higher capacity drives for about the same unit cost, which explains the speed at which we learn new Greek prefixes. (Terabyte disks will be commonplace by the end of the decade, and petabyte configurations are now possible.) These economic factors will be important in understanding the target designs of various drive types.

Classifying Disk Drives by Application

While a growing number of disk drives are finding their way into mobile and consumer appliances (e.g., notebooks, music and video recorders, personal electronics), disk drives for the computing industry are segmented into enterprise and desktop applications. Also arising is a new segment called “nearline enterprise” that combines some of the attributes of the classic desktop and enterprise markets.

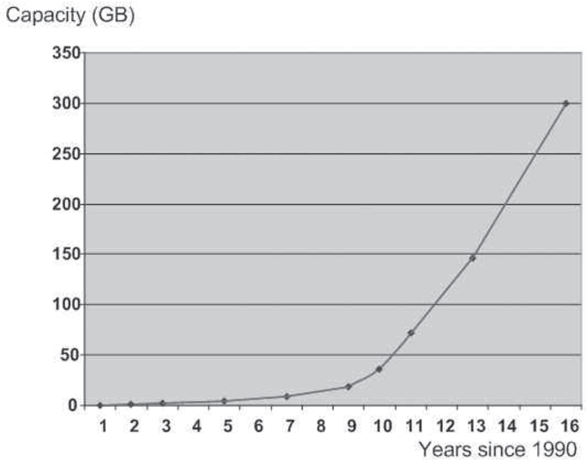
Application Attribute	*High-Performance Enterprise*	*Nearline Enterprise*	*Typical 2006 Desktop*
Rotational speed (rpm)	15,000	7,200	5,400–7,200
Interface	FC, SAS	SATA	SATA
Avg Power:			
operating	18–20 W	10–13 W	8–12 W
idle	12–14 W	7–9 W	6–9 W
Nonrecoverable read errors per bits read	1 sector per 10^{15}–10^{16}	1 sector per 10^{14}–10^{15}	1 sector per 10^{14}
Serial link rate (Gb/s)	2–4 FC, 3.0 SAS	3.0 SATA	1.5–3.0 SATA
Noise (ISO 7779, bels)			
idle	3.5–3.8	2.8–3.4	2.5
performance seek	4.3–5.9	3.5–3.9	3.1–3.7
Capacities (2006)	37–174 GB	320–500 GB	160–320 GB
Performance:			
sustained transfer	58–98 MB/s	35–65 MB/s	32–58 MB/s
average seek	3–4 ms	8–9 ms	8–10 ms
Relative price per GB	5–10x	1.5x	1x

Notable niches include 300 GB, 10k rpm FC, and 150 GB; 10k rpm SATA drives exist, but are not as broadly sourced among vendors.

Capacity

Although the capacities of each drive category will change over time, the lowest capacities are found in the enterprise markets, where performance is more important than capacity. The highest capacities are found in the nearline market, where disks are sometimes used for secondary storage, replacing tape for disk-to-disk backup applications or for storing less frequently used data that still require online access. The desktop market, where cost/GB is the lowest, focuses on the capacities—these typically lie somewhere between performance and nearline enterprise capacities, and strong discounting takes place as inventory is purged from one capacity generation to the next.

Even though SCSI/FC disk drive capacity has been growing exponentially at a compound annual growth rate of 53.2% over the past fifteen years, it has slowed dramatically over the past five.[2] Whereas capacity would normally double every 18–19 months given trends from the early 1990s, the last five years of data indicate we are doubling capacity only every 29–30 months. One of the reasons for this change is the need to balance reliability with capacity. As a product generation matures, the various electromechanical margins are eroded as capacities and performance increase. Decreasing head fly heights and increasing spindle speed and platter count all make it more difficult to maintain MTBF and unrecoverable error rate (UER) specifications. The ceilings encountered in recent years are partly related to maintaining the same or better reliability with disk drives spinning 50–100% faster, thus generating more heat and mechanical stresses. This is another reason why the highest capacity drives are not found in the performance enterprise, but, rather, in the desktop and nearline categories. This year, perpendicular recording will provide a new generation of drives with more margin, and capacity growth should improve as a result.



Year	*Capacity (GB)*	*Rate of Increase*	*Annual Rate of Increase*
1990	0.5		
1991	1	100.0%	100.0%
1992	2	100.0%	100.0%
1994	4	100.0%	41.4%

Source: "Why Tape Won't Die," *Enterprise Storage Forum*, June 16, 2005

Power

Power is another area of differentiation and generally increases in proportion to performance. Lower-speed drives consume less power, make less noise, and generate less heat, placing less demand on air conditioning. But they also provide lower sustained transfer rates and I/Os per second compared to performance enterprise drives. However, for many applications that do not demand high I/O per second rates, SATA drives are often a better choice. Archived email, digital photographs, or archived customer records do not require high transaction rates, and using high-performance enterprise drives for such bulk information can be wasteful. Although power differences may not seem significant, if a large disk user such as Google or Yahoo had 1,000 drives running 24/7, the difference in electricity costs between performance and nearline disk drives could amount to more than a quarter of a million dollars a year in electricity for power and cooling.

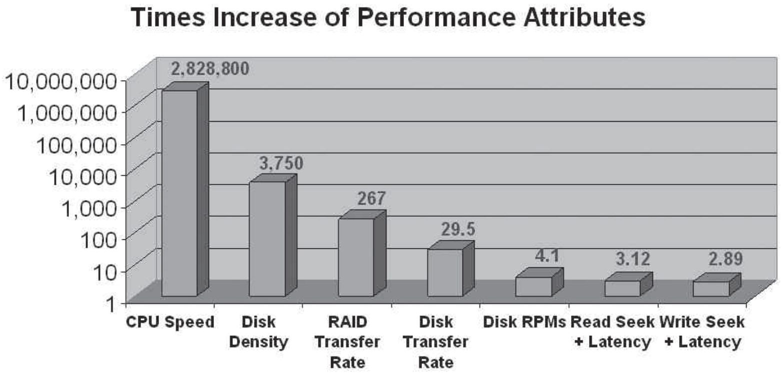
Reliability

A UER on SATA of 1 in 10^{14} bits read means a read failure every 12.5 terabytes. A 500 GB drive has 0.04E14 bits, so in the worst case rebuilding that drive in a five-drive RAID-5 group means transferring 0.20E14 bits. This means there is a 20% probability of an unrecoverable error during the rebuild [3]. Performing the same calculation for a 174 GB enterprise drive with a UER of 1 in 10^{15}, we get a 1.2% probability of data loss. Although SATA is expected to reach a UER of 10^{-15} by 2007, and enterprise drives 10^{-16} in the same timeframe, corresponding to 2% and 0.1%, respectively, this is still unacceptably high for many enterprise applications.

This phenomenon is not going away—as drives get larger, the problem becomes worse because there are more bits to move in a rebuild. Furthermore, product reliability can vary greatly among vendors as well as among product families from the same vendor. Instead of relying on advertised average failure rate (AFR), MTBF, and UER numbers from vendors, storage integrators tend to use their own empirical information to assess overall product quality and determine the necessary data protection methods. Since all drive integrators work with basically the same disk drives, what storage integrators are looking for is a means of making customer data availability more immune to drive reliability. Whereas reliability continues to be an important metric to control support and maintenance costs, measures such as double-parity RAID, full mirroring, rebuilding only used capacity, end-end checksums, and background media scans can help make the differences in reliability among drive families less important when it comes to ensuring overall customer data availability.

Performance

Disk drive performance in general has been relatively static compared to CPU clock speed and areal density growth, but it remains a meaningful differentiator between FC/SAS and SATA drives.



All 15k rpm drives and almost all 10k rpm drives available today have only FC, SCSI, or serial attached SCSI (SAS) interfaces. Enterprise drive suppliers in general have been reluctant to rush toward providing 10k speeds in a SATA drive, to avoid cannibalization of their high-margin markets as well as to keep costs low for their volume markets. Because many OLTP enterprise applications are limited in performance by IOP rates, by putting low-speed drive assemblies behind SATA interfaces the drive industry will remain segmented, barring any new designs that fill the gap between low-cost SATA and high-performance SAS drives. It might be possible to construct a high-performance OLTP system with “half-speed” drives, but the infrastructure and connectivity costs would make the solution impractical at the high end. However, important issues for the disk industry involve the amount of enterprise data being created that does not demand high-performance FC/SAS storage, and whether or not end users will begin matching their data to storage attributes using Information Lifecycle Management (ILM) and other tools to help lower their disk hardware costs. The recent growth of nearline SATA storage is evidence that users are becoming more aware of these options.

Rotational vibration plays a role in performance as well. Mechanical interferences caused by vibration patterns increase seek time, since it takes longer for heads to settle on track in the presence of severe vibration. Also, if the actuator vibrates off track, this can result in read retries and aborted writes. Since device driver timeouts can be lengthy, even a small number of retries can prove costly to performance. It’s not unusual to see desktop drives drop to 50% of their nominal peak performance in the presence of 10 rad/s^2 of vibration, whereas enterprise drives might see no drop-off until around 15 rad/s^2 and might hit 50% of their nominal peak performance at 40 rad/s^2. Rotational vibration is also exacerbated by random operation and bursty workloads—the kind often found in enterprise high-OLTP traffic applications. More stringent rotational vibration specifications may be needed for SATA cabinets to ensure that performance remains at expected levels.

Finally, tests have shown that drives designed for desktop workloads can fail more frequently when exposed to heavier workloads. When Seagate performed accelerated life testing on three groups of 300 desktop drives while exposing them to high-duty-cycle sequential workloads, these drives failed twice as often as when they were exposed to normal desktop workloads. And, when exposed to random server workloads, they failed four times as often [4]. If nearline systems are deployed in the wrong workload environments without the proper data protection precautions, loss of data availability could result.

Interfaces

Over the past five years there has been a rapid adoption of serial disk interfaces over their parallel counterparts. Virtually no new computer designs are incorporating parallel SCSI or ATA today, and disk drive manufacturers will ramp down their production of parallel interfaces as demand lowers for legacy applications. The move to serial interfaces has been motivated by several factors: the inability of scaling parallel cables in both speed and distance, the cost and bulk of parallel cables and connectors in embedded desktop applications, the larger number of devices supported by serial protocols, and the ability to support more than one disk type over the same wire protocol.

FIBRE CHANNEL

The most broadly networked disk protocol is Fibre Channel. At the high end, 256-port nonblocking switches are available from multiple vendors, with 4 Gb/s and multi-kilometer distances supported through various copper and fiber-optic cabling options. At the low end, Fibre Channel Arbitrated Loop switches are available for interconnecting disk drives within RAID or disk enclosures over high-speed backplanes. Although the Fibre Channel architecture makes it convenient for connecting drives directly to initiators without protocol conversion, Fibre Channel as a storage system interface carries more momentum than as a disk drive interface. Part of the reason is the realization that the disk drive doesn't need to have as much network intelligence as is required by the Fibre Channel standards. Furthermore, bridging and RAID technologies are becoming more prevalent, allowing Fibre Channel to be used where its distance and multi-initiator capabilities are best leveraged—at the server interface—while allowing the disk drive interface to be chosen independently.

Fibre Channel as a disk drive interface is expected to level off in volume owing to the rise of both SAS (at the high end) and SATA (in nearline) beginning in 2007. One reason is that although only a few vendors are committed to producing Fibre Channel drives, almost every drive vendor is offering both SAS and SATA, making for increased competition. Also, SAS will offer the same performance characteristics as Fibre Channel, with the option of tunneling SATA protocols over the same physical and link layers.

SATA

One of the motivating factors for SATA was bandwidth. The maximum theoretical limit for parallel IDE interfaces was 133 MB/s. The 1.5, 3.0, and 6.0 Gb/s interfaces defined for SATA correspond to 150, 300, and 600 MB/s, offering a growth path that parallel interfaces could not match. SATA was also looked upon as an opportunity for nonenterprise drive vendors to gain a toehold in the enterprise space. A number of features were added to enable this:

- Native Command Queuing (NCQ) with scatter/gather features to improve random I/O performance
- 32-bit CRC checking for data and commands
- Hot-plug, blind-mate connectors for active sparing in RAID environments
- Point-to-point cabling versus "daisy-chaining," and SAS physical layer support

- The definition of port multipliers, allowing the connection of up to 15 disks to the same port
- Active–passive port selectors and active–active port multiplexors that provide dual-initiator options for higher availability

SAS

SAS and SATA are unique in that although SATA can be used to connect initiator ports directly to target ports in a point-to-point fashion for embedded desktop applications, the SAS protocol was defined to support both SAS and SATA drives over the same interconnect network. The same underlying physical and link-layer protocols support both interfaces, which presents a unique and compelling value proposition for many storage integrators. For the first time, both performance-oriented SAS and value-oriented SATA drives can be supported using the same cable plant.

Three transport protocols are supported over the SAS physical and link layers:

- Serial SCSI Protocol (SSP), which defines the mapping of SCSI commands over the link layer. Frame formats are based on Fibre Channel Protocol.
- Serial ATA Tunneling Protocol (STP), which defines connection delimiters, frames, and flow control unique to SATA devices.
- Serial Management Protocol (SMP), which adds management functions for the SAS expanders (circuit switches that distribute SAS traffic) using simple request-response functions related to discovery, status, and low-level hardware control.

SCSI Application	*SATA Application*	*Management Application*	*Architecture Defines*
SSP Transport	STP Transport	SMP Transport	Framing and information units
SSP Link	STP Link SAS Link	SMP Link	Encoding, primitives, flow control, connection management
	SAS Phy		Cables, connectors, electrical

FC VERSUS SAS DISKS IN THE ENTERPRISE

SAS is growing at the expense of SCSI, which was a premeditated outcome for early industry supporters of SAS. What perhaps was not expected was the rate at which SAS would gain in popularity at the expense of FC. Although this has not happened yet, both IDC and Seagate market research expect that within the next 12–18 months, storage suppliers will be shipping more SAS+SATA drives than FC+SCSI to enterprise customers, and within a year after that, two-thirds of enterprise drive shipments will be SAS+SATA. Considering how new these interfaces are, that adoption rate is unprecedented. Four reasons that may explain this trend are as follows:

1. There is a great deal of competition. Many of the silicon and HDD vendors that missed the FC bandwagon in the mid-1990s are attacking the SAS market with a vengeance to make sure they don't get left behind again in the lucrative enterprise market. The increased competition and combined marketing forces of these suppliers, along with price advantages, advanced feature sets, and greater motivation for interoperability compared to FC, are making SAS more attractive from a developer's perspective.
2. The new breed of high-density 1–2U and blade-based servers has increased demand for small-form-factor drives. The 2.5" drive interface of choice is SAS for this market, which has grown more quickly than many expected and is expected to accelerate the adoption of SAS in general.

3. With SATA support available using the same expander complex as SAS, and with SAS drives promising performance identical to that of FC, many developers are looking at SAS infrastructure as a means of getting two products for the development cost of one. FC–SATA bridging and tunneling solutions are either proprietary or late to the game, have fewer vendors supporting them, and have given SAS-SATA a lengthy head start.

4. SAS is leveraging many of the lessons learned from implementing high-speed serial interfaces. The SAS link and physical layers from FCP to 8b/10b encoding borrow from Fibre Channel. Also, the first SAS implementations are coming in the form of expanders for direct disk attachment, and cascaded expanders allow dozens of disk drives to be directly connected to host bus adapters without the need for external retiming hubs or switches. It wasn't until the later stages of adoption that commercially available loop switches provided options for native disk attachment, forcing early adopters to use external hubs and switches or to restrict themselves to modest configurations using primitive loop bypass circuits. Early switch interoperability issues combined with limited vendor selection also slowed adoption.

Fibre Channel still has its advantages. One is maturity: Fibre Channel is in its tenth year of multivendor implementation, whereas SAS is in its second, and there are bound to be early implementation glitches in any new technology. In addition, what started out as a relatively straightforward drive interface definition is sliding down the slippery slope of complexity that has somewhat plagued Fibre Channel as a disk interface. Zoning, security, and other "network" features threaten to delay standards and add complexity, and the SAS community must avoid the temptation to be all things to all developers. Fibre Channel is a better system network interface, provides distances up to multiple kilometers using fiber-optic options, and finally has multiple vendors providing interoperable switch solutions at both the high end and the low end. Attempts to compete with FC in this arena may slow the interoperability of storage subsystem components, cause a ripple effect back to the drive interface itself by adding complexity, and inadvertently slow the overall adoption of SAS if architecture, design, or interoperability problems result.

The bottom line is that 4 Gb and 8 Gb Fibre Channel will continue be the dominant storage system interconnect in the enterprise for the foreseeable future, but we'll see SAS begin to take significant Fibre Channel market share in 2007 as a disk interface, and before the end of the decade more SAS drives will be shipped than FC and SCSI put together.

SAS VERSUS SATA DISKS IN THE ENTERPRISE

Historically, the overriding priority for SATA drive design has been cost/GB, and this tradeoff shows up in the following areas when comparing SATA to SAS (or FC) drives in the enterprise [4]:

Attribute	*SAS/FC Feature Differentiators*
Mechanical	Larger magnets, stiffer covers, air control devices, faster seeks, low rotational vibration susceptibility
Head stack	More heads, low mass/high rigidity, higher-cost designs
Motor	Higher rpm, less runout, more expensive
Electronics	Dual processors, multi-host, dual-port, twice the firmware, high rpm control and rotational position sensing, superior error correction, smart servo algorithms, more sophisticated performance optimization and command scheduling, deeper queues, larger caches, and more sophisticated data integrity checks
Disks	More platters, smaller diameter, full media certification, and fully characterized
Format	Variable sector sizes (e.g., SATA is moving to large, fixed 4096-byte sectors)

Workloads that are optimal for nearline storage are sequential reads, compliance data, archived email, and other record archives with low duty cycles and low IOP requirements. Workloads optimal for performance storage are random reads and writes, high IOP rates, and high-duty-cycle traffic. Real-time OLTP workloads are an example.

The new features in SATA described previously will put pressure on the normally simple differentiation between the classic desktop and the classic enterprise drive. The cost advantage of SATA, particularly for nearline workloads, is compelling enough for drive integrators to be willing to spend a little more on data protection and enclosure features to accommodate these drives. While end users will want the best of all worlds, drive vendors will continue to prefer to withhold performance and reliability features from SATA drives to maintain their margins in their performance markets as well as to use the same drives to fight for market share on the desktop. This is why performance SAS and nearline SATA drives will continue to coexist in the enterprise for the foreseeable future.

However, systems are now being introduced that can accept both SATA and SAS drives coexisting in the same enclosure. This means that, for the first time, the choice of SATA versus SAS can become a post-purchase decision for customers. It is only fitting that, after 30 years of evolution, storage technology has finally allowed the consumer to more directly dictate the ultimate winner.

REFERENCES

[1] IDC Worldwide Disk Storage Systems Market Forecast and Analysis, 2002-9: http://www.itresearch.com/getdoc.jsp?containerId=33477.

[2] Enterprise Storage Forum, June 16, 2005, “Why Tape Won’t Die”: http://www.enterprisestorageforum.com/continuity/features/article.php/3513406.

[3] WinHEC 2005, “SATA in the Enterprise,” and Seagate Market Research: http://download.microsoft.com/download/9/8/f/98f3fe47-dfc3-4e74-92a3-088782200fe7/TWST05005_WinHEC05.ppt.

[4] Enterprise Storage Forum, Dec. 15, 2005, “Storage Headed for Trouble”: http://www.enterprisestorageforum.com/technology/features/article.php/3564426.

MARSHALL KIRK MCKUSICK

disks from the perspective of a file system

Dr. Marshall Kirk McKusick writes books and articles, consults, and teaches classes on UNIX- and BSD-related subjects. He has twice served on the Board and as president of USENIX.

kirk@usenix.org

MOST APPLICATIONS DO NOT DEAL with disks directly. Rather, they store their data in files in a file system. One of the key tasks of the file system is to ensure that the file system can always be recovered to a consistent state after an unplanned system crash (e.g., due to a power failure).

Although the file system must recover to a consistent state, that state usually reflects the state of the file system sometime before the crash (often data written in the minute before the crash may be lost). When an application needs to ensure that data can be reliably recovered after a crash, it does an fsync system call on the file or files that contain the data in need of long-term stability. Before returning from the fsync system call, the file system must ensure that all the data associated with the file can be recovered after a crash, even if the crash happens immediately after the return of the fsync system call.

