FEBRUARY 2009 VOLUME 34 NUMBER 1

;login:

THE USENIX MAGAZINE

OPINION

Musings — RIK FARROW — 2

SYSADMIN

Package Managers Still Vulnerable: How to Protect Your Systems — JUSTIN SAMUEL AND JUSTIN CAPPOS — 7

STORM: Simple Tools for Resource Management — MARK DEHUS AND DIRK GRUNWALD — 16

Programming the Virtual Infrastructure — PAUL ANDERSON — 20

Python: An Untapped Resource in System Administration — COREY BRUNE — 26

COLUMNS

Practical Perl Tools: Be the Browser — DAVID N. BLANK-EDELMAN — 30

Pete's All Things Sun (PATS): Analytics for the Masses (of Storage Administrators) — PETER BAER GALVIN — 38

iVoyeur: Message in a Bottle—Replacing Email Warnings with SMS — DAVE JOSEPHSEN — 45

/dev/random — ROBERT G. FERRELL — 49

BOOK REVIEWS

Book Reviews — ELIZABETH ZWICKY ET AL. — 51

USENIX NOTES

The International Olympiad on Informatics — ROB KOLSTAD — 56

Got a Minute? — JANE-ELLEN LONG — 57

USENIX Discussion List — ANNE DICKISON — 58

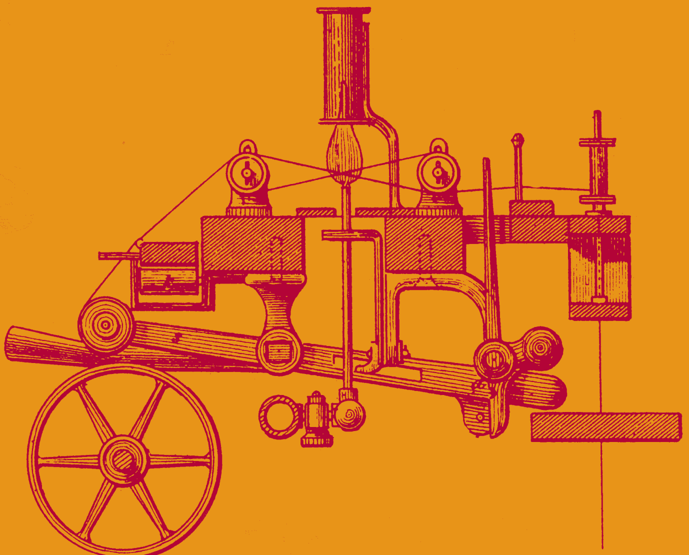
New Videos Available! — ANNE DICKISON — 58

CONFERENCES

LISA '08 Reports — 59

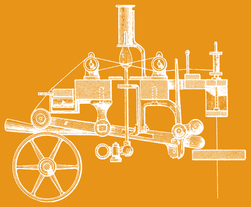
CHIMIT '08 — 82

Report on the 8th Privacy Enhancing Technologies Symposium — 86



USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



1st Workshop on the Theory and Practice of Provenance (TaPP '09)

Co-located with FAST '09

FEBRUARY 23, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/tapp09

7th USENIX Conference on File and Storage Technologies (FAST '09)

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

FEBRUARY 24–27, 2009, SAN FRANCISCO, CA, USA
http://www.usenix.org/fast09

2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '09)

Sponsored by ACM SIGPLAN and SIGOPS in cooperation with USENIX

MARCH 11–13, 2009, WASHINGTON, D.C., USA
http://www.cs.purdue.edu/VEE09/

First USENIX Workshop on Hot Topics in Parallelism (HotPar '09)

MARCH 30–31, 2009, BERKELEY, CA
http://www.usenix.org/hotpar09

8th International Workshop on Peer-to-Peer Systems (IPTPS '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/iptps09

2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '09)

Co-located with NSDI '09

APRIL 21, 2009, BOSTON, MA, USA
http://www.usenix.org/leet09

6th USENIX Symposium on Networked Systems Design and Implementation (NSDI '09)

Sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

APRIL 22–24, 2009, BOSTON, MA, USA
http://www.usenix.org/nsdi09

12th Workshop on Hot Topics in Operating Systems (HotOS XII)

Sponsored by USENIX in cooperation with the IEEE Technical Committee on Operating Systems (TCOS)S

MAY 18–20, 2009, MONTE VERITÀ, SWITZERLAND
http://www.usenix.org/hotos09

2009 USENIX Annual Technical Conference

JUNE 14–19, 2009, SAN DIEGO, CA, USA
http://www.usenix.org/usenix09

18th USENIX Security Symposium

AUGUST 12–14, 2009, MONTREAL, CANADA
http://www.usenix.org/sec09

22nd ACM Symposium on Operating Systems Principles (SOSP '09)

Sponsored by ACM SIGOPS in cooperation with USENIX

OCTOBER 11–14, 2009, BIG SKY, MT, USA
http://www.sigops.org/sosp/sosp09/

23rd Large Installation System Administration Conference (LISA '09)

Sponsored by USENIX and SAGE

NOVEMBER 1–6, 2009, BALTIMORE, MD, USA
http://www.usenix.org/lisa09
Submissions due: April 30, 2009

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

contents

OPINION

Musings — 2
RIK FARROW

SYSADMIN

Package Managers Still Vulnerable: How to Protect Your Systems — 7
JUSTIN SAMUEL AND JUSTIN CAPPOS

STORM: Simple Tools for Resource Management — 16
MARK DEHUS AND DIRK GRUNWALD

Programming the Virtual Infrastructure — 20
PAUL ANDERSON

Python: An Untapped Resource in System Administration — 26
COREY BRUNE

COLUMNS

Practical Perl Tools: Be the Browser — 30
DAVID N. BLANK-EDELMAN

Pete's All Things Sun (PATS): Analytics for the Masses (of Storage Administrators) — 38
PETER BAER GALVIN

iVoyeur: Message in a Bottle—Replacing Email Warnings with SMS — 45
DAVE JOSEPHSEN

/dev/random — 49
ROBERT G. FERRELL

BOOK REVIEWS

Book Reviews — 51
ELIZABETH ZWICKY ET AL.

USENIX NOTES

The International Olympiad on Informatics — 56
ROB KOLSTAD

Got a Minute? — 57
JANE-ELLEN LONG

USENIX Discussion List — 58
ANNE DICKISON

New Videos Available! — 58
ANNE DICKISON

CONFERENCES

LISA '08 Reports — 59

CHIMIT '08 — 82

Report on the 8th Privacy Enhancing Technologies Symposium — 86

;login:

VOL. 34, #1, FEBRUARY 2009

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Casey Henderson
Jane-Ellen Long
Michele Nelson
Jennifer Peterson

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.

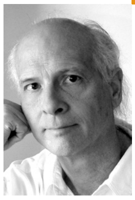
$90 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $120 per year.

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@usenix.org

LISA '08 IN SAN DIEGO SEEMS TO ME like a place far away, in both time and space. But the memories are still fresh in my mind as I write this: sitting outside eating, drinking, and talking, listening to talks and papers, and meeting people I haven't seen in a while in the halls. In this column, I am going to pick up on a couple of subthemes from the conference: documentation and system configuration.

You can read about the invited talk on writing documentation in the reports on LISA in this issue. As I read them, they brought to mind a couple of things from my own deep, dark past. Yes, I once wrote documentation *and liked doing it*. Somewhere along the way I lost that feeling, and I am still wondering why.

Hardware

When I started consulting, I fell into writing hardware documentation. I had pestered George Morrow [1] and his wife about getting work, and one day I got a call asking me if I could revise a disk controller manual. They had taken an existing floppy-disk controller card and reworked it so that it was I/O-mapped instead of memory-mapped. Rewriting the manual meant translating all memory references to I/O references, starting with the old manual as a WordStar file.