The file system implements the fsync system call by finding all the dirty (unwritten) file data and writing these data to the disk. Historically, the file system would issue a write request to the disk for the dirty file data and then wait for the write-completion notification to arrive. This technique worked reliably until the advent of track caches in the disk controllers. Track-caching controllers have a large buffer in the controller that accumulates the data being written to the disk. To avoid losing nearly an entire revolution to pick up the start of the next block when writing sequential disk blocks, the controller issues a write-completion notification when the data are in the track cache rather than when they are on the disk. The early write-completion notification is done in the hope that the system will issue a write request for the next block on the disk in time for the controller to be able to write it immediately following the end of the previous block.

This approach has one seriously negative side effect. When the write-completion notification is delivered, the kernel expects the data to be on stable store. If the data are only in the track cache but not yet on the disk, the file system can fail to deliver the integrity promised to user applications using the fsync system call. In particular, semantics will be violated if the power fails after the write-completion notification but before the data are written to disk. Some vendors eliminate this problem by using nonvolatile memory for the track cache and providing microcode restart after a power failure to determine which operations need to be completed. Because this option is expensive, few controllers provide this functionality.

Newer disks resolve this problem with a technique called *tag queuing*. With tag queuing, each request passed to the disk driver is assigned a unique numeric tag. Most disk controllers supporting tag queuing will accept at least 16 pending I/O requests. After each request is finished, the tag of the completed request is returned as part of the write-completion notification. If several contiguous blocks are presented to the disk controller, it can begin work on the next one while notification for the tag of the previous one is being returned. Thus, tag queuing allows applications to be accurately notified when their data have reached stable store without incurring the penalty of lost disk revolutions when writing contiguous blocks.

Tag queuing was first implemented in SCSI disks, enabling them to have both reliability and speed. ATA disks, which lacked tag queuing, could either be run with their write cache enabled (the default), to provide speed at the cost of reliability after a crash, or with the write cache disabled, which provided reliability after a crash but at about a 50% reduction in write speed.

To try to solve this conundrum, the ATA specification added an attempt at tag queuing with the same name as that used by the SCSI specification, Tag Command Queueing (TCQ). Unfortunately, in a deviation from the SCSI specification, TCQ for ATA allowed the completion of a tagged request to depend on whether the write cache was enabled (issue write-completion notification when the cache is hit) or disabled (issue write-completion notification when media is hit). Thus, it added complexity with no benefit.

Luckily, with SATA there is a new definition called Native Command Queueing (NCQ) that has a bit in the write command that tells the drive if it shall report completion when media has been written or when cache has been hit. Provided that the driver correctly sets this bit, the disk will have the correct behavior.

In the real world, many of the drives targeted to the desktop market do not implement the NCQ specification. To ensure reliability, the system must either disable the write cache on the disk or issue a cache-flush request after every metadata update, log update (for journaling file systems), or fsync system call. Because both of these techniques lead to noticeable performance degradation, they are often disabled, putting file systems at risk in the event of power failures. Systems for which both speed and reliability are important should not use ATA disks. Rather, they should use drives that implement Fibre Channel, SCSI, or SATA with support for NCQ.

TOM HAYNES

introduction to ZFS

Tom Haynes is an NFS developer for Sun Microsystems, Inc., and is interested in the cost differential between open source and commercial offerings. He is exploring those costs by using OpenSolaris to design a NAS appliance.

tdh@excfb.com

THE ZETTABYTE FILE SYSTEM (ZFS) IS the replacement file system for UFS. In a nutshell, ZFS creates pools of data across multiple disks. It manages the complexity of formatting, partitioning, mirroring, and other tasks for the administrator.

You can search on Google and find many glowing testimonials about how ZFS was deployed, about how easy it was, about how great the software is, etc. But how much fun is it to just read about everything working out as expected? How often does that occur in your experience? Do we tune into "I Shouldn't Be Alive" on the Discovery Channel or "I Haven't Died Yet" on the Established Channel?

When I proposed this article, I wanted to write about a 1-terabyte NAS file server based on OpenSolaris. To minimize cost, all of the components were to be commodity parts and the drives would be SATA. What I'm going to write about is an exploration of ZFS on a 300 GB IDE drive. Oh, and I'm going to illustrate how the best-laid plans go astray. I'm also not going to define all of the ZFS or filesystem terminology—again, you can pick this stuff up online.

When you deploy either Solaris 10 or OpenSolaris on hardware not manufactured by Sun Microsystems, you need to do some research for compatibility. The best resource is the BigAdmin HCL, maintained at http://www.sun.com/bigadmin/hcl/. This Hardware Compatibility List details experiences with various x86 systems and components with the different flavors of Solaris. I picked the MSI K8N Master2-FAR motherboard because of the support for the NVIDIA nForce4 chip set, the support for the two GigE Ethernet controllers, and the ability to support four SATA drives without an additional controller card. Note that this MSI MB utilizes an NVIDIA nForce4 Host Bus Adapter. The Sun Ultra 20 also utilizes NVIDIA nForce chip sets to handle the HBA duties.

Right after I ordered this MB, bug 6363449 was filed on the Ultra 20. Basically, the NVIDIA nForce4 gets confused with the ZFS label written to the SATA drives. There are some measures mentioned in the bug report to get the drives working, but they do not work on my MB.

I had finally constructed my system, loaded Nevada b27 on it, done some fun things with ZFS, and powered the machine off for the night. That's when I found out about the bug. See, the fan was very loud. The system would not reboot the next morning. Considering the bad luck I had with the system, I named it wont, as in “wont work.” I was able to identify the bug with help on the OpenSolaris discussion forums. I tried booting with the drive entries set to “none” in the BIOS, but still no joy. I disconnected the SATA drives and the system booted fine. By the way, the SATA connectors are very fragile; I broke one, and I would advise you not to reinsert the cables too often.

A limiting factor in getting parts working in a home office is that you might just have one of everything. I don’t have another system in which I can put a different VTOC on the drives. (Just like I only had one power supply, one MB, one case, and one video card when I was troubleshooting the original reason the system would not boot: The video card was not compatible with the MB.)

I actually learned a lot about OpenSolaris during this very frustrating process. Among other things, I figured out how to use kmdb (kernel debugger), how to boot the system into the console from grub, how to wire the console, and how to force a core.

So I’ve hit the cutting edge of OpenSolaris and it appears I have two choices:

1. Convince the Solaris SATA developers that the bug needs to be fixed ASAP.
2. Hunker down and fix the issue myself.

The only reason there was any urgency on this bug was the deadline for this article. And I’ve been too busy with my new job to tackle the code myself.

But is there a third choice, besides RMAing the whole mess and trying my luck again?

Yes, there is actually a cheap alternative—just add another IDE drive. ZFS is quite capable of working with slices and not just disks. Sure, you introduce a single point of failure and bypass many of the benefits of having mirrored storage. But the goal is to play with ZFS, and to do so cheaply. I must admit I struggled with this decision; I’m used to NAS boxes that have a single storage partition spread over multiple disks, not a NAS box that has multiple storage partitions spread across a single disk.

I took a 300 GB IDE drive and created four equal slices of 68 GB. You can do this with the following format:

Note that, under OpenSolaris, disks are assigned names of the form controller

```
# format
Searching for disks...done

AVAILABLE DISK SELECTIONS:
        0. c0d0 <DEFAULT cyl 4862 alt 2 hd 255 sec 63>
                /pci@0,0/pci-ide@6/ide@0/cmdk@0,0
        1. c0d1 <DEFAULT cyl 36477 alt 2 hd 255 sec 63>
                /pci@0,0/pci-ide@6/ide@0/cmdk@1,0
Specify disk (enter its number): 1
format> partition
partition> p
Current partition table (original):
Total disk cylinders available: 36477 + 2 (reserved cylinders)

Part  Tag        Flag    Cylinders        Size                 Blocks
0     stand      wm      3 -  8879        68.00 GB        (8877/0/0)      142609005
1     stand      wm      8880 - 17756     68.00 GB        (8877/0/0)      142609005
```

```
2   backup      wm   0 - 36476       279.43 GB   (36477/0/0)   586003005
3   stand       wm   17757 - 26633   68.00 GB    (8877/0/0)    142609005
4   stand       wm   26634 - 35510   68.00 GB    (8877/0/0)    142609005
5   stand       wm   35510 - 36476   7.41 GB     (967/0/0)     15534855
6   unassigned  wm   0               0           (0/0/0)       0
7   unassigned  wm   0                           0             (0/0/0)       0
8   boot        wu   0 -             0           7.84 MB       (1/0/0)       16065
9   alternates  wu   1 -             2           15.69 MB      (2/0/0)       32130
```

ID and disk ID. So "c0d1" is the slave on the first controller. We can further reference the different slices on the disk. For now, think of slices as partitions. It isn't entirely accurate, but it is the concept we want to work with here.

The first thing we can try is to create a ZFS pool; if we were using entire disks, we could think of the pool as a volume of disks. If we were to add RAID to the mix, you would then be able to remove a disk from the volume, if it failed, and replace it with a spare. The file system would then rebuild the missing data on that new disk.

For right now, we want to construct a pool of storage that is larger than any single available unit. Perhaps we need some scratch space for a computational job.

```
# zpool create zoo c0d1s0 c0d1s1
# zpool list
NAME   SIZE   USED    AVAIL   CAP   HEALTH    ALTROOT
Zoo    135G   57.5K   135G    0%    ONLINE    -
```

So the system has a 135GB pool to use for storage. What this means is that the data can span the two slices. With this configuration, there is no redundancy.

We could instead have created a mirrored pool—one that halves your available storage but keeps an exact copy of the contents. Under this model, if one side becomes corrupt, you can break the mirror and activate the other side. With normal RAID configurations, you can survive a single disk failure. Mirroring allows you to survive multiple disk failures on one of the sides.

```
# zpool destroy zoo
# zpool create zoo mirror c0d1s0 c0d1s1
# zpool list
NAME   SIZE    USED    AVAIL   CAP   HEALTH    ALTROOT
zoo    67.5G   57.5K   67.5G   0%    ONLINE    -
```

Note that the mirror does indeed halve the storage. Also, we lost some space for ZFS overhead. Perhaps we want to add some additional storage:

```
# zpool add zoo c0d1s2 c0d1s3
invalid vdev specification
use '-f' to override the following errors:
/dev/dsk/c0d1s2 overlaps with /dev/dsk/c0d1s0
# zpool add zoo mirror c0d1s3 c0d1s4
invalid vdev specification
use '-f' to override the following errors:
/dev/dsk/c0d1s4 contains a ufs filesystem.
/dev/dsk/c0d1s4 overlaps with /dev/dsk/c0d1s5
```

The zpool command is keeping me from shooting myself in the foot. Slice 2 should never be used, and slice 4 earlier had a UFS file system. That should be easy to fix, but I'm more concerned with the data that exist on slice 5. Notice that it was just when I moved to ZFS that I found out about the overlap. I'm in the process of exploring how OpenSolaris DVDs are made bootable, and /dev/dsk/c0d1s5 contains the contents of the x86 DVD—see http://www.kanigix.org for more details on this project. So if I lose the data, I have it on DVD.

Now let's make the new file system real and use it to save the data. We have a ZFS pool, but now we need to create a file system on that pool and allow it to be utilized. A good question is, Why take the extra step? Why not make the pool the base unit? The reason is that we want to be able to store multiple file systems in a pool. What if we want to clone a file system? What if we want to take a snapshot of a file system? Taking this design path from the start saves the pain of trying to retrofit this functionality later—say, when many customers have vital data to be protected during an upgrade.

```
# zfs create zoo/x86
# df -h | grep zoo
zoo       67G    99K    67G    1%     /zoo
zoo/x86   67G    98K    67G    1%     /zoo/x86
# ls -la /zoo
total 6
drwxr-xr-x        3 root          sys     3 Mar 19 23:16 .
drwxr-xr-x        42 root         root    1024 Mar 19 23:08 ..
dr-xr-xr-x        3 root          root    3 Mar 19 23:17 .zfs
drwxr-xr-x        2 root          sys     2 Mar 19 23:16 x86
```

ZFS created the file system and mounted it for me. One of the ease-of-use factors of ZFS is that it automates many of the manual steps used with creating other file systems and making them ready for use.

I can use cpio to safely copy the data over to the new file system:

```
# chown tdh:staff /zoo/x86
# cd /kanigix/
# find . -depth -print | cpio -pudm /zoo/x86
6608816 blocks
# df -h /kanigix /zoo /zoo/x86
Filesystem        size    used    avail   capacity Mounted on
/dev/dsk/c0d1s5   7.3G    3.1G    4.1G    44%      /kanigix
zoo               67G     99K     64G     1%       /zoo
zoo/x86           67G     3.2G    64G     5%       /zoo/x86
```

Notice that although /zoo and /zoo/x86 appear to be different file systems, they share the same storage.

We copied the data over because we need to recreate the slice on which it resided—slices 4 and 5 shared a block. We now need to fix the two slices (remembering to comment out the entry in /etc/vfstab). After using format (and the subcommand of partition), these slices now look like this:

```
4  stand   wm   26634 - 35510   68.00GB   (8877/0/0)   142609005
5  stand   wm   35511 - 36476   7.40GB    (966/0/0)    15518790
```

Although I modified slice 5, I did not do so for slice 4. zpool will still think there is a valid UFS file system on that slice, so we need to force it to use that slice:

```
# zpool add -f zoo mirror c0d1s3 c0d1s4
# zpool list
NAME   SIZE    USED    AVAIL   CAP    HEALTH          ALTROOT
zoo    135G    3.21G   132G    2%     ONLINE          -
# df -h /zoo /zoo/x86
```

```
Filesystem        size    used    avail   capacity        Mounted on
zoo               134G    99K     131G    1%              /zoo
zoo/x86           134G    3.2G    131G    3%              /zoo/x86
```

We are now using about 268GB of raw disk space to provide a mirrored pool. Again, by using a single disk, the mirroring will only provide minimal benefit. Conceivably, someone could corrupt the slices with the format command—but we don't expect that. But as a cheap tour of the ZFS feature set, this setup works.

A common ZFS task is to create NFS exported home directories with a quota. We use inheritance to say that any file systems created inside /export/zfs are to exported via NFS, are to be compressed, and will have a 10GB quota. Note that we are creating file systems within other file systems. We are setting defaults, which can be overridden at any time.

```
# zfs create zoo/home
# zfs set mountpoint=/export/zfs zoo/home
# zfs set sharenfs=on zoo/home
# zfs set compression=on zoo/home
# zfs set quota=10G zoo/home
# zfs create zoo/home/nfsv2
# zfs create zoo/home/nfsv3
# zfs create zoo/home/nfsv4
# zfs list
NAME              USED    AVAIL   REFER   MOUNTPOINT
zoo               3.21G   131G    99.5K   /zoo
zoo/home          395K    10.0G   99.5K   /export/zfs
zoo/home/nfsv2    98.5K   10.0G   98.5K   /export/zfs/nfsv2
zoo/home/nfsv3    98.5K   10.0G   98.5K   /export/zfs/nfsv3
zoo/home/nfsv4    98.5K   10.0G   98.5K   /export/zfs/nfsv4
zoo/x86           3.21G   131G    3.21G   /zoo/x86
```

One thing to note here is that zoo/x86 is only available as /zoo/x86. Since it is not under zoo/home, the defaults we provided do not apply. Also note that it is not exported. Finally, if we do go to /zoo, we will not see "home."

And we check that the home directories are exported on the box wont:

```
[tdh@adept ~]> showmount -e wont
Export list for wont:
/export/zfs          (everyone)
/export/zfs/nfsv2    (everyone)
/export/zfs/nfsv3    (everyone)
/export/zfs/nfsv4    (everyone)
```

By the way, I never enabled the NFS server on wont. I know how to do it, but I did not have to do anything, since ZFS did it for me. Note that I am responsible for creating user accounts and changing ownership of the root of the file systems.

A cautionary note here is that the quotas are on the file system and not per user. ZFS does a lot for you behind the scenes, but it doesn't know that these are user accounts we are creating. So if the user nfsv2 were to copy files under the /export/zfs/nfsv3 file system, the charge would be against the file system and not against either of the two user accounts.

```
# useradd -m -u 1094 -g 100 -c "Mr. NFSv2" -d /export/zfs/nfsv2 nfsv2
# chown nfsv2:100 /export/zfs/nfsv2
# ls -al /export/zfs
total 10
```

```
drwxr-xr-x    5 root    sys      5 Mar 20 00:33 .
drwxr-xr-x    4 root    sys      512 Mar 20 00:31 ..
dr-xr-xr-x    3 root    root     3 Mar 20 00:40 .zfs
drwxr-xr-x    2 nfsv2   protos   2 Mar 20 00:33 nfsv2
drwxr-xr-x    2 nfsv3   protos   2 Mar 20 00:33 nfsv3
drwxr-xr-x    2 nfsv4   protos   2 Mar 20 00:33 nfsv4
```

We can test snapshots to see whether we can safeguard our data, in this case a copy of this article. When you take a snapshot of a file system, you are basically telling the OS that if the contents are changed, keep a copy of the old contents. This copy stays until the snapshot is deleted.

There are different ways to achieve this, but a common approach employs copy-on-write. Initially the two file systems (the original and the copy) point to the same inodes and blocks. The savings here is that the snapshot consumes minimal storage. We can see that here when we create the snapshot:

```
# zfs snapshot zoo/home/nfsv4@monday
# zfs list
NAME                     USED    AVAIL   REFER   MOUNTPOINT
zoo                      3.21G   131G    99.5K   /zoo
zoo/home                 404K    10.0G   100K    /export/zfs
zoo/home/nfsv2           98.5K   10.0G   98.5K   /export/zfs/nfsv2
zoo/home/nfsv3           98.5K   10.0G   98.5K   /export/zfs/nfsv3
zoo/home/nfsv4           108K    10.0G   108K    /export/zfs/nfsv4
zoo/home/nfsv4@monday    0       -       107K    -
zoo/x86                  3.21G   131G    3.21G   /zoo/x86
```

The accounting shows that only zoo/home/nfsv4 has any storage. When the contents are changed, the original blocks are weaved into the snapshot space and the new ones are created inside the live file system. We can see that when we delete the file:

```
> ls -la
total 23
drwxr-xr-x   2 nfsv4   protos   4 Mar 20 01:38       .
drwxr-xr-x   5 root    sys      5 Mar 20 00:33       ..
dr-xr-xr-x   3 root    root     3 Mar 20 01:43       .zfs
-rw-r--r--   1 nfsv4   protos   0 Mar 20 01:04       it
-rw-r--r--   1 nfsv4   protos   11808 Mar 20 01:38 zfs.txt
> rm zfs.txt
> ls -la
total 5
drwxr-xr-x   2 nfsv4   protos   3 Mar 20 01:43       .
drwxr-xr-x   5 root    sys      5 Mar 20 00:33       ..
dr-xr-xr-x   3 root    root     3 Mar 20 01:43       .zfs
-rw-r--r--   1 nfsv4   protos   0 Mar 20 01:04       it
> zfs list | grep nfsv4
zoo/home/nfsv4               206K 10.0G   98.5K   /export/zfs/nfsv4
zoo/home/nfsv4@monday        107K -       108K     -
```

The storage has now transferred over to the snapshot. Also, the snapshot storage is coming from the containing file system. Note how the other numbers (USED and REFER) increased.

We can recover either the entire snapshot or just the file. To get the file back:

```
> ls -la .zfs/snapshot/monday/
total 21
```

```
drwxr-xr-x     2 nfsv4  protos   4 Mar 20 01:38 .
dr-xr-xr-x     3 root   root     3 Mar 20 01:43 ..
-rw-r--r--     1 nfsv4  protos   0 Mar 20 01:04 it
-rw-r--r--     1 nfsv4  protos   11808 Mar 20 01:38 zfs.txt
> cp .zfs/snapshot/monday/zfs.txt .
> ls -la
total 6
drwxr-xr-x     2 nfsv4  protos   4 Mar 20 01:44 .
drwxr-xr-x     5 root   sys      5 Mar 20 00:33 ..
dr-xr-xr-x     3 root   root     3 Mar 20 01:44 .zfs
-rw-r--r--     1 nfsv4  protos   0 Mar 20 01:04 it
-rw-r--r--     1 nfsv4  protos   11808 Mar 20 01:44 zfs.txt
```

At first the snapshot consumed no space, but as we caused it to deviate from the original, it was forced to keep the content.

```
> zfs list | grep monday
zoo/home/nfsv4@monday  106K    -       107K    -
```

As we change the copy in the live file system, we can see that the two files differ:

```
> diff zfs.txt .zfs/snapshot/monday/zfs.txt  | wc -l
      55
```

As alluded to earlier, we could restore the entire snapshot. Perhaps an errant script did an rm -rf or a virus corrupted everything. With our example:

```
# zfs rollback zoo/home/nfsv4@monday
# zfs list | grep nfsv4
zoo/home/nfsv4         108K    10.0G   108K    /export/zfs/nfsv4
zoo/home/nfsv4@monday  0       -       108K    -
```

I have tried to provide a taste of what ZFS can do for you and how you do not need to spend a lot of money on disks to take it for a spin. I did not explore all of the features—for example, creating a RAID pool, backing up a snapshot to tape, or cloning a file system. I showed perhaps the most common example, creating user accounts, and while I could have picked something different, for example, staging areas for external Web servers, I picked it for a reason.

When I taught undergraduate CS courses, I would have loved the ability to couple ZFS with Zones. Imagine that each student or group has its own virtual server and its own file system. One student cannot inadvertently rob the rest of the use of the machine and students cannot go look at each other's source code. They cannot complain that they accidentally deleted their files (i.e., a snapshot will keep that dog away from their homework). Also, if the due time is 5 p.m., just take a snapshot of the file systems. There is no need to worry about some industrious student changing the timestamps on the files.

Attention, Members:
Are You Getting the Most Out of Your Membership?

Become an active member of the Association. This is your community: get involved!

We are proud of our 30-year history of offering services to the advanced computing systems community. The support and participation of our members make us able to offer some of the most highly respected conferences and publications in the industry.

We have recently added the benefit of a Jobs Board for all USENIX and SAGE members, as well as additional benefits for our SAGE, Educational, Corporate, and Supporting members. We encourage you either to upgrade your membership or to talk to your employer about an institutional membership with USENIX.

In addition to the great benefits you already enjoy, we are offering these new benefits:

STANDARD USENIX MEMBERSHIP: INDIVIDUAL ($115 PER YEAR) AND STUDENT ($40 PER YEAR)

- The USENIX Jobs Board: Looking for a new job? USENIX members have direct access to offerings from top-notch potential employers. Members can also post resumes. For information on how to post, see http://www.usenix.org/jobs/.

SAGE MEMBERSHIP: INDIVIDUAL ($40 PER YEAR) AND STUDENT ($25 PER YEAR)

- Resume posting service
- The latest Short Topics in System Administration booklet for every member

USENIX EDUCATIONAL MEMBERSHIP ($250 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to two additional copies of *;login:* per issue (email office@usenix.org with your request)

USENIX CORPORATE MEMBERSHIP ($460 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to four additional copies of *;login:* per issue (email office@usenix.org with your request)
- Up to five conference registrations at the USENIX member price for your staff (email conference@usenix.org for a discount code to use in registering)
- Your company name listed on our Corporate Members Web page, http://www.usenix.org/membership/corporate.html.

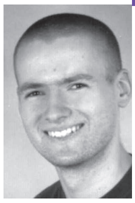
USENIX SUPPORTING MEMBERSHIP ($2500 PER YEAR)

- The USENIX Jobs Board (see above)
- Up to four additional copies of *;login:* per issue (email office@usenix.org with your request)
- Tarballs of any USENIX conference Proceedings from the year *before* your membership term begins (email office@usenix.org with your request)

For a full listing of all benefits or to join online, please see http://www.usenix.org/membership.

STEFAN BÜTTCHER AND
CHARLES L.A. CLARKE

adding full-text filesystem search to Linux



Stefan Büttcher received a Master's degree in computer science from the University of Erlangen, Germany. Since 2004, he has been a Ph.D. student at the University of Waterloo, Canada. His research interests include all aspects of high-performance search engines, especially index maintenance strategies for dynamic text collection. Stefan is the main developer of the Wumpus search engine.

sbuettch@plg.uwaterloo.ca

Charles Clarke is an Associate Professor in the School of Computer Science at the University of Waterloo. His research interests include information storage and retrieval, software development tools, and programming language implementation. Clarke received his Ph.D. from the University of Waterloo in 1996. From 1996 to 1999 he was an assistant professor in the Department of Electrical and Computer Engineering at the University of Toronto. He has also held software development positions at a number of computer consulting and engineering firms.

claclark@plg.uwaterloo.ca

IN THE PAST TWO YEARS, FULL-TEXT desktop search systems have experienced an amazing updraft. For Windows, there now are about a dozen independent desktop search engines from which the user can choose. For Linux, the situation is different; only a few desktop search systems exist.

In this article we report on experiences we had while developing Wumpus, a full-text filesystem search engine for Linux. We discuss major design decisions and point out some changes that, from a search engine developer's point of view, need to be made to the Linux kernel to support real-time filesystem indexing and search.

The goal of our research efforts is the development of a unified filesystem search engine that can be used by multiple users and that can cover multiple storage devices, both local and network-wide (local hard drives, USB sticks, NFS mounts, etc.). Search results returned by the engine should always be consistent with the current content of the file system. Inconsistencies resulting from recent file changes should have a lifetime of at most a few seconds.

The vehicle we are using to reach that goal is the Wumpus search engine, a hybrid filesystem search and general-purpose information retrieval system. Wumpus is free software, licensed under the terms of the GNU General Public License, and is available for download from the Wumpus Web site, http://www.wumpus-search.org/. It is work in progress and not yet suitable for everyday use as a filesystem search engine.

Wumpus is a keyword-based search engine. It supports state-of-the-art result ranking algorithms, as well as structural queries (phrase queries and near operators) and Boolean operators. Its backend index data structure is a set of inverted files. Each inverted file realizes a mapping from terms to their respective occurrences within the file system. (For a thorough discussion of inverted files and their advantages over alternative index data structures, see Zobel et al. [4]). In conjunction, the inverted files can be used to efficiently obtain a list of all occurrences of a given term within the file system. The result of a search query (e.g., "find the set of all files containing the given query terms") can then be produced by combining the lists of all query terms in a straightforward way.

When new files are created or existing files are modified, index information for the new data is added to in-memory index buffers. Whenever the amount of these in-memory data exceeds a certain threshold, they are written to disk, resulting in a new on-disk inverted file. Several inverted files may exist in parallel and are merged in a hierarchical fashion when it is appropriate to do so. This can be done very efficiently. A detailed description of index maintenance strategies for dynamic text collections can be found in the literature [2, 3].

When we started to develop our search engine, we had to make several major design decisions. Among the most important were index locality decisions. In a typical Linux installation, the file system will contain files belonging to more than a single user. It will also span across multiple mount points, representing different storage devices. These two aspects of filesystem search define two independent locality axes (the user axis and the device axis, as shown in Table 1). We had to decide whether index information should be stored locally or globally along each axis. Other locality axes, such as the time axis, exist and also play a role in filesystem indexing, but the user and the device axEs are the most important.

		Device Axis	
		Local	*Global*
User Axis	*Local*	A separate index for each user on each device	Per-user indices, each covering all devices
	Global	Device-specific indices, each containing data for all users	A single index covering all users and all devices

TABLE 1: THE TWO MAIN LOCALITY AXES IN MULTIUSER, MULTIDEVICE FILESYSTEM SEARCH

User Axis: A Single, Global Index to Be Accessed by All Users

Most existing desktop search tools maintain per-user indices. Although this is acceptable in single-user search environments, in pure desktop search environments (i.e., without the option to search the entire file system), and in environments with a small number of users and very little interaction among them (as is the case in a typical Windows system), it is not a good idea in a true multiuser filesystem search environment. Maintaining per-user indices, where each index only contains information about files that may be searched by the respective user, leads to two types of problems:

- Redundancy: Many files (such as man pages and other documentation files) can be accessed by all users in the system. All these files have to be independently indexed for each user in the system, leading to a massive storage overhead in systems with more than a handful of users.
- Performance: If per-user indices are used, then even a single `chmod` or `chown` operation can trigger a large number of disk operations, because the respective file needs to be completely reindexed (or data need to be copied from one user's index to another user's index) each time a user executes `chown`. Even in a system with only two users, this can be exploited to realize a denial-of-service attack on the indexing service.

The only solution to these problems is to use a single index that is shared by all users in the system, instead of many per-user indices. This index is maintained by a process with superuser rights that can add new information to the index when new files are created and remove data from the index when files are deleted. `chmod` and `chown` operations can then be dealt with by simply updating index metadata, without the need to reindex the file content.

Of course, to guarantee data privacy, the global index, because it contains information about all indexable files in the system, may never be accessed directly by a user.

Instead, whenever a user submits a search query, it is sent to the indexing service (running with superuser rights). The indexing service then processes the query, fetching all necessary data from the index, and returns the search results to the user, applying all security restrictions that are necessary to make the search results consistent with the user's view of the file system, while not revealing any information about files that may not be accessed by the user who submitted the query. The problem of applying user-specific security restrictions to the search results is nontrivial, but it can be solved (see [1] for details).

Device Axis: Local, Per-Device Indices

When experimenting with various desktop search systems for Windows, we noticed that most of them had problems with removable media. They either refused to index data on removable media altogether, or they added information about files on removable media to the index, but removing the medium from the system later on was not reflected by the index, and search results still referred to files on a USB stick, for example, even after the stick had been unplugged.

If index data are stored in a global, system-wide index, it is not clear how to deal with removable media. Should the index data be removed from the index immediately after the medium is removed from the system? If not, how long should the indexing service wait until it removes the data? Should external hard drives be treated as removable media?

The only solution to these problems is to maintain per-device indices. In Linux, for instance, this means that each device (/dev/hda, /dev/hdb, etc.) will get its own local index that only contains information about files on that particular device. Whenever a device is removed from the file system, the indexing process associated with that device is terminated. Whenever a device is added to the file system, a new indexing process is started for the new device (or not, depending on parameter settings). Search queries are processed by combining the information found in the individual per-device indices and returning the search results, which may refer to several different devices, to the user.

For network file systems such as NFS mounts, this means that the index is not kept on the client side, but on the server that contains physical storage device. This requires additional communication between the NFS server and the client during the processing of a search query and is a potential bottleneck in situations where an NFS server is accessed by a large number of clients and where many users want to search for data on the server. Nonetheless, this is the only way to allow the index to be updated in real time, as it is impossible for an NFS client to be informed of all changes that take place in a remote file system.

Maintaining per-device indices also makes it possible to remove a storage device from one computer system and attach it to another one without needing to reindex the files stored on the device. Since the index is kept on the device itself, all index information will immediately be available on the new system. As far as we know, the same approach is followed by Apple's Spotlight.

Filesystem Event Notification

To be able to fully implement this type of filesystem search framework, a comprehensive filesystem event notification interface is needed so that the operating system kernel can inform the indexing service about changes in the file system, that is, changes to the content of a file or changes to its metadata, such as file name and access privileges. Many operating systems provide system calls that allow a process to register for changes in a certain part of the file system (usually a directory, or a subtree rooted at a given directory) and to receive notifications about all filesystem events affecting that part of the file system.

In Windows, for example, an application can use the FindFirstChangeNotification system call (and related functions) to register for a variety of filesystem events in a given directory. The system call can also be used to register for changes in arbitrary subdirectories of the given directory. The latter is called a *recursive watch* and is very useful if one wants a process to monitor the entire file system.

THE TRADITIONAL LINUX NOTIFICATION SYSTEM: dnotify

In Linux, filesystem event notification had traditionally been realized through the dnotify interface. In dnotify, a process can register for changes to the contents of a particular directory by obtaining a handle to that directory and performing an fcntl system call for the handle. Events will be sent to the process in the form of UNIX signals. As soon as the process releases a handle, it will no longer be notified of changes in the directory associated with it.

This approach has two major problems. First, the interface requires an application to keep an open handle to each directory that is being watched for changes. For very large file systems, with hundreds of thousands of directories, this is not feasible. Second, it is not possible to register for recursive watches that include all subdirectories of the given directory. Again, for large file systems this is problematic. After a system reboot, for example, the entire file system needs to be scanned to obtain a handle to every directory. Depending on the size of the file system, this can take from several minutes to several hours.

THE NEW LINUX NOTIFICATION SYSTEM: inotify

Since version 2.6.13 (August 2005), the Linux kernel supports a second event notification interface, inotify. inotify is, for instance, used by the Beagle (http://www.gnome.org/projects/beagle/) search system.

The new interface removes dnotify's main shortcoming, the necessity of having an open handle to every directory in the file system. With inotify, an application obtains a handle to an inotify queue object and subsequently registers for event notification for all directories in which it is interested. The queue handle can be treated like an ordinary file handle, allowing synchronous and asynchronous I/O.

dnotify's second main shortcoming, the necessity of scanning the entire file system after a system reboot, is shared by inotify. Recursive watches are not supported. With inotify, a process has to register for each directory separately. The rationale behind this is that it allows file permission to be checked during the registration process; the request can then simply be rejected if the process does not have sufficient access privileges. If recursive watches were supported, this check would need to be performed at notification time, potentially adding significant overhead to the notification system. Unfortunately, inotify's security model does not take into account the possibility of access privileges being changed after a user obtains a watch for a directory. If the user does have read permissions for a directory and is granted the right to watch the directory, but loses read permission for the directory later on, inotify will still notify the user about changes in the directory.

The nonexistence of recursive watches in inotify introduces potential race conditions, for example when files and directory hierarchies are extracted from an archive and files are moved to other directories before the indexing service can register for changes in the new directories. This adds additional complexity to the indexing system and could have been avoided if recursive watches were supported.

SHORTCOMINGS OF THE EXISTING LINUX NOTIFICATION SERVICES

In addition to the absence of recursive watches, the existing filesystem notification facilities of the Linux kernel lack a few other features that are desirable for full-text filesystem search and imperative for the framework we propose:

- Fine-grained file change notification: When the content of a file is changed, inotify (and dnotify) rather laconically reports "file changed" but does not elaborate on which exact part of the file is affected by the change. Suppose a user has a large mailbox file, containing thousands of messages, and a single message is appended to the existing file. With inotify, the indexing service will have to guess that the change was only an append operation, but it can never be sure without rereading the entire file, which might take a long time, depending on the size of the mailbox file. A more detailed notification message, including the start and the end offset of the part of the file affected by the change, is desirable. This feature is trivial to implement but will probably require a change of the current inotify interface to userspace processes.
- Unmount request notification: Maintaining per-device indices requires the indexing system to have open files on each mounted device. This is imperative, as all index maintenance strategies for dynamic search systems rely on the ability to buffer updates in memory and only perform physical index updates from time to time. As a consequence, devices cannot be unmounted any more ("umount: device is busy"). To be able to unmount a device, the indexing process for that device needs to be terminated first. However, during the short period of time between shutting down the indexing process and unmounting the device, files can be changed. Those changes will never be detected by the indexing system unless it performs an exhaustive scan every time a device is added to the file system. To solve this problem, the operating system needs to provide atomic unmount operations that can include actions of userspace processes. Although this would probably require major changes to the Linux kernel, it seems to be the only clean solution to the unmount dilemma.

AN EXPERIMENTAL SOLUTION: fschange

The problems discussed here are addressed by the experimental fschange notification system. fschange is a patch for the Linux kernel and is available online (http://stefan.buettcher.org/cs/fschange/). After the kernel is updated, it can be accessed by a userspace process through the proc file system: /proc/fschange. In contrast to the existing notification interfaces part of the Linux kernel, fschange does not require a process to register for each directory individually. It provides a global view of the file system. By reading from /proc/fschange, the process obtains information about all changes taking place in the entire file system. Consequently, a process needs to have superuser privileges to be allowed to read from the file.

Because fschange provides a global view, exhaustive disk scans after a reboot or after mounting a new device are no longer necessary. Race conditions stemming from the necessity to register for each directory individually are eliminated, too. In addition to filesystem indexing, the interface can also be used by other types of applications (e.g., backup and file replication systems).

fschange supports most of the message types provided by inotify, plus a few others, such as mount notifications (needed to create a new indexing process when a storage device is added to the system). When a file is changed through a write or an mmap operation, fschange tells the user-space process not only the name of the file that was changed but also the start and end offset of the part of the file affected by the change.

The unmount problem discussed in the foregoing is addressed by providing two unmount events: UNMOUNT_REQ, indicating that a process requested unmounting an active storage device and that the request was rejected owing to open files for the device; and UNMOUNT, indicating that a storage device was successfully unmounted. When the indexing service receives a UNMOUNT_REQ notification, it terminates the process for the storage device affected by the unmount, closing open files for that device. In our prototype system, the umount system tool was modified in such a way that it sends a sequence of unmount requests to the kernel until the kernel reports a successful execution of the unmount operation or until a time-out (usually a few seconds) is reached. This strategy does not really solve the unmount dilemma, but at least it allows one to unmount file systems without losing excessive amounts of index data, which would otherwise be impossible.

Conclusion

We believe that a true filesystem search engine for Linux, providing each user with a global view of the searchable file system, is badly needed. We have outlined some important properties of such a search engine and discussed why it is difficult to implement a search engine with these properties, given the current support for filesystem notification provided by the Linux kernel. We hope that some of the functionalities we suggest will be added to the existing kernel services in the future, opening the way for real-time filesystem search in Linux.

REFERENCES

[1] S. Büttcher and C.L.A. Clarke, “A Security Model for Full-Text File System Search in Multi-User Environments,” *Proceedings of the 4th USENIX Conference on File and Storage Technologies (FAST 2005),* San Francisco, U.S.A., December 2005.

[2] S. Büttcher and C.L.A. Clarke, “Indexing Time vs. Query Time Trade-offs in Dynamic Information Retrieval Systems,” *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005)*, Bremen, Germany, November 2005.

[3] N. Lester, A. Moffat, and J. Zobel, “Fast On-Line Index Construction by Geometric Partitioning,” *Proceedings of the 14th ACM Conference on Information and Knowledge Management (CIKM 2005),* Bremen, Germany, November 2005.

[4] J. Zobel, A. Moffat, and K. Ramamohanarao, “Inverted Files versus Signature Files for Text Indexing,” *ACM Transactions on Database Systems,* 23(4):453–490, 1998.

PABLO NEIRA AYUSO

Netfilter's connection tracking system

Pablo Neira Ayuso has an M.S. in computer science and has worked for several companies in the IT security industry, with a focus on open source solutions. Nowadays he is a full-time teacher and researcher at the University of Seville.

pneira@lsi.us.es

FILTERING POLICIES BASED UNIQUELY on packet header information are obsolete. These days, stateful firewalls provide advanced mechanisms to let sysadmins and security experts define more intelligent policies. This article describes the implementation details of the connection tracking system provided by the Netfilter project and also presents the required background to understand it, such as an understanding of the Netfilter framework. This article will be the perfect complement to understanding the subsystem that enables the stateful firewall available in any recent Linux kernel.

The Netfilter Framework

The Netfilter project was founded by Paul "Rusty" Russell during the 2.3.x development series. At that time the existing firewalling tool for Linux had serious drawbacks that required a full rewrite. Rusty decided to start from scratch and create the Netfilter framework, which comprises a set of hooks over the Linux network protocol stack. With the hooks, you can register kernel modules that do some kind of network packet handling at different stages.

Iptables, the popular firewalling tool for Linux, is commonly confused with the Netfilter framework itself. This is because iptables chains and hooks have the same names. But iptables is just a brick on top of the Netfilter framework.

Fortunately, Rusty spent considerable time writing documentation [1] that comes in handy for anyone willing to understand the framework, although at some point you will surely feel the need to get your hands dirty and look at the code to go further.

THE HOOKS AND THE CALLBACK FUNCTIONS

Netfilter inserts five hooks (Fig. 1) into the Linux networking stack to perform packet handling at different stages; these are the following:

- PREROUTING: All the packets, with no exceptions, hit this hook, which is reached before the routing decision and after all the IP header sanity checks are fulfilled. Port Address Translation (NAPT) and Redirections, that is, Destination Network Translation (DNAT), are implemented in this hook.

- LOCAL INPUT: All the packets going to the local machine reach this hook. This is the last hook in the incoming path for the local machine traffic.
- FORWARD: Packets not going to the local machine (e.g., packets going through the firewall) reach this hook.
- LOCAL OUTPUT: This is the first hook in the outgoing packet path. Packets leaving the local machine always hit this hook.
- POSTROUTING: This hook is implemented after the routing decision. Source Network Address Translation (SNAT) is registered to this hook. All the packets that leave the local machine reach this hook.