I could read circuit diagrams and had actually patched CP/M device drivers, even relocated one so that it worked at different memory addresses (by using a FORTH program), so this sounded easy. And it was. I whipped through the job, turned it in (on an 8-inch floppy disk), and got paid on the spot. This project led to many other writing projects for Bay Area companies in the early 1980s.

I was a carefree sort of guy back then and spent three months sailing to Panama in 1984. When I went back to Morrow Designs, I discovered that the next hardware manual had been given to a professional documentation writer. I was disappointed, but the manager explained that it would cost them a lot less.

Right on cue, the documentation writer shows up, and she tells everyone that she has written the first 10 pages of documentation. Then she asks the engineers sitting nearby how to install this hard disk controller she needed to document in her single-board computer system. This was an impossibility, as there were no slots for add-ons in her system. The manager asked me to leave the two of them alone for a few minutes. Shortly after, I learned I had gotten the job.

The disk controller was very cool: An early RISC processor handled both DMA and disk reads and writes. Timing was critical, as early disk controllers actually read and wrote analog data from the drives. The consultant who had designed the board had created his own language for programming the RISC chip, and the 30-some pages of printout were my main source for writing the manual.

The controller emulated IBM mainframe channel controllers, in that the device driver programmer could create a linked list of commands in memory, then tell the controller to carry out the list of commands, notifying the CPU with interrupts when desired. Documenting the short list of commands only took perhaps 30 pages. Later, a device driver programmer told me that writing the device driver from my documentation was easier than any other driver he had ever written. The slick design was certainly mostly responsible. But so was having accurate documentation.

I later learned that the professional writer had complained that she was rejected because a man had shown up. However, that wasn't the real issue: Her talent was in writing, not in reading circuit diagrams and homebrew source code for rare RISC processors. I could do that as well as write.

And that, I believe, speaks to the core issue in documentation. People who write most documentation are writers, and not generally technically apt. Real engineers, for the most part, hate writing documentation. Once a project, be it hardware or software, is working, it is no longer interesting. So the documentation gets written by someone with at best a weak technical background and grudging access to the engineers. And you get documentation written by someone who has little clue.

Today, buying software without any documentation is common, and hardware documentation rarely goes beyond badly translated explanations of how to connect cables and perhaps install firmware (but only when using Windows). A huge after-market in books that provide documentation (consider O'Reilly's missing manual series) has developed, where an outsider reverse-engineers a product and writes about it in a book that will be outdated within a year.

Perhaps there are examples of great documentation out there, for hardware or software. I fondly recall unpacking Sun 3/60s, finding a short length of cable and a BNC T connector, and wondering what the hell it was for (Ethernet, dummy!). Like all "real men," and probably many women as well, I had to figure it out for myself. Well, who bothers to read the manual?

Sysadmins and Documentation

When I began writing my system administration book, I had already written several UNIX system manuals. One thing I had learned to do was take good notes, and I preached that in my first chapter of my book.

Today, I take notes inconsistently. And I regret this whenever I am faced with solving the same problem I solved months earlier but didn't take good notes about the process. My usual excuse was just like the engineers': I am too busy to waste time taking notes. Later I pay for not "wasting" the time, by wasting more time. Puzzle-solving is fun, but not so fun when the time pressure is great.

Janice Gelb, in her “WTFM: Documentation and the System Administrator” invited talk, covers this angle and much more, so read the report—the slides are online and this talk is available in streaming video from LinuxPro[2]. And start writing good documentation. If you cannot write, work with someone else who can write by patiently answering their questions. Provide examples with comments. If your work is poorly documented, you might be the only one who ever uses it, no matter how brilliantly designed it is.

Configuration Management

The configuration management tools war appeared to have cooled down. Either there weren’t loud arguments for a particular tool during workshops, or I missed them. Yet the architects of the four most popular tools—Cfengine, Bcfg2, LCFG, and Puppet—were all present, and I got to talk with all of them in the hallway or over meals.