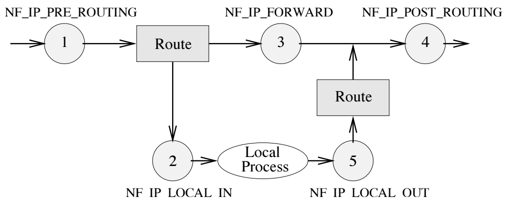


FIGURE 1: NETFILTER HOOKS

Therefore we can model three kind of traffic flows, depending on the destination:

- Traffic going through the firewall, in other words, traffic not going to the local machine. Such traffic follows the path: PREROUTING FORWARD POSTROUTING.
- Incoming traffic to the firewall, for example, traffic for the local machine. Such traffic follows the path: PREROUTING INPUT.
- Outgoing traffic from the firewall: OUTPUT POSTROUTING.

One can register a callback function to a given hook. The prototype of the callback function is defined in the structure nf_hook_ops in netfilter.h. This structure contains the information about the hook to which the callback will be registered, together with the priority. Since you can register more than one callback to a given hook, the priority indicates which callback is issued first. The register operation is done via the function nf_register_hook(...).

The callbacks can return several different values that will be interpreted by the framework in the following ways:

- ACCEPT: Lets the packet keep traveling through the stack.
- DROP: Silently discards the packet.
- QUEUE: Passes the packet to userspace via the nf_queue facility. Thus a userspace program will do the packet handling for us.
- STOLEN: Silently holds the packet until something happens, so that it temporarily does not continue to travel through the stack. This is usually used to collect defragmented IP packets.
- REPEAT: Forces the packet to reenter the hook.

In short, the framework provides a method for registering a callback function that does some kind of packet handling at any of the stages previously detailed. The return value issued will be taken by the framework that will apply the policy based on this verdict.

If at this point you consider the information provided here to be insufficient and need more background about the Linux network stack, then consult the available documentation [2] about packet travel through the Linux network stack.

The Connection Tracking System and the Stateful Inspection

The days when packet filtering policies were based uniquely on the packet header information, such as the IP source, destination, and ports, are over. Over the years, this approach has been demonstrated to be insufficient protection against probes and denial-of-service attacks.

Fortunately, nowadays sysadmins can offer few excuses for not performing stateful filtering in their firewalls. There are open source implementations available that can be used in production environments. In the case of Linux, this feature was added during the birth of the Netfilter project. Connection tracking is another brick built on top of the Netfilter framework.

Basically, the connection tracking system stores information about the state of a connection in a memory structure that contains the source and destination IP addresses, port number pairs, protocol types, state, and timeout. With this extra information, we can define more intelligent filtering policies.

Moreover, there are some application protocols, such as FTP, TFTP, IRC, and PPTP, that have aspects that are hard to track for a firewall that follows the traditional static filtering approach. The connection tracking system defines a mechanism to track such aspects, as will be described below.

The connection tracking system does not filter the packets themselves; the default behavior always lets the packets continue their travel through the network stack, although there are a couple of very specific exceptions where packets can be dropped (e.g., under memory exhaustion). So keep in mind that the connection tracking system just tracks packets; it does not filter.

STATES

The possible states defined for a connection are the following:

- NEW: The connection is starting. This state is reached if the packet is valid, that is, if it belongs to the valid sequence of initialization (e.g., in a TCP connection, a SYN packet is received), and if the firewall has only seen traffic in one direction (i.e., the firewall has not yet seen any reply packet).
- ESTABLISHED: The connection has been established. In other words, this state is reached when the firewall has seen two-way communication.
- RELATED: This is an expected connection. This state is further described below, in the section “Helpers and Expectations.”
- INVALID: This is a special state used for packets that do not follow the expected behavior of a connection. Optionally, the sysadmin can define rules in iptables to log and drop this packet. As stated previously, connection tracking does not filter packets but, rather, provides a way to filter them.

As you have surely noticed already, by following the approach described, even stateless protocols such as UDP are stateful. And, of course, these states have nothing to do with the TCP states.

THE BIG PICTURE

This article focuses mainly in the layer-3 independent connection tracking system implementation nf_conntrack, based on the IPv4 dependent ip_conn_track, which has been available since Linux kernel 2.6.15. Support for specific aspects of IPv4 and IPv6 are implemented in the modules nf_conntrack_ipv4 and nf_conntrack_ipv6, respectively.

Layer-4 protocol support is also implemented in separated modules. Currently, there is built-in support for TCP, UDP, ICMP, and optionally for

SCTP. These protocol handlers track the concrete aspects of a given layer-4 protocol to ensure that connections evolve correctly and that nothing evil happens.

The module nf_conntrack_ipv4 registers four callback functions (Fig. 1) in several hooks. These callbacks live in the file nf_conntrack_core.c and take as parameter the layer-3 protocol family, so basically they are the same for IPv6. The callbacks can be grouped into three families: the conntrack creation and lookup, the defragmented packets, and the helpers. The module nf_conntrack_ipv6 will not be further described in this document, since it is similar to the IPv4 variant.

IMPLEMENTATION ISSUES

BASIC STRUCTURE

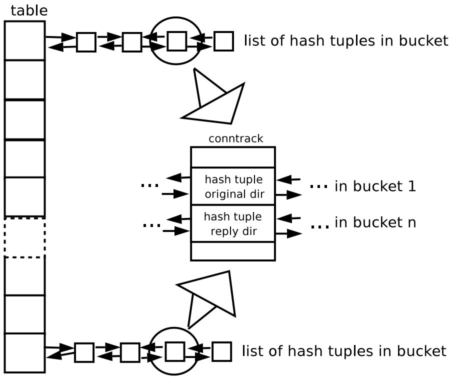


FIGURE 2: CONNECTION TRACKING STRUCTURE

The connection tracking system is an optional modular loadable subsystem, although it is always required by the NAT subsystem. It is implemented with a hash table (Fig. 2) to perform efficient lookups. Each bucket has a double-linked list of hash tuples. There are two hash tuples for every connection: one for the original direction (i.e., packets coming from the point that started the connection) and one for the reply direction (i.e., reply packets going to the point that started the connection).

A tuple represents the relevant information of a connection, IP source and IP destination, as well as layer-4 protocol information. Such tuples are embedded in a hash tuple. Both structures are defined in nf_conntrack_tuple.h.

The two hash tuples are embedded in the structure nf_conn, from this point onward referred to as *conntrack*, which is the structure that stores the state of a given connection. Therefore, a conntrack is the container of two hash tuples, and every hash tuple is the container of a tuple. This results in three layers of embedded structures.

A hash function is used to calculate the position where the hash tuple that represents the connection is supposed to be. This calculation takes as input parameters the relevant layer-3 and layer-4 protocol information. Currently, the function used is Jenkins' hash [3].

The hash calculation is augmented with a random seed to avoid the potential performance drop should some malicious user hash-bomb a given hash chain, since this can result in a very long chain of hash tuples. However, the conntrack table has a limited maximum number of conntracks; if it fills up, the evicted conntrack will be the least recently used of a hash chain. The size of the conntrack table is tunable on module load or, alternatively, at kernel boot time.

THE CONNTRACK CREATION AND LOOKUP PROCESS

The callback nf_conntrack_in is registered in the *PREROUTING* hook. Some sanity checks are done at this stage to ensure that the packet is correct. Afterward, checks take place during the conntrack lookup process. The subsystem tries to look up a conntrack that matches with the packet received. If no conntrack is found, it will be created. This mechanism is implemented in the function resolve_normal_ct.

If the packet belongs to a new connection, the conntrack just created will have the flag confirmed unset. The flag confirmed is set if such a conntrack is already in the hash table. This means that at this point no new conntracks are inserted. Such an insertion will happen once the packet leaves the framework successfully (i.e., when it arrives at the last hook without being dropped). The association between a packet and a conntrack is established by means of a pointer. If the pointer is null, then the packet belongs to an invalid connection. Iptables also allows us to untrack some connections. For that purpose, a dummy conntrack is used.

In conclusion, the callback nf_conntrack_confirm is registered in the LOCAL INPUT and POSTROUTING hooks. As you have already noticed, these are the last hooks in the exit path for the local and forwarded traffic, respectively. The confirmation process happens at this point: The conntrack is inserted in the hash table, the confirmed flag is set, and the associated timer is activated.

DEFRAGMENTED PACKET HANDLING

This work is done by the callback ipv4_conntrack_defrag, which gathers the defragmented packets. Once they are successfully received, the fragments continue their travel through the stack.

In the 2.4 kernel branch, the defragmented packets are linearized, that is, they are copied into contiguous memory. However, an optimization was introduced in kernel branch 2.6 to reduce the impact of this extra handling cost: The fragments are no longer copied into a linear space; instead, they are gathered and put in a list. Thus all handling must be fragment-aware. For example, if we need some information stored in the TCP packet header, we must first check whether the header is fragmented; if it is, then just the required information is copied to the stack. This is not actually a problem since there are available easy-to-use functions, such as skb_header_pointer, that are fragment-aware and can linearize just the portion of data required in case the packet is defragmented. Otherwise, header-checking does not incur any handling penalty.

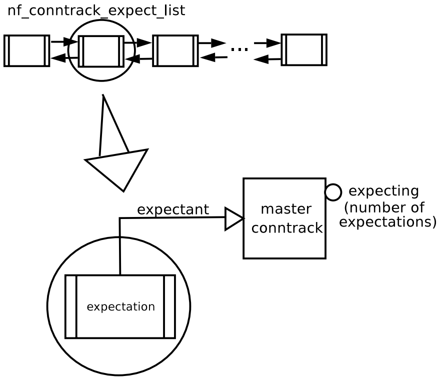


FIGURE 3: RELATIONSHIP BETWEEN A CONNTRACK AND AN EXPECTATION

HELPERS AND EXPECTATIONS

Some application-layer protocols have certain aspects that are difficult to track. For example, the File Transfer Protocol (FTP) passive mode uses port 21 for control operations to request some data from the server, but it uses TCP ports between 1024 and 65535 to receive the data requested instead of using the classical TCP port 20. This means that these two independent connections are inherently related. Therefore, the firewall requires extra information to filter this kind of protocol successfully.

The connection tracking system defines a mechanism called *helpers* that lets the system identify whether a connection is related to an existing one. To do so, it defines the concept of *expectation*. An expectation is a connection that is expected to happen in a period of time. It is defined as an nf_conntrack_expect structure in the nf_conntrack_core.h file.

The helper searches a set of patterns in the packets that contain the aspect that is hard to track. In the case of FTP, the helper looks for the PORT pattern that is sent in reply to the request to begin a passive mode connection (i.e., the PASV method). If the pattern is found, an expectation is created and is inserted in the global list of expectations (Fig. 3). Thus, the helper defines a profile of possible connections that will be expected.

An expectation has a limited lifetime. If a conntrack is created, the connection tracking system searches for matching expectations. If no matching can be found, it will look for a helper for this connection.

When the system finds a matching expectation, the new conntrack is related to the master conntrack that created such an expectation. For instance, in the case of the FTP passive mode, the conntrack that represents the traffic going to port 21 (control traffic) is the master conntrack, and the conntrack that represents the data traffic (e.g., traffic going to a high port) is related to the conntrack that represents the control traffic.

A helper is registered via nf_contrack_helper_register, which adds a structure nf_conntrack_helper to a list of helpers.

Conclusions and Future Work

Netfilter's connection tracking system is not a piece of software stuck in time. There is considerable interesting work in progress targeted at improving the existing implementation. It is worth mentioning that during the 4th Netfilter Workshop [4], some work addressing replacing the current hash table approach with a tree of hash tables [5] was presented. The preliminary performance tests look promising.

Fortunately, the subsystem described in this document is accessible not only from the kernel side. There exists a userspace library called libnetfilter_conntrack that provides a programming interface (API) to the in-kernel connection tracking state table.

With regards to the helpers, support for Internet telephony protocols such as H.323 and VoIP are on the way. In addition, there is also some work in progress on providing the appropriate mechanisms to allow people to implement their own protocol helpers in userspace, a feature that Rusty dreamed of in the early days of the Netfilter Project.

ACKNOWLEDGMENTS

I would like to thank Harald Welte and Patrick McHardy for spending their precious time reviewing my contributions, as well as many others. Thanks are also owed to my Ph.D. director, Rafael M. Gasca (University of Seville, Spain), and to Laurent Lefevre and the RESO/LIP laboratory (ENS Lyon, France) for the student research period of February to July 2004.

REFERENCES

[1] Paul Russel and Harald Welte, “Netfilter Hacking How-to”: http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.txt.

[2] Miguel Rio et al., “A Map of the Networking Code in Linux Kernel 2.4.20,” Technical Report DataTAG-2004-1, FP5/IST DataTAG Project, 2004.

[3] Bob Jenkins, “A Hash Function for Hash Table Lookup”: http://burtleburtle.net/bob/hash/doobs.html.

[4] 4th Netfilter Workshop, October 2005: http://workshop.netfilter.org/2005/.

[5] Martin Josefsson, “Hashtrie: An Early Experiment,” October 2005: http://workshop.netfilter.org/2005/presentations/martin.sxi.

MARKOS GOGOULOS AND DIOMIDIS SPINELLIS

using Linux live CDs for penetration testing

Markos Gogoulos is a research assistant in the ELTRUN Software Engineering and Security Group at the Athens University of Economics and Business and a free software movement enthusiast.

mgogoulos@gmail.com

Diomidis Spinellis is an associate professor in the Department of Management Science and Technology at the Athens University of Economics and Business and author of the books *Code Reading: The Open Source Perspective* (Addison-Wesley, 2003) and *Code Quality: The Open Source Perspective* (Addison-Wesley, 2006).

dds@aueb.gr

WHAT WOULD YOU THINK IF IN minutes you could have a full Linux system with almost all the necessary tools for penetration testing and security auditing, without having to install it on a dedicated machine? Whether you are a security professional or a system administrator, a bootable Linux live CD can be your best friend.

What Is Penetration Testing?

Penetration testing is a focused attempt to look for security holes. These can be design weaknesses or technical flaws and vulnerabilities in critical resources for a network. The test focuses on a network's infrastructure, servers, and workstations. Penetration testers try to break into a network, attempting to locate and document all security flaws, so that they can be fixed. Usually penetration testers are supplied with specific instructions as to which systems and networks to test. If you are to undertake such an effort, make sure you obtain written permission from a person authorized to give it, before even preparing for the test. Also notify all system administrators whose systems will be affected, because the test may create a heavy traffic load on the network and generate intrusion-detection system alerts. Penetration testing is quite similar to hacking—that's why it is also called ethical hacking—but differs in that it is arranged and approved by the network's owner and aims at locating all security flaws. This contrasts with hacking, where the goal is typically to find a single series of flaws that is sufficient for system intrusion. Whereas in hacking creativity has a major impact on the results and an instinctive, probably self-developed procedure is being followed, professional penetration testing involves the use of a methodology that will be followed to assure that results are accurate and complete.

The Need for a Methodology

A penetration testing methodology provides a framework that is followed to ensure that the results will be accurate and complete. As far as we know, the only publicly available methodology for penetration testing is the *Open Source Security Testing Methodology Manual* (OSSTMM) [1]. As quoted on OSSTMM's site:

> The OSSTMM is a peer-reviewed methodology for performing security tests and metrics. The OSSTMM test cases are divided

> into five channels (sections) which collectively test: information and data controls, personnel security awareness levels, fraud and social engineering control levels, computer and telecommunications networks, wireless devices, mobile devices, physical security access controls, security processes, and physical locations such as buildings, perimeters, and military bases. The OSSTMM focuses on the technical details of exactly which items need to be tested, what to do before, during, and after a security test, and how to measure the results. New tests for international best practices, laws, regulations, and ethical concerns are regularly added and updated.

OSSTMM is publicly available for downloading. If followed, OSSTMM ensures that a thorough penetration testing has been undertaken. It also comes with Report Requirements Templates, to assist in the creation of final reports, and a legal penetration testing checklist, containing features to consider, such as privacy and protection of information and authorization for the test. Note that OSSTMM does not give instructions on how to accomplish the penetration testing or what tools to use for it; there are numerous sites on the Internet and books for this task, along with institutions and companies that will happily charge you to attend their seminars and get (a portion of) this knowledge.

Open Source or Proprietary Tools?

Security-related tools exist in both OSS and commercial platforms. Most of the commercial tools are generally more professional looking; however, keep in mind that these are difficult or impossible to modify to fit your needs, and their cost is often significant. Moreover, there are no commercial tools available for several tasks. Also, commercial tools are often created after OSS tools have been available for some time, and therefore such tools lag in the technologies they use. Typical examples of this state of affairs are current WEP analysis and cracking tools. Many OSS security-related tools are maintained by a large team of people, and hundreds of developers contribute to the project. Generally, OSS tool updates are more frequent and signatures for vulnerability assessment tools for the newly discovered vulnerabilities are added soon after they are publicly available. In this area the reflexes of the OSS community appear to be far quicker, and therefore the best tools for penetration testing are not commercial.

What Is a Linux Live CD?

Linux live CDs are Linux systems based on a certain distribution that operate from the distribution CD-ROM without the need to set up the system and without the use of the local hard drive. They perform automated hardware configuration with great success. As a result, within a few minutes from booting, you'll have in front of you a full graphical Linux environment, with all the peripherals identified and a number of preinstalled programs ready to be used. One category of Linux live CDs targets security. Most of those CDs are based on Knoppix or Slax distributions. (Knoppix is a distribution based on Debian, whereas Slax is based on Slackware.)

Alternatives

Live CD distributions for security can be split into the following categories: Penetration Testing, Forensics, and Secure Desktop. The Forensics category focuses on tools for the noninvasive study and retrieval of data from various types of file systems, whereas the Secure Desktop distributions focus on programs and servers providing secure protocol implementations. Penetration Testing live CDs include programs for enumeration, network scanning and analysis, vulnerability assessment, and exploitation of security vulnerabilities.

A system for penetration testing requires a lot of work to set up, as it involves gathering the programs, installing them, and keeping them up-to-date. A live CD for penetration testing, such as the ones that we will examine here, saves you this effort.

Typically a penetration testing CD will contain the following:

- attack and penetration testing tools
- enumeration tools
- tools for scanning and network port analysis
- vulnerability scanners targeting known problems
- CIFS (SMB) scanners
- sniffers and network analyzers
- tools for the exploitation of common vulnerabilities (e.g., Metasploit Framework, Exploit Tree)
- HTTP proxy tools
- fuzzer tools
- tools for router scanning and exploitation
- tools for spoofing and session hijacking
- tools for password cracking and brute-force attacks

Let's go through some of the available live CDs for penetration testing. You can locate the live CDs in the security category of the frozentech list.[2] All distributions comprise a basic set of penetration testing tools (nmap, nessus, nikto, Metasploit Framework) plus some additional tools to make the system more functional, such as editors, Web browsers, and image viewers. You can see a summary of the features of some prominent distributions in Table 1.

Our personal favorite is the Auditor security collection [3]: It includes all the tools we listed, and more. What we like most about Auditor is the organization of the programs into separate categories, its orientation toward professional administrators, and its cutting-edge functionality. In the wireless sector, the Auditor truly shines, coming with the most complete tool collection for wireless network penetration testing. Some of those programs, such as the wireless LAN scanner Kismet, are notorious for their time-consuming and difficult installation; with Auditor this functionality comes out-of-the-box. Furthermore, Auditor uniquely incorporates tools for Bluetooth penetration testing.

Although some tools are missing from Auditor, with a little additional work an installed system can be transformed into a state-of-the-art base for penetration testing. For example, tools we found missing from Auditor are those for database auditing, for Novell Netware auditing, and for SMB and Kerberos sniffing. Some of these tools exist for Linux, whereas others can operate through Wine. Furthermore, it would be desirable if the system had, by default, read/write capabilities for NTFS file systems. In addition, one could add the Achilles and Spike Web interception proxies; apart from their other capabilities, these automatically test Web applications for buffer overflows and SQL injection.