There were two invited talks, one by Paul Anderson, architect of LCFG, who wanted to look at configuration management from a different perspective, and has written about it in this issue (page 20). Then there was an invited talk from Ticketmaster about its own tool, Spine. Like many before them, the Ticketmaster sysadmins decided that nothing in existence really worked for the company’s model, so they built their own toolset.

I got another data point about the state of these tools when architects from two of the big four approached Jordan Hubbard, Director of UNIX Development at Apple (and well-known FreeBSD developer before that) after Jordan’s talk. Each claimed that his tool was a popular way of managing Apple products, and each time Jordan suggested that they create a document describing the configuration knobs they wanted from Apple to expose via APIs.

Jordan’s comment got me thinking. His goal was to publish an API that would isolate the internal details from configuration management tools. I imagined that this would mean that Apple engineers could document this API and still be free to mangle the file formats used for configuration however they liked. But I also wondered what this API would need to cover to handle the most common configuration management tasks.

Currently, all tools edit configuration files, along with other tasks such as restarting or reloading services. If there was a common API for most operating systems, this would make system management easier. However, current tools present a one-tool-does-everything approach to configuration management. In my mind, I can break down the tool into a set of components: a secure communication protocol, a client-side, and a server-side. The server-side seems like the real place to innovate, with components for storing current and past configurations, a GUI to make using the tool simple, more GUI tools for monitoring the effectiveness of the tool, and, of course, CLI versions.

So even though there are many configuration management tools in existence today, there still seems room for research on any of these aspects. I can imagine sysadmins being able to mix-and-match their favorite tools, based on desired features, something that I can only dream about today because of the monolithic architectures.

The Lineup

I started the lineup with an article about insecurities in Linux package management. Cappos and Samuels have written for *;login:* before (February

2008), and when they got slash-dotted this summer, I asked if they would write for us again.

The issue this time has to do with how different Linux distros secure their package management systems. The right way to do it is to have a signed manifest from a trusted source and to use that manifest to verify packages that can be downloaded using mirrors. But this is not how most distros do this, and the consequences are scary indeed.

Dehus and Grunwald cover a different angle of package management. Based on their LISA '08 paper, they describe the system they developed for instancing virtual servers. A key feature of their project, STORM, is that packages that are common to many services get shared, so that these packages can be updated in just one location.

Paul Anderson expounds on the topic of his invited talk at LISA in the next article. Paul ponders about the parallels between the adoption of programming languages and the adoption of configuration management tools and languages. He wonders whether that aren't lessons to be learned there.

Corey Brune decided to evangelize Python for system administration. He wrote a script for extracting password records, converting them to LDAP data, and adding them to an LDAP database. Brune uses this to explain features of Python, a scripting language that is showing up in more places all the time.

David Blank-Edelman shows how to screen-scrape Web pages using WWW::Mechanize. As usual, David provides us with clear examples for using this module, including methods for automating form-filling and submission.

Peter Galvin waxes enthusiastic over a DTrace-based feature found in new Sun storage appliances. Although these are truly appliances, with no visible traces of either DTrace or Solaris, Peter both explains and demonstrates the cool ways of viewing current and past performance of a storage appliance that uses DTrace under the hood.

Dave Josephsen bemoans the death of email, then quickly moves on to explain the use of SMS or Asterisk as a replacement for remote notification systems. I'd heard about texting used in this way before, and it's cool to have a concrete set of examples using Nagios and Nokia phones.

Robert Ferrell goes where no sane sysadmin has gone, and that is into the thicket that is the vi versus emacs debate. Well, sort of.

In the book reviews, Elizabeth Zwicky leads off with three graphic introductions to statistics (no kidding). I add a nongraphic suggestion later, for those of you who don't care whether your statistics textbook is funny. Zwicky then writes about Peter Salus's new book on open source software, then on a book concerning the use of diagrams for explanations and as a way of winning arguments. Finally, she compares two books on the use of Photoshop CS4 for photographers.