From the other distributions that we examined we found Whax [4] and KCPentrix [5] most interesting. Both distributions include features that Auditor lacks. For example, Whax contains snort accompanied with acid and other front-ends, as well as tools for vulnerability enumeration through the so-called Google hacking techniques. In the vulnerability scanners category, Whax has modules for the scanner Retina and Foundstone tools operating through Wine (both Windows tools). Furthermore, Whax includes tools for database auditing: for instance, Absinthe for blind SQL injection, and other tools for auditing Oracle and Cisco systems. Beyond the Metasploit Framework, an advanced open-source platform for developing, testing, and using exploit code, Whax includes Exploit Tree, a properly supported exploit source code base with an update capability. In addition, Whax contains several exploit collections for client-side attacks: vulnerabilities for Internet Explorer as well as exploit archives from the securityfocus.com, packetstormsecurity.com, and milworm.com sites. Both Whax and KCPentrix are founded on Slax and therefore share many features, with Whax offering slightly more material.

The Phlak [6] live CD consists of only a few tools. What impresses us in Phlak is its accompanying security-oriented documentation, which is well organized in different categories. We found this to be very useful and think that other distributions could benefit from adopting this approach. For example, OSSTMM could be included on a security-related live CD.

	GUI	*System Apps*	*Installation Program*	*Vulnerability Scanners*	*Exploit Tools*	*Version in 2005*	*Documents/ Penetration Testing Material*	*Wireless Pen*	*Bluetooth Pen*
Auditor	Y	Y	Y	Y	Y	Y	N	Y	Y
Whax	Y	Y	Y	Y	Y	Y	N	Y	N
KCPentrix	Y	Y	N	Y	Y	Y	N	Y	N
Phlak	Y	Y	Y	Y	Y	Y	Y	Y	N
Knoppix-std	Y	Y	Y	Y	N	N	N	Y	N

TABLE 1. DISTRIBUTION COMPARISON TABLE

Penetration Testing

Often the penetration testing process is presented as a mixture of science and art. Furthermore, complete penetration testing involves something more than the simple execution of various vulnerability scanners targeting some systems: The penetration tester aims at tracing *all* the possible violation pathways, by following a well-defined methodology.

Even if the penetration testing results depend on the knowledge and skills of the penetration tester, there are some tasks that are most customarily followed. Usually, you will initially enumerate the systems or the networks that are to be tested, to obtain basic information about them, for example IP address ranges, gateways, and administrator names. Subsequently, through port scanning, you will locate open ports and services that are running on them. Any network service is a potential door to the system. Services that currently run may be vulnerable to a known vulnerability, something that a vulnerability scanner will show, but they can also be traced manually if you get a connection to an open port, read the banner, and afterward check if the service version is vulnerable to some flaw.

Most services will reveal their version from a banner with little effort, but even those tailored not to reveal such information can be tricked sometimes. It is important to locate all existing shares in Windows systems or NFS exports in UNIX. With brute-force tools, you can try to crack passwords that give access to shares or to the system, through SSH, FTP, Web protocols, webmin, or another service. By using a sniffer you can see unencrypted protocols (a formerly common and controversial pastime at USENIX conferences), as well as passwords or other sensitive data that pass through the network. For example, a few years ago, one of us used a sniffer to demonstrate to the public that sensitive data used in a particular setup of a popular e-government application were being transmitted in plaintext form. You can also use Ettercap and Dsniff to perform more sophisticated attacks, utilizing somewhat esoteric techniques, such as ARP spoofing for sniffing through switches. Several other tools that are incorporated in Auditor allow you to test network security and to locate risky setups through spoofing, traffic injection, or DHCP flooding.

When you locate vulnerabilities, you will have to try to exploit them before documenting possible solutions, to ensure that you don't report any false positives or false negatives. For example, an application may be lying about its version, or it may have been configured with a workaround to avoid a particular vulnerability. This is where tools like the Metasploit Framework come in. These tools allow you to avoid false positives and directly check for security gaps. In addition, with such tools you can demonstrate the actual problems, because sometimes system administrators know of certain problems in their network, but they fail to address them, in the mistaken belief that their network is not at risk.

In light of the fact that in many networks Web applications—which are most probably supported by a database—house valuable assets, you'll need to test them separately for how they behave on unexpected input, SQL injection, and other attacks. You could perform this job using tools such as Nikto, Spike, Achilles, or Paros.

Discussion

Obviously, these tools are extremely powerful and in the hands of unauthorized people they cause many problems and chaos on a network. Some may claim that distributions such as Auditor make it easier for script kiddies and other wrongdoers to accomplish their attacks. However, nowadays anyone with a browser can easily find information about the programs Auditor contains; try, for example, Googling for "dhcp flooder." Script kiddies would require some additional effort to install them; eventually though, the tools will work for them.

Conclusions

With a live CD like Auditor you as a system administrator could run Nessus periodically on your systems to check whether there are any security-related problems, or you could use it as a base system for a more complete penetration test. Most of the live CDs we examined allow you to install tools not included in the distribution, and some of the tools support the automated downloading of updates. Both features will help you keep your penetration testing system up-to-date. When the time for downloading the updates becomes excessive, just burn a CD with an updated distribution. Finally, keep in mind that these distributions are maintained by unpaid volunteers; don't forget that these projects depend on contributions from our community for maintenance and improvements.

REFERENCES

[1] Open Source Security Testing Methodology Manual (OSSTMM): http://www.osstmm.org.

[2] Frozentech list with live CDs for security: http://www.frozentech.com/content/livecd.php?pick=All&showonly=Security&sort=&sm=1.

[3] Auditor security collection: http://www.remote-exploit.org/index.php/Auditor_main.

[4] Whax: http://www.iWhax.net.

[5] http://www.kcpentrix.net/Site/.

[6] Phlak: http://www.phlak.

The Fund to Establish the John Lions Chair in Operating Systems at the University of New South Wales

USENIX announces the creation of a matching fund to establish the John Lions Chair in Operating Systems at the University of New South Wales.

The University of New South Wales is establishing an endowed Chair to recognize the enormous contribution made by John Lions to the world of computing. USENIX will match up to $250,000 in donations made through USENIX, now through December 31, 2006. To donate, see below.

The Chair, to be called the John Lions Chair in Operating Systems, will enable an eminent academic to continue the John Lions tradition of insightful and inspirational teaching in operating systems. The creation of the Chair will perpetuate the John Lions name, and new generations of students will benefit from his legacy.

HOW DO I DONATE TO THE JOHN LIONS FUND?

USENIX will match your donation to the John Lions Fund, now through December 31, 2006. Donations can be made by sending a check, drawn on a U.S. bank and made out to the USENIX Association, to:

John Lions Fund
USENIX Association
2560 Ninth St., Suite 215
Berkeley, CA 94710

or by making a donation online at https://db.usenix.org/cgi-bin/lionsfund/donation.cgi.

Your contribution may be tax-deductible as allowed by law under IRS Code Section 501(c)(3). Check with your tax advisor to determine whether your contribution is fully or partially tax-deductible.

DAVID BLANK-EDELMAN

practical Perl tools: Car 10.0.0.54, where are you?

David N. Blank-Edelman is the director of technology at the Northeastern University College of Computer and Information Science and the author of *Perl for System Administration* (O'Reilly, 2000). He has spent the last 20 years as a system/network administrator in large multiplatform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the chair of the LISA '05 conference.

dnb@ccs.neu.edu

WHEN SOMEONE ASKS ME ABOUT Web services, I hear this loud buzzing sound, because those words are all the latest rage. Give it a year or two, and either the ballast will get changed so they don't buzz as much, or people will have moved on to something else. In the meantime, let's take a look at something in that ballpark.

If this were a more theory-oriented column we might talk about the fundamentals of Web services. We'd probably look at how Web services are sometimes just an extension of the standard client-server module in that they often entail one server consuming the output from another server as part of performing a task for a user. We'd note that XML is a key component of many Web services because it provides a lingua franca/Esperanto in which these server-to-server conversations can take place. Having a well-structured language for this purpose makes it much easier to write the software for either end of the transaction. Mention of XML would no doubt lead to talk of more complex protocols built on XML such as XML-RPC and SOAP. WSDL (the Web Services Description Language), a way of describing the possible conversations for a Web service, would also be a natural segue. For good measure, we might even get into REST (REpresentational State Transfer) as another way of thinking about Web services.

But that's all good material for a column with a different bent. This time we're going to focus on something a little more fun in the general vicinity of Web services. If you are interested in the fundamentals (and you probably should be), there's a decent *Programming Web Services with Perl* book by Randy J. Ray and Pavel Kulchenko. However, today's topic is one of my favorite Web service application realms: geocoding.

Geocoding from Postal Addresses

Let's start with one of the standard tasks: Given a postal address of some sort, is it possible to locate that address on the planet such that we could plot it on a map? This is one example of a process known as geocoding. Doing geocoding well (where well means "could use it for commercial applications") is actually a fairly hard problem for a number of reasons, including all the data being suspect. Postal addresses can be ambiguous, the geographical data are sometimes incomplete or incorrect, and both humans and nature are always changing the surface features of the planet. This is all said to help form a disclaimer that holds true for everything in this column. Try the examples here, but don't depend too heavily on them. If you need professional geocoding done, hire a professional.

> Disclaimer 1: I have no commercial or other relationship to the various Web service providers mentioned in this article beyond occasionally paying for the cheaper ones so that I can play with them.
>
> Disclaimer 2: Often when people in the United States talk about geocoding, they really mean "North America geocoding" and are much less concerned with finding points outside of the U.S. Setting aside the standard U.S. ethnocentrism, we see that this is also a function of the availability of data. The U.S. government makes a passable data set available for free; most other countries don't have an equivalent. If you are interested in non-U.S. geocoding, the people at www.nacgeo.com have a relatively inexpensive commercial offering that may suit your needs.

If we leave out the expensive for-pay geocoding services, there are still a few geocoding methods available to us. The first one Perl people tend to turn to is the geocoder.us Web site/service provider, because they provide not only a free set of Web services but also the Geo::Coder::US module on CPAN should you desire to set up your own server. geocoder.us offers several different flavors of Web service, including XML-RPC, SOAP, REST, and "plaintext" REST. We're going to pick XML-RPC to start with, because the code to use it is very simple:

```
use XMLRPC::Lite;
my $reply = XMLRPC::Lite
    -> proxy ( 'http://rpc.geocoder.us/service/xmlrpc' )
    -> geocode( '2560 9th Street, Berkeley, CA')

    -> result;

foreach my $answer (@{$reply}){
    print "lat: "   . $answer->{'lat'}
        . " long: " . $answer->{'long'} . "\n";
}
```

First we load the XMLRPC::Lite module, which is bundled in the SOAP::Lite distribution. The proxy() method (which, despite its name, doesn't have anything to do with a Web proxy or any other kind of proxy) is used to specify where the query will be directed. We make our remote call out to that server using the geocode() method and ask XMLRPC::Lite to return the result.

The code for printing the result may look a little more complex than necessary. geocode() returns a list of hashes, one hash per result of the query. Some queries can yield multiple answers (e.g., if we asked for "300 Park, New York, NY" there might be a 300 Park Street, a 300 Park Drive, and a 300 Park Lane). There's only one 9th Street in Berkeley, so it would have been easier (but less robust) to write the following:

```
print "lat: "  . $reply->[0]->{'lat'}  .
      "long: " . $reply->[0]->{'long'} . "\n";
```

If you decide for some reason that you don't like the results you receive from geocoder.us, there are a number of other cheap geocoding services available; these include ontok.com (but be warned that later versions of SOAP::Lite do not play nicely with its SOAP interface) and Yahoo!'s REST-based geocoding API (for fewer than 5000 queries a day). Let's look at the latter. To use this service, we need to apply for a free application ID at http://api.search.yahoo.com/webservices/register_application. With that ID we can then use the API described at http://developer.yahoo.com/maps/rest/V1/geocode.html. Here's some sample code to do that:

```
use LWP::Simple;
use URI::Escape;
use XML::Simple;

# usage: scriptname <location to geocode>
my $appid  = "{your appid here}";
my $requrl = "http://api.local.yahoo.com/MapsService/V1/geocode";

my $request = $requrl .
    "?appid=$appid&output=xml&location=" . uri_escape( $ARGV[0] );

my $response = XMLin( get($request), forcearray => ['Result'] );

foreach my $answer ( @{$response->{'Result'}} ){
    print "Lat: $answer->{Latitude} " .
          "Long: $answer->{Longitude} \n";
}
```

One of the pleasant properties of REST interfaces is that they are really easy to query. If you know how to retrieve a Web page in Perl using a GET or PUT, you can use a REST interface. In the preceding example, we construct the URL by taking the base Yahoo! REST request URL and adding a few parameters, that is, the required appID, our preferred output format, and a URL-encoded version of the location to query. This gets handed to LWP::Simple's get() routine, the output of which we immediately parse using XML::Simple.

XML::Simple would ordinarily hand us back a hash that contained a single hash if the geocode server returned a single response. If the server returned several answers—remember the ambiguous address case in our last example—it would provide a hash that contained a list of hashes, one for each answer. When it came time to display the results, we could have written code to distinguish between the single answer data structure and the multianswer data structure, using ref(), and act accordingly, but that's too much work. Instead, we take the easy way out and ask XML::Simple (via forcearray =>['Result']) to always hand us back a hash with a list of hashes. The code for results output then gets to do an easy foreach walk over that list.

Note that if this code seems a little too complex for you, there's even a simpler way to do it courtesy of the Geo::Coder::Yahoo module. This module has exactly two calls in it, one to create the search object and another to call the geocoding API. The latter call returns a list of hashes, with no XML parsing required. Use whichever one suits your fancy.

Now that we've seen a couple of ways to turn an address into its corresponding latitude and longitude, what can we do with that information? The obvious answer to this question is to plot the information on a map. There are a number of good Web services for doing this, including Google Maps (http://www.google.com/apis/maps/), Yahoo! Maps (http://developer.yahoo.com/maps/), and TerraServer (http://terraservice.net/webservices.aspx). For fun, you can generate KML or KMZ (compressed KML) files for Google Earth (http://earth.google.com/kml/) and fly between your data points.

The process of plotting geocode data into one of these maps usually involves fiddling with HTML and that icky Javascript stuff. In Perl we luck out for Google Map creation, because Nate Mueller has written an

HTML::GoogleMaps module that makes the process really easy. Here's a sample CGI script that displays a map with labeled marker pointing at the USENIX mothership:

```
use HTML::GoogleMaps;

my $coords = [-122.291713, 37.859524]; # 2560 9th Street, Berkeley, CA
my $map = HTML::GoogleMaps->new( key => '{your api key here}' );
$map->center( $coords );          # center it on the address
$map->zoom( 2 );                  # zoom it to street level

# add a marker at the address using the given html as a label
# (and don't change the size of that label)
$map->add_marker(
   point    => $coords,
   noformat => 1,
   html     => "<a href='http://www.usenix.org'>USENIX</a> office" );

# add some map controls (zoom, etc)
$map->controls( "large_map_control", "map_type_control" );

# create the parts of the map
my ($head, $body, $js) = $map->render;

# output the HTML (plus CGI-required Content-Type header) for that map
print "Content-Type: text/html\n\n";
print << "EOH";
<html>
<head>
<title>;login test</title>
$head
</head>
EOH

print "<body> $body $js</body> </html>\n";
```

Note: The above code uses HTML::GoogleMaps to generate output for the Google Maps v1 API. A few days after this article was submitted for publication, Google released version 2 of their Maps API. They are pushing developers to upgrade before v1 is decommissioned. Luckily, the author of HTML::GoogleMaps is hard at work and should have a v2-compliant update to his module by the time you read this.

There's much more that can be done with Google Maps and the other services. Be sure to check out the respective documentation for these services and products.

Geocoding from IP Addresses

Let's circle back to the original question that started this column, namely, "Given a postal address of some sort, is it possible to locate that address on the planet?" It seems eminently doable that one could take a postal address and look it up on some list to find its coordinates. That seems like something you can picture rows and rows of clerks in little green visors doing in a big, nondescript office somewhere in the Midwest.

It's a lot more magical if I tell you, "Give me the name of your computer on the Internet and I can make a guess as to where that computer is located." There's something about crossing over the virtual/physical divide that makes this task seem all the more impressive. There are a number of reasons (besides impressing people at parties) for wanting to geocode from an IP address, and we'll get to those in a minute as well.

The first step of the process is to turn the DNS fully qualified domain name into an IP address. That's straightforward with the Net::DNS module:

```
use Net::DNS;
my $resolv = Net::DNS::Resolver->new;

my $query = $resolv->search( $ARGV[0] );

die "No response for that query" if  !defined $query;

# only print addresses found in A resource records
foreach my $resrec ( $query->answer ){
    print $resrec->address . "\n" if ($resrec->type eq "A");
}
```

Chances are you'll only be geocoding a name that has one IP address associated with it, but the code listed here tries to give you back all of the addresses returned in response to your query. Note that if you plan to do this sort of lookup many times (e.g., when parsing a log file), you'll want to maintain a cache of your results as you go along so you can avoid beating up the name servers needlessly. If you plan to process massive amounts of data, you'll probably want to look into some of the asynchronous DNS libraries such as adns (http://www.chiark.greenend.org.uk/~ian/adns/) to handle parallel queries well.

Now that we have an IP address in hand, it is time to bring Web services back into the picture. There are a few fairly cheap (for the amount of data I push through them) providers. The following examples use the service provided by maxmind.com, because that is the one I've played with the most. We're going to concentrate on Web services, but it should be noted that MaxMind and several other providers offer both a Web services interface to their data and a database subscription that allows you to download the data to your server for faster lookups.

For MaxMind's Web service, we just need to construct a simple HTTP GET (or PUT, if that is your fancy) similar to what we did for the Yahoo! API in a previous example. The main difference between that example and this one is the format returned. Here we get Comma/Character Separated Values (CSV) instead of something in XML format:

```
use LWP::Simple;
use Text::CSV_XS; # This is the faster version of Text::CSV

# usage: scriptname <IP address to geocode>

my $maxmkey  = "{maxmind key here}";

my $requrl = "http://maxmind.com:8010/f";
my $request = $requrl . "?l=$maxmkey&i=$ARGV[0]";

my $csvp = Text::CSV_XS-> new(); # (or Text::CSV->new())

    $csvp->parse(get($request));

my ($country, $region, $city, $postal, $lat, $lon,
    $metro_code, $area_code, $isp, $org, $err) = $csvp->fields();
```

You've already seen what we can do with the results of a latitude and longitude geocoding; let's briefly look at how the other fields could be pressed into service.

If we ran some code against our Web server log, we could use $country to create a nice Web page showing the flags from the countries that have visited the site that day. There are a number of places to get the flag data. For example, ip2location.com, one of MaxMind's competitors, offers a whole set of tiny flag gifs available for download for free from http://www.ip2location.com/products.asp. If you prefer larger flags from a more interesting source, the Geo::CountryFlags module will download them on the fly for you from the Central Intelligence Agency's World Factbook. That's a simple process:

```
use Geo::CountryFlags;
# returns the path to the flag file it downloaded or undef if not found

my $path = Geo::CountryFlags->new->get_flag('{country code}');
```

On a more techie note, we could use the information on what country the request comes from to direct someone to their closest mirror Web site, provide a reasonable default in a Web form asking for address information, or even provide the site in a different language. There are some helpful Perl modules that use a local database (from one of the subscription services mentioned earlier) to handle this process for you. For example, if you use Apache::Geo::Mirror, then the documentation points out that you can put this in your Apache configuration:

```
PerlModule Apache::Geo::Mirror
<Location /CPAN>
  PerlSetVar GeoIPDBFile "/usr/local/share/geoip/GeoIP.dat"
  PerlSetVar GeoIPFlag Standard
  PerlSetVar GeoIPMirror "/usr/local/share/data/mirror.txt"
  PerlSetVar GeoIPDefault us
  PerlHandler Apache::Geo::Mirror->auto_redirect
</Location>
```

and your Web server will automatically redirect a client to the right mirror site based on the country associated with its IP address.

To wind this column down with a last Web services flourish, let's end with one more fun use of this sort of data. If we geocode an IP address associated with a U.S. address and get back a zip code, it is easy to provide the weather forecast for that zip code. I know of at least four U.S. weather services that are free for noncommercial use:

- NOAA's National Weather Service has a SOAP-based service; details are at http://www.weather.gov/xml/.
- Weather.com provides an XML-based service; details are at http://www.weather.com/services/xmloap.html (though it comes with a whole boatload of requirements you have to satisfy if you want to use it on your Web site).
- Yahoo! provides weather information via RSS; see http://developer.yahoo.com/weather/. You'll need to parse the RSS format using something like XML::RSS (or even XML::Simple).
- http://www.rssweather.com also provides weather info via RSS.