Jason Dusek is up next, beginning with a review of *Maven: The Definitive Guide*. Maven is a tool for code build management, and Jason clearly describes how this book handles the uneven documentation surrounding the open-source project. Then Jason takes a deeper dive into *Design Concepts in Programming Languages*, a book about programming language theory that appears most useful to those designing languages, whether big or small.

Sam Stover reviews *The Craft of System Security* and explains why he likes it. I have two short reviews, one on *Statistics in a Nutshell* and the other on *Getting Started with Arduino*.

Although I never set out to be a writer, and even hated writing right through college, I eventually became not just a writer but a successful one. Along the way, I've noticed just how important it is, in the businesses of system administration, consulting, and publishing, to be able to write skillfully. If I had really understood the importance of English 101, I would have looked at it very differently, as I do now.

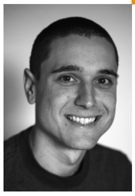
REFERENCES

[1] George Morrow: http://en.wikipedia.org/wiki/George_Morrow_(computers).

[2] LISA08, Thursday Tech lineup: http://www.usenix.org/events/lisa08/tech/#thursday.

JUSTIN SAMUEL AND JUSTIN CAPPOS

package managers still vulnerable: how to protect your systems



Justin Samuel is an undergraduate student at the University of Arizona. He is the author of the RequestPolicy Firefox extension, a tool for increasing the privacy and security of browsing.

jsamuel@cs.arizona.edu

Justin Cappos is a Post Doc at the University of Washington. His research interests revolve around building large distributed systems and gaining experience from their practical use.

justinc@cs.washington.edu

SERIOUS VULNERABILITIES HAVE BEEN discovered in all Linux package managers. Although most major package managers and distributions have begun addressing these issues, some of the most popular are not fully protecting their users. We'll look at known vulnerabilities in package managers, what is being done to fix them, and how you can protect your systems even if you're using a vulnerable package manager.

By package managers, we're talking about the tools we all use to update the software on our systems, such as APT, YUM, and YaST. Even if you don't invoke your package manager directly but instead do so through a graphical interface or a scheduled job, the same vulnerabilities exist.

The vulnerabilities we discovered require that an attacker be able to respond to a client when the package manager is downloading files. Unfortunately, we also found that it is extremely easy, in many cases, for attackers to position themselves to do this. By becoming a public mirror for a distribution's repositories, attackers can often target as many clients as they have bandwidth to support [1].

Here we'll look at the most common package managers and the distributions using them, which are, generally:

APT:	used by Debian and Ubuntu
APT-RPM:	used by ALT Linux and PCLinuxOS
Pacman:	used by Arch Linux
Portage:	used by Gentoo
Slaktool:	used by Slackware
Stork:	used for research on PlanetLab
URPMI:	used by Mandriva
YaST:	used by openSUSE and SUSE Linux Enterprise
YUM:	used by Fedora, Red Hat Enterprise, and CentOS

In this article we will focus on Linux package management systems. With BSD systems, a common approach for updates is the distribution of the Ports collection [2]. The Ports collection is often distributed insecurely, although the portsnap tool for FreeBSD, which we are currently investigating, may be a secure alternative [3].

How Package Managers Work

Package managers do the job of installing new or updated software on our systems. To be able to do this, they run as root. Thus, what they do and the software they install affect the security of the entire system. If a package manager can be tricked by an attacker into installing a malicious package the attacker has created (or even just an old package that has known vulnerabilities), the attacker can compromise the system. It's therefore critical that package managers be secure.

To be secure, they must install the software we intended them to install. If we didn't want a package installed (e.g., a vulnerable or malicious package), it shouldn't be installed. If we did want a package installed (e.g., an update that fixes a security flaw), that package should be installed. Also, of course, a malicious party shouldn't be able to cause our package manager to crash our system while it is running.