To end this column, let's put several of these parts together. The following is a CGI script that attempts to determine your zip code from your IP address and then queries Yahoo! for your current weather conditions and forecast:

```
use LWP::Simple;
use Text::CSV_XS;
use XML::RSS;

my $maxmkey  = "{maxmind key here}";
my $requrl  = "http://maxmind.com:8010/f";
my $request = $requrl . "?l=$maxmkey&i=$ENV{'REMOTE_ADDR'}";
my $csvp = Text::CSV_XS->new();

$csvp->parse( get($request) );
my ($country, $region, $city, $postal, $lat, $lon,
    $metro_code, $area_code, $isp, $org, $err) = $csvp->fields();

print “Content-Type: text/html\n\n”;
print << “EOH”;
<html><head><title>;login test</title></head>
<body>
EOH
print “<p>Hi there “ . $ENV{‘REMOTE_ADDR’} . “!</p>\n”;

if ($postal) {
    my $rss = new XML::RSS;
    $rss->parse(
      get(“http://xml.weather.yahoo.com/forecastrss?p=$postal”) );
    print “<h1>” . $rss->{items}[0]->{‘title’} . “</h1>\n”;
    print $rss->{items}[0]->{‘description’}, “\n”;
}
print “</body></html>\n”;
```

Pretty cool, eh?

And with that, I’m afraid we have to bring this issue’s column to a close. Take care, and I’ll see you next time.

[Correction: In my last column I pointed people at the PUGS project (www.pugscode.org). At the time I wrote the column it hadn’t come to my attention that the developer who started the project had changed her name to Audrey. My apologies to Ms. Tang for the mistake.]

ROBERT HASKINS

ISPadmin: policy enforcement

Robert Haskins has been a UNIX system administrator since graduating from the University of Maine with a B.A. in computer science. Robert is employed by Shentel, a fast-growing network services provider based in Edinburg, Virginia. He is lead author of *Slamming Spam: A Guide for System Administrators* (Addison-Wesley, 2005).

rhaskins@usenix.org

IN THIS EDITION OF ISPADMIN, I TAKE a look at the policy enforcement area. This is a critical area for service providers who need to provide existing or new services to their customers in a low-cost, accurate, but quickly provisioned fashion. As a direct result of policy enforcement systems, the provider can more accurately track its customer's services while at the same time reducing its cost to acquire and retain subscribers. Although policy enforcement is directly related to Remote Authentication Dial In User Sevices (RADIUS), it really is a combination of everything a service provider does: provide service, authenticate users, bill subscribers, and more.

Policy Enforcement Background

Policy enforcement is the ability to apply access control to services across a network in a consistent, sane manner. It is related to the ideas of authentication and authorization, which are both critical to all service provider operations. After all, if you are allowing anyone to access your services, you probably aren't making much money! To be specific, authentication is the act of proving without a shadow of a doubt "who the user is," and authorization is "what services that user is allowed to access."

For example, an Apache .htaccess/.htpasswd file combination can be thought of as a simple policy enforcement system. This is because it controls who is allowed to access what resources on an Apache Web server. In a similar fashion, a UNIX passwd file controls who has access to the system, but it is more difficult to specify what services that user is allowed to access on the system in question. On traditional UNIX systems, policy enforcement is usually accomplished by a combination of additional access files beyond the passwd file.

In a hosted Web service provider environment, the group file can be used to control access to uploaded Web server files. In a similar fashion, fields in the passwd file can control access to what that user can do on the host. For example, the SHELL field in the passwd file might be set to /etc/nologin in the case of a host running a POP3 server. This would effectively disable interactive logins for the POP3 user but allow access to that user's mailbox. Of course, these mechanisms lie outside of any network-based controls on the host, such as iptables firewall or TCP wrapper.

If you are familiar with the traditional ISP dial-up network, you might be aware that the RADIUS protocol is often used to authenticate and authorize users (as well as account for them). The RADIUS protocol has excellent policy control abilities, enabling equipment manufacturers to define their own features and control mechanisms by virtue of the Vendor-Specific attribute in the RADIUS dictionary. (For a background on RADIUS, please see the April 2001 ISPadmin column titled "RADIUS" as well as [1].)

A Short History and Policy Enforcement Vendors

Historically, policy enforcement systems (like many parts of service provider operations) were developed in-house. RADIUS-only policy enforcement engines continue to form the basis of many ISPs' operations. However, if the provider wants features such as subscriber self-provisioning and/or next-generation services (video, voice, gaming, etc.) then plain vanilla RADIUS-based solutions won't work.

On the traditional telephone company side (i.e., non–IP-based network policy control), companies such as Lucent and Nortel have been the big players. Of course, a telephone company's proprietary switch must have the associated company's proprietary policy control engine to control it, because telephone systems usually lack open standards for provisioning and controlling their services. However, with the advent of IP networks and associated openness for provisioning services, the policy management arena has blossomed. Companies that have products in this market include Broadhop [2], Bridgewater Systems [3], and Tazz Networks [4].

What Are Policy Enforcement Systems?

Policy enforcement engines control access to services on a provider's network. These systems can take many forms and can be quite specialized in the case of traditional telephone networks. An example of a simple policy enforcement engine would be RADIUS. In fact, policy enforcement engines in the IP world are often built around service-provider-grade RADIUS systems. However, modern IP-based policy enforcement engines handle much more functionality than just RADIUS. Some of the additional services provided by enforcement engines include the following:

- DHCP services
- Subscriber self-provisioning/upgrading
- Subscriber/customer support
- Billing system interface for accounting detail
- Plan/package management

These services will be examined in some detail in the next sections.

DHCP SERVICES

DHCP services don't have to be integrated into the policy engine, but the provider gets a higher degree of control if they are. For example, if the provider wants to offer a service that uses a device that doesn't support RADIUS (e.g., some game consoles or VoIP handsets), then assigning an IP address via DHCP and the associated MAC address is often the only way that this can be done. Without policy control of the DHCP server, integration is much harder at best, or impossible in the worst case.

SUBSCRIBER PROVISIONING AND SUPPORT

One big reason for implementing policy control is to reduce subscriber signup and support costs. This can be accomplished by implementing systems that allow a subscriber to sign up as a new customer, add or change services, view his or her bill, and perform other functions, all without incurring the cost of a phone call to the support center. This is accomplished by simply integrating the provider's support and signup Web site into the policy enforcement system (if one already exists).

Allowing customers to add services also increases the likelihood that impulse purchases will occur. For example, if the subscriber knows that merely pressing the "turbo" button will increase DSL speed from 0.5M b/s to 3 Mb/s for 60 minutes to download a large file six times as fast, it is much more probable that the subscriber will buy the service. The easier it is for a customer to buy a product, the more likely it is that the customer will buy it.

BILLING SYSTEM INTEGRATION

Integration into the service provider's billing system is the key to successful deployment of policy-based systems. Often, the service plans offered by the provider are in the billing system and must be transferred to the policy enforcement engine easily and quickly. Alternatively, the policy control engine must give the provider the ability to create and manage the billing plan and associated services if no direct integration with the billing system is warranted.

In addition to plans, RADIUS accounting records must be transferred to the provider's billing system so that customers can be billed. Sometimes, rating can be done on the accounting records prior to sending billing detail to the billing system for updating customer records.

Of course, newly provisioned customers must be sent to the provider's billing platform. Any new or changed customer data (resulting from signups or service changes) must be transferred to the billing system as well, so that the master billing database is kept up-to-date. In IP-based policy engines, this can be done much more easily than was possible in the past by utilizing a standard XML interface.

Policy Enforcement and Equipment

End-user access devices such as a dial-in remote access server (RAS), BRAS (DSL access equipment), and wireless access points must interact with the policy enforcement engine. These devices actually enforce the policy served by the policy engine. Often, the policy enforcement software must be programmed to support the device even though the policy engine acts just like a "normal" RADIUS server. This is a result of the tight integration between the RADIUS server and other components of the policy control engine.

There are a number of devices that can be used as a gateway device to enforce policy where no such device exists (e.g., RAS or BRAS). Cisco has implemented its Service Selection Gateway (SSG) software in its current IOS releases. Of course, the Cisco hardware platform must support the SSG capability [5]. Other gateway device manufacturers include Nomadix [6] and Colubris [7]. Another lower-cost option would be a "roll your own" solution using mini-ITX or Soekris hardware (see the October 2005

ISPadmin column for background). Although the “do it yourself” price is right, it does take some work to set up your own device to act as a gateway/policy enforcement device.

Conclusion

In this edition of ISPadmin, I’ve looked at what the policy enforcement engine is and how it fits into the service provider environment. Policy enforcement is critical to any service provider wanting to reduce its operating cost while improving the level of service to the customer. Enforcement engines are implemented as commercial software packages, owing to their specific application in service provider environments. Policy enforcement ties together many of the disparate services utilized by a service provider, including RADIUS, DHCP, billing, provisioning, and customer signup. Gateway devices such as Nomadix, Colubris, and Cisco’s SSG IOS version are often used to implement policy on end-user connections.

REFERENCES

[1] RADIUS-related RFC listing: http://www.freeradius.org/rfc/.

[2] Broadhop home: http://www.broadhop.com/.

[3] Bridgewater Systems home: http://bridgewatersystems.com/.

[4] Tazz Networks home: http://www.tazznetworks.com/.

[5] Cisco 6400 policy enforcement device:
http://www.cisco.com/en/US/products/hw/routers/ps314/products_data_sheet091-86a008007ce99.html.

[6] Nomadix AG 3000 home: http://www.nomadix.com/products/platforms/ag3000.

[7] Colubris Multi Service Controller: http://www.colubris.com/global-wireless-network-management/multiservice-controllers.asp.

Background on policy management engines:
http://www.lightreading.com/document.asp?doc_id=77367.

HEISON CHAK

VoIP watch

Heison Chak is a system and network administrator at SOMA Networks. He focuses on network management and performance analysis of data and voice networks. Heison has been an active member of the Asterisk community since 2003.

heison@chak.ca

WE COVERED THE VARIOUS POTENTIALS and benefits of VoIP in the last issue of *;login:*. This month we will spend some time investigating how we can take advantage of VoIP for some toll-bypass fun. Whether you are engulfed in technologies or are involved in managing the tight budget that allows your staff to play with these technologies we will hopefully keep you thinking about or occupied by VoIP.

Building and deploying your own VoIP platform to bypass expensive telecommunication costs among partners and employees can be fun. With the amount of involvement in planning, designing of features, and managing and maintaining stability of the system, coupled with user expectations, VoIP managers and operators can be overwhelmed by the workload very quickly. Offloading some of these responsibilities to existing VoIP communities can leverage the technology and allow creation of an innovative way of communicating within an organization without much subsequent overhead.

Private VoIP Service Providers and Communities

Some VoIPs do not work well with NAT because they use layer 3 addresses in a layer 4 protocol. For example, an SDP message (the payload within a SIP packet) may contain the private RFC1918 address for which the SIP INVITE (beginning of a SIP conversation) may have originated. If the SIP recipient beyond a NAT firewall tries to respond and contact this private address, the conversation can never be established. There are different techniques for getting around this issue by mangling layer 3 addresses in layer 4 protocols so that replies can be routed properly.

Of the various protocols widely used today, SIP seems to have dominated over its competitors. Even though SIP shares some of the limitations and restrictions around NAT gateways and firewalls much like its predecessor (i.e., H.323), many still prefer the protocol because of its wide acceptance.

There are a handful of private VoIP communities supporting various protocols and applications, including, among others:

- Free World Dial-Up (SIP)
- Skype (proprietary protocol)
- Gizmo project (SIP)
- Vonage (SIP)

Whether they are bridging between a VoIP user and the PSTN world or providing peer-to-peer (P2P) communication over the Internet, these private communities allow your VoIP packets to travel within their framework. Although pure VoIP communication is certainly a free service, it is important to understand that VoIP service isn't always a giveaway—especially when one of the legs of the conversation originates from or terminates on the PSTN. The costs are usually associated with the capacity (or the lack thereof, in this case) that is and can be provisioned on the PSTN service subscribed by the VoIP service provider (or whoever is providing the bridge between VoIP and PSTN).

Free World Dial-Up (FWD, http://www.freeworlddialup.com), one of the first such private services, is based on SIP. Members can connect to other members by dialing an account ID or via a SIP URI. FWD has supernode(s) with which a client application (also known as a SIP User Agent) registers.

The SUA can be a soft SIP phone running on a PC, or it can be a dedicated computer with an embedded OS and a real handset and keypad (i.e., IP phones). Besides P2P communication, FWD also supports delivery to regular telephone numbers in cities around the world via SIP, as well as offering a global calling plan that allows calls to be made to worldwide destinations at competitive rates.

With Cisco's migration from the H.323 protocol to SIP-based IP phones and Microsoft's introduction of MSN Messenger to replace its H.323-based NetMeeting conferencing software, there is no doubt that SIP is gaining in popularity. However, this does not preclude others from inventing their own VoIP protocol.

Skype is no longer just hype; it's proven to be of interest to many home users and business travelers, as its ease of installation and call quality surpasses some of the early IP telephony software running on Windows and Mac platforms. To expand its footprint, Skype has partnered with hardware manufacturers to maximize usage of their VoIP products.

The biggest obstacle between the open source community and Skype is the lack of openness in the Skype protocol. There are hacks for, say, an Asterisk server wanting to send VoIP traffic to the Skype network—through the use of a middleware machine running the Skype software. However, this is not an elegant solution; thus the Gizmo project was born. Gizmo is not yet an open source project, but it employs open standards—SIP, allowing someone to call to and from other SIP networks on hardware/software clients around the world. Gizmo is a relative newcomer compared to Skype, but both offer similar paid services—the ability to accept inbound PSTN calls from cities around the world and to terminate outbound calls to the PSTN for just a few pennies per minute.

Most people may not consider Vonage to dwell in the same realm as some of the VoIP communities discussed. In principle, Vonage allows P2P calling without any additional service charge with an ATA (analogue telephone adapter). But Vonage takes it one step further by replacing the traditional POTS line and providing all subscribers with call-in and call-out type services for a monthly charge.

Build Your Own VoIP Platform, and More . . .

With the availability of open source PBX and other telephony platforms, one can easily go about replicating the infrastructure of some of these private communities. All that is required is a machine that can route and bridge calls between the VoIP and the PSTN, using an appropriate protocol:

- SIP, with a way of mangling the layer 3 address described above
- IAX (Inter Asterisk Exchange) protocol, which is designed to work well with NAT

For simplicity, let us assume a 64-kbps CODEC (i.e., the ITU G.711), which is used since it is the most widely supported CODEC for sampling voice stream; an ATA device that supports DHCP for those who work remotely at their home office; and a software-based IP phone and a Bluetooth headset for traveling users to use the VoIP service where Internet is available, via Ethernet at airport lounges or a WiFi connection at hotel hotspots. Last but not least, provide a Web tool so that employees may forward their office extensions to wherever they wish—to a soft VoIP phone or a landline.

If there isn't already a corporate standard in Instant Messaging and VoIP, choosing one that does not rely on a proprietary protocol is highly recommended. Some may argue that software that supports open standards tends to have fewer compatibility issues and a quicker turnaround time for patch releases. The Gizmo project definitely places high on the list in this respect, as it mimics the capability of Skype with the use of SIP.

A Gizmo user can populate the contact list with friends who already have a Gizmo account; he or she can also use the same software to access company voicemail and connect to any IP-enabled extensions in the office while traveling. This is made possible by creating SRV records in the company's DNS:

```
_sip._udp        IN       SRV      20 0 5060 pbx.mycompany.com
```

SIP requests made to the URI sip:1508@mycompany.com (along with any SIP URI requests) will be directed to pbx.mycompany.com. In this case, using Asterisk as an example, extension 1508 can handled by directing the caller to a SIP-based IP phone, followed by an IAX-based IP phone, and finally to a voicemail account after 40 seconds of ringing:

```
exten => 1508,1,Dial(SIP/cisco_phone,20) ; using SIP
exten => 1508,2,Dial(IAX2/iax_phone,20)  ; using IAX no one answered after 20 seconds
exten => 1508,3,Voicemail(u1508)  ; finally, place the caller into voicemail
```

It is also possible to allow Gizmo users using Asterisk to bridge into another private VoIP community. Special extensions of Asterisk can be created to handle bridging of specific Vonage numbers, FWD numbers, or even another Skype user (using the aforementioned hack). If users want to make calls to the PSTN, they can choose either to use the native Gizmo CallOut features or to use a special extension on Asterisk to make outgoing calls and let the Asterisk dial plan decide how the call should be made (either via IP or the PSTN).

This process is good enough to support outgoing calls made from a Gizmo client to anyone:

- within the corporation
- with a Gizmo account
- with any other private VoIP community account
- on the PSTN

How about incoming calls? How can we provide a convenient way for partners and customers to contact employees without incurring steep long distance or international charges? Of course, the most favorable move is to advertise SIP URI contacts. Instead of calling the main office number followed by keying in extension numbers (or dialing the DIDs directly), the callee can simply become sip:ext@mycompany.com:

Heison Chak +1-416-977-1414 (ext. 1508) or +1-416-348-1508 becomes
sip:1508@mycompany.com

Gizmo and CounterPath (formerly Xten) X-Lite (another SUA) are capable of connecting to such a URI from Windows, Linux, or Mac OS.

To support callers on the PSTN who prefer to dial directly from their cell phones and landlines, simply giving out a SIP URI is not helpful, unless their communication devices can handle such context—although such capability shouldn't be too far off, given that there are already cell phones that are Skype-capable. However, until this capability is widely available, the easiest workaround is to find a provider who can supply a SIP-based call-in number in or near the city from which most overseas calls are made (i.e., where the callers are based).

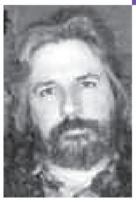
Currently, Gizmo provides call-in numbers in the United Kingdom and in the United States, and Skype with is bigger landscape can provide numbers in 13 different countries around the world (covering Europe, Asia, North America, and Latin America). Until the Gizmo project increases its coverage significantly, one may need to do more research to find local SIP providers that allow soft-phone options. For example, a VoIP account with Hong Kong Broadband costs around U.S. $6.15/month, and Asterisk can be set up to register with the VoIP server so that incoming calls from the PSTN will be delivered via the Internet to a machine that is physically hosted in North America:

```
sip.conf:
register => 1234567:password@hkbn.net/1234 ; Register 1234567 at SIP provider as 1234 here
extensions.conf:
exten => 1234,1,Dial(SIP/cisco1)        ; calls dest. for 1234567 from HK will ring IP phone cisco1
```

This can provide great savings to a North American–based corporation for communicating with overseas customers or partners. However, since VoIP accounts are not managed under the umbrella provider (e.g., Gizmo or Skype), be prepared for considerable management headaches in judging whether the savings are worth it. For a corporation with high call volume, it may be. Otherwise, it is still a fun IT project for you fellow admins out there.

ROBERT G. FERRELL

/dev/random



Robert is a semiretired hacker with literary and musical pretensions who lives on a small ranch in the Texas Hill Country with his wife, five high-maintenance cats, and a studio full of drums and guitars.

rgferrell@greatambience.com

HUMAN EXISTENCE IS A SERIES OF nested loops. The parent loop is the cycle of birth, death, and reincorporation of one's component parts into another generation via decomposition. (We'll ignore spirituality for now; I don't have that much space.) Tucked snugly within that master rotation are a plethora of secondary loops: Babies grow up to have babies; sports cars are purchased, wrapped around utility poles, and replaced; consumer electronics become obsolete before we get them home and must be upgraded; garbage cans are filled with packing peanuts, microwave popcorn containers, and blister-pack debris, dragged to the curb, and dragged back. For those of us in the IT industry, especially, there is another familiar iteration that I will call the "employment-go-round." Every three to five years the urge to change jobs/personnel seems to come over us/our employers, respectively. Various solutions for escaping from this carnival ride have been proposed, but I can personally testify that at least one of them does not function as advertised. Allow me to elaborate.