For all package managers, the basic process they follow is similar. First, they download information about available packages from a remote repository. They then use this information to decide which package to install. The repository is usually an HTTP or FTP server. The information downloaded (called *metadata*) gives details about the packages that are available on the repository. These details include information about the versions of each available package, any additional packages they require in order to work properly, what functionality they provide, their cryptographic hashes, their file sizes, and so on.

In practice, a common approach employed by most package managers is to start off each time they run by downloading a single file from the repository. This file, the *root metadata*, is a small file that describes the contents and layout of the repository, usually giving the names and hashes of other files on the repository that actually contain the detailed information the package manager needs. The package manager determines whether the files listed in the root metadata are different from the ones it last downloaded; if so, it downloads these new files from the repository and verifies that what it received has the same hashes as were listed in the root metadata.

After downloading all of the metadata it needs, the package manager uses the information to decide whether there is anything that should be installed. If the package manager was being run for the purpose of updating the system, it looks at all of the packages currently installed and checks whether the metadata describes newer versions (higher version numbers) of those same packages. If the package manager is being used to install a new package on the system, it looks for the highest version number of that package listed in the metadata.

When deciding whether a package can be installed, the package manager makes sure that either the other software a package requires is already installed or, if not, that this software can also be obtained from the repository (that is, that it is available according to the metadata). This process of *dependency resolution* continues until there are no more dependencies to resolve, there is some form of conflict, or some dependencies cannot be resolved. The package manager only continues if it can resolve all dependencies.

Next, the package manager downloads the individual packages it wants to install and then installs them. At this point, if your package manager has been tricked by an attacker, you're in big trouble.

Traditional Security in Package Managers

Some package managers don't make any pretense of being secure in the first place (such as Slaktool on Slackware and Pacman on Arch Linux). We strongly recommend against using package managers that don't try to be secure.

Of course, most popular package managers do, in fact, try to perform the update and installation process described here securely. Until recently, most of them considered themselves completely secure.

As you may know or have already guessed, their security mechanisms are based on the use of cryptographic signatures. The security differences in current package managers largely come down to what is actually signed. The options are either signatures on the root metadata, omn the packages themselves, or, in some cases, on metadata that describes the packages.

Signatures on the root metadata mean that the first file the package management client downloads has a signature the client can check. By checking the root metadata file's signature and then verifying that the secure hashes of each file downloaded thereafter match the expected hashes, the signature's authority extends to the ultimately downloaded packages. The package managers that use this model include APT, APT-RPM, and YaST.

Another common place for the signature is on each individual package. In this model, the package manager has no signatures to check until it gets to the point where it downloads the actual packages it intends to install. It then checks the package signatures before installation. Package managers that use signatures on individual packages include YUM and URPMI.

Finally, signatures can be placed on files that directly contain the metadata of the packages. This approach is used by Portage and Stork, although they accomplish this in somewhat different ways [1].

The Vulnerabilities

The vulnerabilities that were discovered fall into three main categories: replay/freeze attacks, metadata manipulation attacks, and denial-of-service (DoS) attacks.

REPLAY AND FREEZE ATTACKS

The replay attacks that package managers are vulnerable to are the same replay attacks that cryptanalysts and security-minded people have long been aware of. The attack in this case involves a malicious party responding to a package manager's request for signed metadata (the information about packages available on a repository) with an old signed file. The attacker does not need to compromise the signing key to do this.

The problem basically comes down to the fact that, with the way package managers currently work, once a file is signed and thus trusted by clients, it is always trusted, even after vulnerabilities are discovered in packages that were once considered safe. This, of course, will always happen (which is a large part of why we use package managers: so we can get updates to patch our systems once vulnerabilities are discovered).

So, a replay attack allows an attacker to respond to a client's request for repository metadata with old metadata that lists packages the attacker knows how to exploit. This could even be metadata much older than that which the client has already seen. Unfortunately, with the way all package managers have been written, clients aren't bothered if yesterday they retrieved metadata that is dated from last week but today they retrieved metadata that is dated from a year ago.