Back in 1996 I was working as a defense contractor in the systems department of a large Air Force hospital. It was a decent job, but the parent company kept getting bought out—to the extent that we had a betting pool on what corporate logo would be at the top of our checks the following pay period. With every buyout I ran the risk that I wouldn't be kept on by the new owners, although given my entrenched position in the organization and the insane profit margin they were making by keeping me there, in retrospect there wasn't any good reason for anxiety. Still, I eventually lost patience with the constant uncertainty and when a job offer as a senior UNIX systems administrator for the federal government came along, even though it entailed a move to the Washington, D.C., area, I jumped at the chance to provide myself and my family with some serious job security. Or so I thought.

I'd always heard that once you were a permanent federal employee the paperwork required to terminate you for anything short of gross misconduct was too daunting for even the most seasoned bureaucrat to bother with. Incompetency wasn't so much a liability as a side-effect of employment, from management's point of view—especially since it was often that very trait that got the managers their own jobs. Federal employment might be boring and tedious at times, but it was a reliable dullness that paid relatively well until retirement.

Completely wrong. First of all, it wasn't at all boring or tedious. It was, if anything, at least as frenetic and challenging as any of my private sector positions, and that's pushing the frenzy/challenge envelope. Turns out the iron-clad job security part was also so much horse hockey, although admittedly it did take me almost nine years to make that dark discovery. There's a giant hole in the mythical federal job security blanket through which even a fairly hefty geek can slip like a well-greased melon. It's called "failure to accept a directed reassignment," and basically it means that if your federal employer decides to relocate your position to Perspiration, Nevada, and you don't want to move, you're fired. Period. Personally, I'd be a lot more amenable to simply being told, "We don't want you anymore: Go away!" but this way appears more politically correct, I suppose. Yeah, I could have appealed, but who really wants to work for someone who's being forced to accept you as an employee? Not exactly a cordial environment in which to spend nearly a third of your life. I must needs move on; the circle remains unbroken.

Of course, I'm still a career status federal (ex) employee with what's called "reinstatement eligibility," but that and $4.50 will get you a double Latte Mocha Coconut Frappuccino Macchiato Valencia with extra cinnamon (provided you've got a half-off coupon). I've endured a dozen or so interviews both within and outside the federal government since hitting the streets, but none of them has led to what I would consider a firm job offer. My resume is too long, I guess, or maybe I'm using the wrong font. I've had lots of jobs in my life that are in no way connected with IT, including analytical chemist, enologist, cancer researcher, ornithologist, corporate security administrator, technical writer/editor, professional musician, and radiological safety officer, so I probably just confuse HR people or leave them with the impression that I'm a pathological liar. As a result, while many people pad their resumes, I'm considering stripping mine down (and hope that no one notices the considerable gaps between "conventional" jobs). Today's job hunting tip brought to you by Henry David Thoreau.

Long gone, apparently, are the days when job candidates with a wide breadth and depth of experience were considered valuable assets. The job market today is singularly myopic. It turns its monochrome visage your way only briefly, passing you over for the android standing next to you if you don't fall instantly through a well-worn slot. I have nearly 100 graduate semester hours in assorted disciplines of biology, but I'm never considered qualified as a biologist because that wasn't the title of the last job I held. Yet, whenever I interview for an IT position (in which field I've been working, on and off, since 1977), I invariably get some permutation of the question, "Why do you want a job in IT when your academic training is in biology?" I lower my gaze and reply in shame, "Because I've had a bad habit since childhood of wanting to eat occasionally and sleep relatively unaffected by passing meteorological phenomena." It probably goes without saying that most of these interviews are of the cursory variety.

When next you encounter a bureaucrat with a bad comb-over wringing his hands and whining about the lack of experienced information technologists willing to come work for the government, pull on your hip waders, 'cause it's gettin' deep. I'm here to tell you, that dog won't hunt: They don't even want the ones they have *now*.

book reviews

ELIZABETH ZWICKY

zwicky@greatcircle.com

with Sam Stover, Heison Chak, and Rik Farrow

MIND PERFORMANCE HACKS: TIPS & TOOLS FOR OVERCLOCKING YOUR BRAIN

Ron Hale-Evans

O'Reilly, 2006. 308 pages.
ISBN 0-596-10153-8.

This is a nice book, with a lot of good advice in it. I made the mistake of recommending it to my husband, and then had to pry it out of his hands to review it. Because it contains a lot of disparate stuff, different parts of it are going to appeal to different people. For instance, the author puts an early emphasis on memorization tricks, which I think are flashy but not of much use in daily life, but then he gets into what I think of as the good stuff: managing the information in your life, creativity, decision making, communication, mental clarity. If you are, for instance, a medical student, then memorization tricks may actually be good stuff from your point of view. There are also a bunch of math tricks, some of which I find handy (including a couple I didn't know) and some of which are too much work for the amount of math I do. But what I think is amusing but impractical might be exactly what gets you going.

There is one hack where I think Hale-Evans got it 90% wrong. Hack #74 quite correctly recommends Karen Pryor's marvelous book *Don't Shoot the Dog*. It then promptly turns around and recommends the two least effective methods of self-training available: bribery and punishment. This is so wrong-headed as to be just silly. Pick up *Don't Shoot the Dog* for an explanation of why these are ineffective and what you should do instead. And yes, she explicitly addresses self-training, both teaching yourself new habits and getting rid of bad habits.

I also think that you'd be better advised to seek out a foreign language than an artificial language. Artificial-language writers tend to go for only minorly mind-blowing moves, whereas an in-depth study of a foreign language gets you all sorts of subtle and not-so-subtle changes and access to an entire culture. And a broad study of foreign languages will get you into things like Polynesian pronouns and Swahili or Navaho noun classifiers, which will give you novel ways of breaking up the world. The regularization that goes into artificial languages removes some of the best parts of language study. If you want to make up your own language, you should definitely go study half-a-dozen real languages from different language families first. (Yeah, that is a lot of work. I guess we can tell where my heart is—I'm willing to learn foreign languages for amusement but not finger arithmetic.)

PERFECT PASSWORDS: SELECTION, PROTECTION, AUTHENTICATION

Mark Burnett

Syngress, 2006. 178 pages.
ISBN 1-59749-041-5.

There are a few nice, novel ideas here, and some clear explanations of important concepts (you'd be surprised how many people think a 14-character password is twice as good as a 7-character password). Most of it is not going to be news to experienced administrators, but Burnett's book would be a great gift for people just encountering the password issue, and there are a few lovely ideas, of the "simple but life-changing" form. When you run across particularly annoying and counterproductive password policies, you could force a copy of this book on the authors.

That said, I found it hard to love. I think I would have liked it better if it were shorter; it began to feel repetitive pretty fast. If you're an administrator, it will give you good ideas about what sort of password policy you should have, and what sort of passwords you, personally, should pick, but it offers scant help on communicating a password policy to users, enforcing it, or even making it possible on systems designed for short passwords. And it really, really does focus on what to do when you're faced with a reusable, text password; there's minimal discussion of other options. OK, so that's what it says it's about, but surely a mention of one-time passwords wouldn't go amiss instead of several pages of random character strings?

SOFTWARE SECURITY: BUILDING SECURITY IN

Gary McGraw

Addison-Wesley, 2006. 406 pages. ISBN 0-321-35670-5.

You are a software security person. You are surrounded by crazy developers, who say things like "But what does it matter what encryption algorithm we use?" and "Well, yeah, that would be bad. But nobody would ever do that." Or, perhaps, you are trying to build something out of pieces you strongly suspect were built by such developers. This book is meant for you. It gives you the tools to explain what is wrong and why it's wrong, and if you are part of the development process, the tools to get it right.

If you are a developer who wants to find out about this security stuff, this book will probably work for you too. It certainly tries to explain the issues to developers who don't understand them, but I'm not certain how convincing it will be to somebody who's security-naive or security-hostile.

In general, I like this book a lot, for giving a general explanation of security in the software development process. The weakest section is chapter 2, where it explains a risk management framework. There's lots of supporting verbiage about how difficult the material is, but I came to the conclusion that the material itself isn't particularly mind-boggling; it just hasn't been freed from business-ese enough to make it palatable, so there are lots of sentences such as, "Management of risks, including the notion of risk aversion and technical tradeoffs, is deeply impacted by business motivation." Given that the author can write lucidly about difficult security concepts, either there is a strong effect of a previous author or the author suffers from a common speech impediment, in which a business context causes a sudden inability to communicate rationally.

LINUX DEBUGGING AND PERFORMANCE TUNING TIPS AND TECHNIQUES

Steve Best

Prentice Hall, 2005. 427 pages.
ISBN 0-13-149247-0.

Since I ended up with a bunch of Linux performance tuning books in my first review batch, I've become curious about Linux performance tuning books. This one is very straightforward, mostly talking about the nuts and bolts of how to use the tools—the command line options, what the output looks like, how to set up your situation so you can use a tool. It covers a wide variety of tools and gives good examples of how they can be used and how you interpret the results.

If you have a specific problem to solve, and a general introductory-level understanding of performance tuning and debugging, and you want to know what Linux tools are available and how to use them, this is a good tool for that purpose. It is not something you would want to read from end to end; it's more of a reference work. If you want a general education on Linux performance, Ezolt's *Optimizing Linux Performance* is still the way to go (see my review in October 2005).

PENETRATION TESTER'S OPEN SOURCE TOOLKIT

Johnny Long, Aaron W. Bayles, James C. Foster, Chris Hurley Vincent Liu, Mike Petruzzi, Noam Rathaus, SensePost, and Mark Wolfgang

Syngress, 2006. 704 pages.
ISBN 1-59749-021-0.

Reviewed by Sam Stover

I'd like to clarify a critical point about this book: It is not just a collection of howto's for all of the programs included on the Auditor LiveCD distribution. It is a step-by-step guide into the many facets of penetration testing which uses the Auditor LiveCD to provide most of the tools needed.

If you are looking for a quick reference on how to use every tool included on the Auditor LiveCD, this book will disappoint you, and, honestly, it should. You should be Googling for that. If, however, you would like to learn more about a particular area of pentesting, or the discipline as a whole, you will love this book.

The first two chapters start out of the gate with reconnaissance and enumeration. From there the chapters become a bit more application-specific, focusing on databases, Web servers, wireless networks, and, finally, network infrastructure devices. The last seven chapters deal with either writing your own pentesting tools or using the two most ubiquitous pentesting frameworks, Nessus and Metasploit.

I found that this book worked great as a reference for areas where my knowledge was lacking. For example, I haven't spent much time pentesting databases. So I turned to page 149 and dove in. The chapter was clean, easy to read, and to the point. There were tips for Oracle and Microsoft databases, as well as suggestions for how to make a database more secure.

Nessus and Metasploit get a fair degree of special attention throughout the book, as well they should. Not only are there chapters dedicated to each (four chapters on Nessus and two on Metasploit), but they are also discussed when appropriate in the other chapters. In the database chapter, there is a section on the Nessus database checks tucked between OScanner and SQLAT.

One thing that budding pentesters fail to realize is that the real value in a pentest is not in pointing out the deficiencies but in making suggestions on how to fix them. This book gives you both sides of the equation, which also means that this book should be on the bookshelf of any system, security, or network admin. If you are responsible for a Web farm, why not use the same tips and tricks that the pentesters are using? You don't even have to go out and find the tools: They're already on the included Auditor LiveCD.

In short, I think this book should appeal to a wide and varied audience. Experienced pentesters probably won't find anything new here, but people looking to jump into the industry, as well as any admin, will find this book to be a easy and fun introduction into the mentality and tools of penetration testing.

VOIP HACKS

Ted Wallingford

O'Reilly, 2006. 306 pages.
ISBN 0-596-10133-3.

Reviewed by Heison Chak

As I was upgrading my Asterisk PBX server to the latest release, I started flipping through the pages of *VoIP Hacks*, hoping to get some inspirations from the 100 Internet Telephony tips and tools.

I found most of the hacks clearly written, with enough examples to explicate the descriptions, and there is a good balance among tools that could run on Linux, Windows, and Mac OS X. Hacks on how to use an Intel V.92 Winmodem card to replicate the Digium X100P FXO card (which is no longer carried by Digium) and intercepting a VoIP call on switched networks using ARP poisoning may be a little controversial, but isn't that what hacking is all about?

There were a few times when I wished there had been a little more detail. For example, when the book described examples of building a fax-to-email gateway using spandsp, an example on how to build an email-to-fax gateway could be the very next question on a reader's mind.

I really appreciate the work that went into the hacks; it recalls memories of the drawing board when I was building my VoIP environment with Asterisk. It is definitely a book for the beginner-to-intermediate VoIP enthusiast. Experts may find a lot of the ideas very familiar.

LINUX PATCH MANAGEMENT

Michael Jang

Prentice Hall, 2006. 262 pages.
ISBN 0-13-236675-4.

Reviewed by Rik Farrow

This is a book I wish I had had years ago, when I was tasked with creating a course about UNIX patch management. I found myself confounded with multiple versions of Linux, each with its own peculiar patching software. Jang does a fine job of covering all the major Linux distros, and some smaller ones as well.

Jang splits his focus between using and configuring individual tools, such as apt, yum, and YaST, and explaining how to set up local repositories of patches. Local repositories are important, not just to avoid beating up on your own network connection and the bandwidth of patch servers. Jang covers these issues well, and in enough detail, that you should be able to follow his instructions and set up your own patch repositories.

Jang does not deal with other issues involved in patch management, such as patch testing, reference systems, test deployments of patches, or staggering deployments, but focuses solely on the use of the tools. You can use this book to choose a Linux distro based upon the choice of patch management systems, as well as to support patching your existing Linux systems. I plan on keeping this book handy.

BUILDING EXTREME PCS

Ben Hardwidge

O'Reilly, 2006. 192 pages.
ISBN 0-596-10136-8.

Reviewed by Rik Farrow

This folio-sized book represents a departure from the usual run of O'Reilly products. Beautifully illustrated (in a very geeky sense) with full-color photos of cases, CPUs, water cooling systems, and more, Hardwidge's book takes you on a journey into the world of building extreme PCs that I believe will actually be useful for anyone building their own PC. Why? Because some of his tips will be useful to those who have chosen to DIY instead of buying the latest off-the-shelf clone PC.

Hardwidge includes discussions of all the key PC components. I found his primer on current CPU technologies very helpful, for example, as he explains the difference between current Intel and AMD offerings, differences in cache types, cache levels, etc. None of the explanations is very deep, but this may be exactly what you need when you want answers in a hurry. You might wonder why not Google for this, and I have tried, but I have found much better answers here.

Hardwidge really targets people building Windows systems for gaming, but he also includes silent PCs and PCs suitable for PVRs. *Building Extreme PCs* is almost a coffee-table book in the quality of the illustrations, depending on your taste (or your significant other's feelings about tech in the living room).

USENIX notes

USENIX MEMBER BENEFITS

Members of the USENIX Association receive the following benefits:

FREE SUBSCRIPTION to *;login:*, the Association's magazine, published six times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on such topics as security, Perl, Java, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

ACCESS TO ;LOGIN: online from October 1997 to this month: www.usenix.org/publications/login/.

ACCESS TO PAPERS from USENIX conferences online: www.usenix.org/publications/library/proceedings/

THE RIGHT TO VOTE on matters affecting the Association, its bylaws, and election of its directors and officers.

DISCOUNTS on registration fees for all USENIX conferences.

DISCOUNTS on the purchase of proceedings and CD-ROMs from USENIX conferences.

SPECIAL DISCOUNTS on a variety of products, books, software, and periodicals. For details, see www.usenix.org/membership/specialdisc.html.

FOR MORE INFORMATION regarding membership or benefits, please see www.usenix.org/membership/ or contact office@usenix.org. Phone: 510-528-8649

EXECUTIVE DIRECTOR

Ellie Young,
ellie@usenix.org

ELECTION RESULTS

The results of the election for Board of Directors of the USENIX Association for the 2006–2008 term are as follows:

PRESIDENT

Michael B. Jones,
mike@usenix.org

VICE PRESIDENT

Clem Cole,
clem@usenix.org

SECRETARY

Alva Couch,
alva@usenix.org

TREASURER

Theodore Ts'o,
ted@usenix.org

DIRECTORS

Matt Blaze,
matt@usenix.org

Rémy Evard,
remy@usenix.org

Niels Provos,
niels@usenix.org

Margo Seltzer,
margo@usenix.org

Not elected:
Gerald Carter

Please see http://www.usenix.org/about/elections06results.html for the details.

SAGE UPDATE

STRATA ROSE CHALUP

SAGE, the USENIX Special Interest Group for SysAdmins, is going strong. Here's an update on what's been happening.

The eagerly awaited *System Configuration* booklet by Paul Anderson has been mailed out to current SAGE members and is available online. In addition to a solid grounding in configuration management issues, Paul provides a comparative look at some of the popular tools, such as cfengine, LCFG, Active Directory + System Management Server, CDDLM, and more. Several pages of detailed references grace the back of the booklet, including case study and reference papers.

We're updating the User Groups section of the SAGE Web site with contact info and meeting dates for local groups. We have some "goodie bags" to send out to local groups for their members, but we don't always have a mailing address in addition to the email address! If you are a coordinator for a local group or would like to start one in your area, please get in touch at sage-locals-support@sage.org.

For those readers whose interest tends more toward kernels than configs, there is USENIX's User Groups program: see details at www.usenix.org/membership/ugs.html. Remember, there's no exclusivity or limit on affiliation of groups, so your group can be both a SAGE and a USENIX local group, as well as anything else it wants to be!

We're also pleased to announce a bevy of new and returning features to sage.org:

SAGE Programs Blog

As part of our effort to communicate with our members, we've set up a blog on the SAGE site for monthly Memos to Members, as well as for other content of interest. We'll be posting there several times monthly, and we may also have guest bloggers bringing their views on topics and events. Check it out at blogs.sage.org/strata/.

We're also establishing a blogroll of SAGE and USENIX community members. If you have a blog whose content would consistently be of interest to your fellow members, drop us a note at sage-blogroll@sage.org and we may add it to the blogroll. We'll feature occasional articles as well as links, so if you've posted something recently that you feel is hot and topical, let us know!

SysAdmin Toolkit v1.0

In addition to a SysAdmin Toolkit FAQ, we've put together a couple of handy one-page versions—a site toolkit and personal toolkit basics. Check it out at http://hoshi.sage.org/field/toolbox.html and get a sneak preview of the soon-to-be-unveiled new SAGE Web site.

Are you interested in helping out in maintaining and updating the toolbox? Feel strongly about whether we should have an update committee or just maintain a wiki and edit the responses into the FAQ periodically? Let us know. We'd like this to be a living document, although some things never change.

SAGE IRC Channel Is Back!

There's no shortage of IRC channels around, but some of us missed the old sage-members channel, graciously hosted for many years at another site. Now it's back, hosted at USENIX. Please drop in at irc.sage.org #sage-members and say hi.

Mentoring Self-Service Beta

The sage-members mailing list and IRC channel have long been sites of much informal mentoring in the community. There's a place for slightly more formal mentoring, however, and that's a need we haven't been filling particularly well. Most of us are sufficiently busy that something as potentially open-ended as "mentoring" seems a bit scary as a commitment. That's why we've been studying successful industry and academic mentoring programs to find a workable set of expectations. Good news—such things exist!

Enter the SAGE Self-Service Mentoring area, where you can seek a mentor or offer to become a mentor. We're getting a simple, streamlined beta set up for basic matching of mentors and folks wishing to be mentored. Of course, you may be both—perhaps you're a Solaris wiz who wants to learn more about storage networks, or vice versa. If you're interested in participating in the beta of our Self-Service Mentoring, drop a note to sage-mentoring@sage.org and we'll be in touch. We're especially interested in getting input on mentoring expectations and in getting some useful categories set up. We look forward to hearing from you.

Not a SAGE member? Dues are only $40. Find out more and join at http://www.usenix.org/membership/classes.html#sage.

USENIX and SAGE members: Help define emerging employment descriptions of your field! Please read on regarding an opportunity to participate in an important study.

Your assistance is requested with a critical program sponsored by the United States Department of Labor (USDOL) known as the Occupational Information Network (O*NET). The USDOL is gathering occupational information in an effort to better define worker characteristics such as skills, abilities, activities, and work context for workers in the technology sector of the U.S. economy. As the data is revised, it will be used by employers, workers, educators, and students (http://online.onetcenter.org). Much of the information is already in use by agencies and organizations across the country (http://www.doleta.gov/programs/onet/oina.cfm).