A freeze attack is similar to a replay attack. In fact, from a cryptanalytic point of view, it's actually the same thing. However, it's worth giving it a different name to ensure it gets the attention it deserves. This is because solving all facets of the replay attack problem isn't as simple as making sure that clients never accept metadata that is older than metadata they have already seen. As an attacker can keep giving the client a single version of the metadata starting at one point in time (that is, "freezing" the metadata), the attacker can prevent the client from knowing about new metadata and thus new packages that are available that fix known vulnerabilities.

Therefore, securing package managers against replay and freeze attacks must involve limiting how long any signed metadata is considered valid. The problem is that no package managers currently do this (though a few are now working on this, as we'll see in a moment).

METADATA MANIPULATION ATTACKS

The replay attacks discussed so far are useful for attackers who want to compromise systems whose package managers use signed metadata. However, if a package manager does not used signed metadata (such as with YUM or URPMI), attacks don't have to bother with replaying old metadata. Instead, attackers can just make up their own metadata!

What's the point of attackers making up their own metadata that they serve to clients? There are two main things attackers can do. First, they can mix-and-match the versions of packages that are listed. Second, they can trick clients into thinking that packages have different dependencies and provide different functionality than they really do.

In mixing-and-matching vulnerable package versions by listing them in the same metadata given to a client, attackers make it more likely that, whatever new package a client installs, it is installing a version with a known vulnerability. At least with replay attacks, the attacker has to choose a certain snapshot of historical packages to show the client. If two packages were vulnerable at different times, the attacker can't use a replay attack to make both vulnerable versions available to a client. However, when a package manager doesn't sign metadata, the attacker isn't limited to a single point in time.

As mentioned, the lack of signed metadata allows attackers to lie to clients about what each package provides and requires. By lying to clients about this information, an attacker can significantly increase the chances of a client installing a vulnerable package. For example, if package *foo* has a vulnerability the attacker knows how to exploit, the attacker can provide metadata that says every package depends on package *foo,* in order to ensure that the client installs it when installing any other package.

There are other bad things an attacker can do when package managers don't use signed metadata. The point, though, is that if your package manager does not sign metadata, your systems are at risk. The solution here is for clients to require signed metadata. A package manager should at least sign the root metadata. Depending on the design of the package manager, it may also use signatures on package metadata.

ENDLESS DATA ATTACKS

The last of the categories of attacks on package managers that were made known comprises forms of DoS attacks. These are not intellectually challenging concepts, but they have been universally overlooked and their ramifications are quite serious.

The attack involves a malicious party responding to a client request, be it for metadata or for a package, with an endless stream of data. Depending on the package manager and the specifics of the stream of data (e.g., whether the endless data began after http headers were ended or before), the effects can vary even within the same package manager. The possible effects include filling up the partition where the package manager saves downloaded files or exhausting memory.

Obviously, in either case, bad things can happen. If the file system where the package manager is storing downloaded files is somewhere that other important data is stored (it's usually under "/var/"), that other important data can be corrupted. Likely candidates for corrupted data are databases, mail, and log files. If memory is exhausted, the system slows to a crawl (an attacker could, for example, slow down the stream of endless data before memory is completely used and the OS kills the package manager's process).

Mirrors: The Easy Way to Exploit Clients

How does an attacker go about exploiting these vulnerabilities? Fundamentally, the attacker just needs to be able to respond to the client's requests for metadata and packages. This ends up being easier than it sounds.

For most noncommercial Linux distributions, the repositories from which package managers download files are not actually run by the distributions themselves. Instead, a wide variety of volunteer individuals, companies, and organizations donate a portion of their own server space and bandwidth by acting as repository mirrors. These mirrors of the main repository do nothing other than provide an exact copy of the distribution's main repository. Or, more accurately, that's what they are supposed to do.