The O*NET program is seeking experts in the information technology field, as many related occupations are considered new and emerging since the last complete update conducted by the United States Department of Labor in the late 1970s. The data should be provided by those who have 5 or more years of experience in the noted occupation and have performed in that same arena during the past 6 months. This can include teaching, instructing, etc. A short description of the occupations currently being updated is listed below. Please use the description, not the title, to determine if you may be a good match:

1) O*NET-SOC Occupation Title: Computer Systems Engineers and Architects

Description: "Design and develop solutions to complex applications problems, system administration issues, or network concerns. Perform systems management and integration functions."

2) O*NET-SOC Occupation Title: Web Administrators

Description: "Manage web environment design, deployment, development, and maintenance activities. Perform testing and quality assurance of web sites and web applications."

3) O*NET-SOC Occupation Title: Network Designers

Description: "Determine user requirements and design specifications for computer networks. Plan and implement network upgrades."

4) O*NET-SOC Occupation Title: Computer Security Specialists

Description: "Plan, coordinate, and implement security measures for information systems to regulate access to computer data files and prevent unauthorized modification, destruction, or disclosure of information."

5) O*NET-SOC Occupation Title: Network Systems and Data Communications Analysts

Description: "Analyze, design, test, and evaluate network systems, such as local area networks (LAN), wide area networks (WAN), Internet, intranet, and other data communications systems. Perform network modeling, analysis, and planning. Research and recommend network and data communications hardware and software. This includes telecommunications specialists who deal with the interfacing of computer and communications equipment. Also, may supervise computer programmers."

USENIX is extending this invitation on behalf of the USDOL and O*NET to participate in the collection of information. Please be assured that your decision regarding participation in O*NET will not impact your standing as a member of USENIX. Participation is completely voluntary. RTI International (RTI), a nonprofit research firm, is assisting the Department of Labor with the O*NET data collection effort.

If you wish to participate, please send email indicating your interest in participating, along with your name, telephone number, and, most important, your occupation to Jean Leech at onetjleech@pub.rti.org or contact her at 877-233-7348 ext. 104. Do not contact the USENIX office directly.

Thanks in advance for your time and effort. By participating you will contribute to a key resource providing our nation's citizens with continuously updated occupational information and to overall international standards as a whole. We believe that the participation of our members will help immensely in seeing to it that this study is based on accurate information.

15th USENIX SECURITY SYMPOSIUM

VANCOUVER, B.C., CANADA July 31–Aug. 4, 2006

Electronic Voting Workshop at Security '06: Register Today!

2006 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '06)

Tuesday, August 1, 2006, Vancouver, B.C., Canada
http://www.usenix.org/evt06

EVT seeks to bring together researchers from a variety of disciplines, ranging from computer science and human factors experts through political scientists, legal experts, election administrators, and voting equipment vendors. The workshop will include short paper presentations as well as vibrant panel discussions with substantial time devoted to questions and answers. Attendance at the workshop will be open to the public, although speakers and presentations will be by invitation only.

Also to be held this year at Security '06, two by-invitation-only workshops:

First Workshop on Hot Topics in Security (HotSec '06)

Monday, July 31, 2006, Vancouver, B.C., Canada
http://www.usenix.org/hotsec06

HotSec is intended as a forum for lively discussion of aggressively innovative and potentially disruptive ideas in all aspects of systems security. Attendance will be by invitation only, limited to 35–40 participants, with preference given to the authors of accepted position papers/presentations.

First Workshop on Security Metrics (MetriCon 1.0)

Tuesday, August 1, 2006, Vancouver, B.C., Canada
http://www.usenix.org/metricon06

MetriCon 1.0 is intended as a forum for lively, practical discussion in the area of security metrics. It is a forum for quantifiable approaches and results to problems afflicting information security today, with a bias towards practical, specific implementations. Attendance will be by invitation only and limited to 50 participants. Preference will be given to the authors of position papers/presentations who have actual work in progress.

Preliminary Announcement and Call for Papers **USENIX**

3rd Workshop on Real, Large Distributed Systems (WORLDS '06)

Sponsored by USENIX, the Advanced Computing Systems Association

http://www.usenix.org/worlds06

November 5, 2006 **Seattle, WA, USA**

Co-located with the 7th Symposium on Operating Systems Design and Implementation (OSDI '06), which will take place November 6–8, 2006

Important Dates

Paper submissions due: *July 7, 2006, 11:59 p.m. PDT*
Notification to authors: *August 8, 2006*
Demo submissions due: *September 7, 2006*
Final papers due: *September 8, 2006*

Conference Organizers

Program Co-Chairs

David Andersen, *Carnegie Mellon University*
Neil Spring, *University of Maryland*

Preliminary Program Committee

Mike Afergan, *Akamai*
Mike Dahlin, *University of Texas, Austin*
Marc Fiuczynski, *Princeton University*
Michael Freedman, *New York University*
Krishna Gummadi, *Max Planck Institute for Software Systems*
Dina Katabi, *Massachusetts Institute of Technology*
Jay Lepreau, *University of Utah*
Dan Rubenstein, *Columbia University*
Martin Swany, *University of Delaware*
Matt Welsh, *Harvard University*
Janet Wiener, *Hewlett-Packard*
Ming Zhang, *Microsoft Research*

Overview

The 3rd Workshop on Real, Large Distributed Systems will bring together people who are exploring the new challenges of building widely distributed networked systems and who lean toward the "rough consensus and running code" school of systems building. WORLDS is a place to share new ideas, experiences, and work in progress, with an emphasis on systems that actually run in the wide area and the specific challenges they present for designers and researchers.

Workshop means the emphasis is on focused, fresh ideas and experience. Talks will be short (about 15 minutes long) to leave plenty of time for general discussion. Attendance will consist of contributors to the workshop and a subset of the OSDI attendees, with the number of non-contributors limited to encourage lively discussion between the participants.

Real means that the workshop will concentrate on systems designed to run on a real platform for a period of time. Such systems might be research projects, teaching exercises, or more permanent services, but they should address technical issues of actual widely distributed systems. We also welcome papers that explore the extent to which results obtained from simulation or testbed deployments retain validity when transferred to more representative network environments.

Large refers to the numerical and geographical dimensions of the system: WORLDS emphasizes distributed systems that span a significant portion of the globe and are spread over a large number of sites.

Submitting a Paper

Submissions should be at most 5 U.S. letter pages long, two-column format, using 10-point type on 12-point (single-spaced) leading within a 6.5" x 9" text block. Participants will be invited based on their ability to convince the program committee that they have built, are building, or are experimenting with a Real, Large Distributed System and have useful ideas, tools, experience, data, or research directions to share with the community that will stimulate discussion at the workshop. Submit your paper via the Web form at http://www.usenix.org/worlds06/cfp.

Online copies of the position papers will be made available before the workshop to registered attendees and will be added to the USENIX proceedings library after the workshop. Participants may update their papers to incorporate workshop feedback.

USENIX policy on simultaneous paper submission: Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

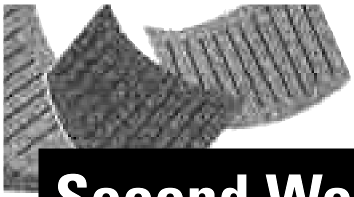
Authors uncertain whether their submission meets USENIX's guidelines should contact the program chairs, worlds06chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Demo Session

This year, WORLDS will again feature a demo session in which researchers will have the opportunity to demonstrate the real, running distributed systems they have built. Authors who have their full 5-page workshop papers accepted will automatically be granted the opportunity to present a demo. Others who wish to present a demo should submit a single-page demo description that (a) concretely describes the research problem solved by the system to be demonstrated and (b) concretely describes what will be shown at the demo. Submit your demo via the Web form at http://www.usenix.org/worlds06/cfp.

Awards

We expect to offer both a best paper award and a best demo award.



Announcement and Call for Papers **USENIX**

Second Workshop on Hot Topics in System Dependability (HotDep '06)

Sponsored by USENIX, The Advanced Computing Systems Association

http://www.usenix.org/hotdep06

November 8, 2006 **Seattle, WA, USA**

HotDep '06 will be held immediately following the 7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6–8, 2006.

Important Dates

Paper submissions due: *July 15, 2006 (firm deadline, no extensions)*
Notification of acceptance: *August 31, 2006*
Final papers due: *September 18, 2006*

Workshop Organizers

Program Co-Chairs

George Candea, *EPFL* and *Aster Data Systems*
Ken Birman, *Cornell University*

Program Committee

Lorenzo Alvisi, *University of Texas at Austin*
David Andersen, *Carnegie Mellon University*
Andrea Arpaci-Dusseau, *University of Wisconsin, Madison*
Mary Baker, *Hewlett-Packard Labs*
David Bakken, *Washington State University*
Christof Fetzer, *Technical University of Dresden*
Roy Friedman, *Technion—Israel Institute of Technology*
Indranil Gupta, *University of Illinois at Urbana-Champaign*
Farnam Jahanian, *University of Michigan and Arbor Networks*
Emre Kiciman, *Microsoft Research, Redmond*
Petros Maniatis, *Intel Research Berkeley*
Andrew Myers, *Cornell University*
David Oppenheimer, *University of California, San Diego*
Geoff Voelker, *University of California, San Diego*
John Wilkes, *Hewlett-Packard Labs*

Overview

Authors are invited to submit position papers to the Second Workshop on Hot Topics in System Dependability (HotDep '06). The workshop will be co-located with the 7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6–8, 2006. The HotDep '05 program is available at http://hotdep.org/2005.

The goal of HotDep '06 is to bring forth cutting-edge research ideas spanning the domains of fault tolerance/reliability and systems. HotDep will center on critical components of the infrastructures touching our everyday lives: operating systems, networking, security, wide-area and enterprise-scale distributed systems, mobile computing, compilers, and language design. We seek participation and contributions from both academic researchers and industry practitioners to achieve a mix of long-range research vision and technology ideas anchored in immediate reality.

Position papers of a maximum length of 5 pages should preferably fall into one of the following categories:

- describing new techniques for building dependable systems that represent advances over prior options or might open new directions meriting further study
- revisiting old open problems in the domain using novel approaches that yield demonstrable benefits
- debunking an old, entrenched perspective on dependability
- articulating a brand-new perspective on existing problems in dependability
- describing an emerging problem (and, possibly, a solution) that must be addressed by the dependable-systems research community

The program committee will favor papers that are likely to generate healthy debate at the workshop, and work that is supported by implementations and experiments or that includes other forms of validation. We recognize that many ideas won't be 100% fleshed out and/or entirely backed up by quantitative measurements, but papers that lack credible motivation and at least some hard evidence of feasibility will be rejected.

Topics

Possible topics include but are not limited to:

- automated failure management, which enables systems to adapt on the fly to normal load changes or exceptional conditions
- techniques for better detection, diagnosis, or recovery from failures
- forensic tools for use by administrators and programmers after a failure or attack
- techniques and metrics for quantifying aspects of dependability in specific domains (e.g., measuring the security, scalability, responsiveness, or other properties of a Web service)
- tools/concepts/techniques for optimizing tradeoffs among availability, performance, correctness, and security
- novel uses of technologies not originally intended for dependability (e.g., using virtual machines to enhance dependability)
- advances in the automation of management technologies, such as better ways to specify management policy, advances on mechanisms for carrying out policies, or insights into how policies can be combined or validated

Deadline and Submission Instructions

Authors are invited to submit position papers by 11:59 p.m. PDT on July 15, 2006. **This is a hard deadline—no extensions will be given.**

Submitted position papers must be no longer than 5 single-spaced 8.5" x 11" pages, including figures, tables, and references; two-column format, using 10-point type on 12-point (single-spaced) leading; and a text block 6.5" wide x 9" deep. Author names and affiliations should appear on the title page.

Papers must be in PDF format and must be submitted via the Web submission form, which will be available on the Call for Papers Web site, http://www.usenix.org/hotdep06/cfp.

Authors will be notified of acceptance by August 31, 2006. Authors of accepted papers will produce a final PDF and the equivalent HTML by September 18, 2006. All papers will be available online prior to the workshop and will be published in the Proceedings of HotDep '06.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program co-chairs, hotdep06chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Registration Materials

Complete program and registration information will be available in September 2006 on the conference Web site. The information will be in both HTML and a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list: http://www.usenix.org/about/mailing.html.

Announcement and Call for Papers **USENIX**

5th USENIX Conference on File and Storage Technologies (FAST '07)

USENIX, The Advanced Computing Systems Association, in cooperation with ACM SIGOPS, IEEE Mass Storage Systems Technical Committee (MSSTC), and IEEE TCOS

http://www.usenix.org/fast07

February 13–16, 2007 **San Jose, CA, USA**

Important Dates

Paper submissions due: *September 4, 2006, 9:00 p.m. EST (this is a firm deadline; sorry, no extensions)*
Notification of acceptance: *November 7, 2006*
Final papers due: *December 19, 2006*
Work-in-Progress Reports/Poster Session proposals due: *January 12, 2007*

Conference Organizers

Program Chairs

Andrea C. Arpaci-Dusseau, *University of Wisconsin, Madison*
Remzi H. Arpaci-Dusseau, *University of Wisconsin, Madison*

Program Committee

Ashraf Aboulnaga, *University of Waterloo*
Mary Baker, *Hewlett-Packard Labs*
Bill Bolosky, *Microsoft*
Scott Brandt, *University of California, Santa Cruz*
Randal Burns, *Johns Hopkins University*
Peter Corbett, *Network Appliance*
Mike Dahlin, *University of Texas, Austin*
Jason Flinn, *University of Michigan, Ann Arbor*
Dharmendra Modha, *IBM Almaden*
Erik Riedel, *Seagate*
M. Satyanarayanan, *Carnegie Mellon University*
Jiri Schindler, *EMC*
Margo Seltzer, *Harvard University*
Kai Shen, *University of Rochester*
Anand Sivasubramaniam, *Pennsylvania State University*
Muthian Sivathanu, *Google*
Mike Swift, *University of Wisconsin, Madison*
Amin Vahdat, *University of California, San Diego*
Carl Waldspurger, *VMWare*
Erez Zadok, *Stony Brook University*

Overview

The 5th USENIX Conference on File and Storage Technologies (FAST '07) brings together storage system researchers and practitioners to explore new directions in the design, implementation, evaluation, and deployment of storage systems. The conference will consist of two and a half days of technical presentations, including refereed papers, Work-in-Progress reports, and a poster session.

Topics

Topics of interest include but are not limited to:

- Archival storage systems
- Caching, replication, and consistency
- Database storage issues
- Distributed I/O (wide-area, grid, peer-to-peer)
- Empirical evaluation of storage systems
- Experience with deployed systems
- Mobile storage technology
- Parallel I/O
- Performance
- Manageability
- Reliability, availability, disaster tolerance
- Security
- Scalability
- Storage networking
- Virtualization

Deadline and Submission Instructions

Submissions will be done electronically via a Web form, which will be available on the FAST '07 Call for Papers Web site, http://www.usenix.org/fast07/cfp. The Web form asks for contact information for the paper and allows for the submission of your full paper file in PDF format.

Submissions must be full papers (no extended abstracts) and must be no longer than thirteen (13) pages plus as many additional pages as are needed for references (e.g., your paper can be 16 total pages, as long as the last three or more are the bibliography). Your paper should be typeset in two-column format in 10 point type on 12 point (single-spaced) leading, with the text block being no more than 6.5" wide by 9" deep.

Authors must not be identified in the submissions, either explicitly or by implication (e.g., through the references or acknowledgments). Blind reviewing of full papers will be done by the program committee, assisted by outside referees. Conditionally accepted papers will be shepherded through an editorial review process by a member of the program committee.

Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, program committees may share information about submitted papers with other conference chairs and journal editors to ensure the integrity of papers under consideration. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case.

Authors uncertain whether their submission meets USENIX's guidelines should contact the program chairs, fast07chairs@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

Accepted material may not be subsequently published in other conferences or journals for one year from the date of acceptance by USENIX. Papers accompanied by nondisclosure agreement forms will not be read or reviewed. All submissions will be held in confidence prior to publication of the technical program, both as a matter of policy and in accordance with the U.S. Copyright Act of 1976.

Submissions violating these rules or the formatting guidelines will not be considered for publication.

One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration upon request.

Best Paper Awards

Awards will be given for the best paper(s) at the conference.

Work-in-Progress Reports and Poster Session

The FAST technical sessions will include slots for Work-in-Progress reports, preliminary results, "outrageous" opinion statements, and a poster session. We are particularly interested in presentations of student work. Please see the Call for Papers Web site, http://www.usenix.org/fast07/cfp, for details.

Birds-of-a-Feather Sessions

Birds-of-a-Feather sessions (BoFs) are informal gatherings organized by attendees interested in a particular topic. BoFs will be held in the evening. BoFs may be scheduled in advance by emailing the Conference Department at bofs@usenix.org. BoFs may also be scheduled at the conference.

Registration Materials

Complete program and registration information will be available in November 2006 on the conference Web site. The information will be in both HTML and a printable PDF file. If you would like to receive the latest USENIX conference information, please join our mailing list: http://www.usenix.org/about/mailing.html.

writing for ;login:

Writing is not easy for most of us. Having your writing rejected, for any reason, is no fun at all. The way to get your articles published in *;login:*, with the least effort on your part and on the part of the staff of ;login:, is to submit a proposal first.

PROPOSALS

In the world of publishing, writing a proposal is nothing new. If you plan on writing a book, you need to write one chapter, a proposed table of contents, and the proposal itself and send the package to a book publisher. Writing the entire book first is asking for rejection, unless you are a well-known, popular writer.

;login: proposals are not like paper submission abstracts. We are not asking you to write a draft of the article as the proposal, but instead to describe the article you wish to write. There are some elements that you will want to include in any proposal:

- What's the topic of the article?
- What type of article is it (case study, tutorial, editorial, mini-paper, etc.)?
- Who is the intended audience (syadmins, programmers, security wonks, network admins, etc.)?
- Why does this article need to be read?
- What, if any, non-text elements (illustrations, code, diagrams, etc.) will be included?
- What is the approximate length of the article?

Start out by answering each of those six questions. In answering the question about length, bear in mind that a page in *;login:* is about 600 words. It is unusual for us to publish a one-page article or one over eight pages in length, but it can happen, and it will, if your article deserves it. We suggest, however, that you try to keep your article between two and five pages, as this matches the attention span of many people.

The answer to the question about why the article needs to be read is the place to wax enthusiastic. We do not want marketing, but your most eloquent explanation of why this article is important to the readership of *;login:*, which is also the membership of USENIX.

UNACCEPTABLE ARTICLES

;login: will not publish certain articles. These include but are not limited to:

- Previously published articles. A piece that has appeared on your own Web server but not been posted to USENET or slashdot is not considered to have been published.
- Marketing pieces of any type. We don't accept articles about products. "Marketing" does not include being enthusiastic about a new tool or software that you can download for free, and you are encouraged to write case studies of hardware or software that you helped install and configure, as long as you are not affiliated with or paid by the company you are writing about.
- Personal attacks

FORMAT

The initial reading of your article will be done by people using UNIX systems. Later phases involve Macs, but please send us text/plain formatted documents for the proposal. Send proposals to login@usenix.org.

DEADLINES

For our publishing deadlines, including the time you can expect to be asked to read proofs of your article, see the online schedule at http://www.usenix.org/publications/login/sched.html.

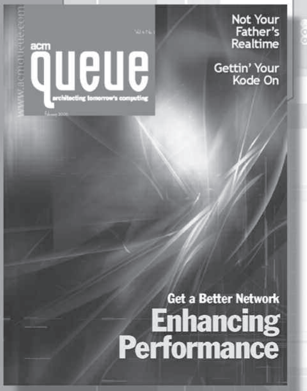
COPYRIGHT

You own the copyright to your work and grant USENIX permission to publish it in ;login: and on the Web. USENIX owns the copyright on the collection that is each issue of *;login:*. You have control over who may reprint your text; financial negotiations are a private matter between you and any reprinter.

FOCUS ISSUES

In the past, there has been only one focus issue per year, the December Security edition. In the future, each issue may have one or more suggested focuses, tied either to events that will happen soon after *;login:* has been delivered or events that are summarized in that edition.

there's a whole lot of technology in the queue. are you ready?
what's next?
acm


acm
queue
architecting tomorrow's computing
Not Your Father's Realtime
Gettin' Your Kode On
Get a Better Network
Enhancing Performance

;login:

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES
RIDE ALONG ENCLOSED