It turns out, though, that it is easy for anyone, attackers included, to become official mirrors for most major distributions. We tested how easy this was with five of the most popular distributions (Debian, Ubuntu, Fedora, CentOS, and openSUSE). The most that was required for any of them was sending an email announcing the availability of our mirror. In some cases, the process was completely automated and mirror registration was done through a Web site. In the short time we had our mirrors available, we served metadata and packages to thousands of systems, including government and military computers [1].

Once an attacker runs a mirror, it can reply to client requests with malicious content. Most distributions claim to regularly monitor the content of the mirrors to ensure that they are updated and accurate, but this only serves the purpose of ensuring that well-intentioned repositories haven't fallen out of date. It is trivial for malicious mirrors to lie to the monitoring service while still attacking actual clients. (We, of course, did not serve malicious content to clients from our mirrors. The vulnerability testing we did was done privately using our own systems.)

Not every distribution uses public mirrors, though. The commercial Linux distributions SUSE Linux Enterprise and Red Hat Enterprise Linux don't use public mirrors, so these distributions are not at risk from malicious mirrors. They both also use SSL when clients talk to repositories, which protects them from man-in-the-middle (MITM) attacks. (During our research, we discovered a bug in Red Hat Enterprise Linux's use of SSL that still allowed MITM attacks, but Red Hat corrected this very quickly.)

One noncommercial distribution, openSUSE, stands out for using an approach of serving all metadata from its own servers and only allowing packages to be served from mirrors. This method decreases the risk to users from malicious mirrors, although it doesn't make users completely safe. Why aren't they completely safe? The reason is that even though malicious mirrors are often the easiest way to exploit package managers, there is still the general risk from a MITM attack.

As a MITM, the attacker can control the responses the clients get unless the client talks to the repository using SSL. Very few package managers and distributions support SSL for talking to the repository (currently, only the commercial distributions use SSL). The potential for these MITM attacks comes from a variety of sources, including everything from an attacker on the wire to DNS cache poisoning [4] and BGP prefix hijacking [5].

The fundamental need is to design package managers such that they cannot be exploited by a malicious mirror or a MITM. A package manager needs to be certain that the metadata it receives is accurate and recent, as well as needing to be safe against DoS attacks during the data transfer itself.

Who Was Vulnerable

The more popular package managers and distributions are working on fixing these vulnerabilities. Before looking at the current state of affairs, let's see how these issues impacted the different package managers and distributions. There is some overlap in security mechanisms and vulnerabilities for the various package managers, but we'll try to simplify here.

The first thing to note is that package managers that didn't use signatures on root metadata were all vulnerable, in varying degrees, to metadata manipulation attacks. This included YUM, Portage, and Stork. The "lower down" the signature was placed (closer to the package), the worse the attacks could be. The worst was the case where only packages themselves were signed, as has been the case with YUM. When only packages were signed, all of the metadata manipulation attacks described here were possible.

With replay and freeze attacks, all package managers were vulnerable. However, the package managers that already signed either the root metadata or package metadata required fewer changes to become secure than those that only signed packages. As we'll see in a moment, this has resulted in package managers such as YaST and APT having been more quickly able to focus on protecting against replay and freeze attacks.

The endless data attacks also affected all package managers. These are arguably just implementation flaws, although they are rooted in the more fundamental issue of the client having too much trust in the party it is talking to.

Some smaller distributions have security features available through their package manager but do not use those features. APT-RPM, for example, supports root metadata signatures but PCLinuxOS doesn't use them.

Then there were some other odds and ends. One notable discovery was that Fedora allows users to register mirrors through Fedora's Web site and specify them as responsible for arbitrary CIDR blocks. Attackers can use this to target specific IP address ranges by telling clients in those ranges to communicate with the attacker's mirror. Unfortunately, it appears that Fedora plans to keep this system despite the risks it poses. To understand how serious this is, consider that an attacker could register its mirror as responsible for a block of government or military IP addresses and then attack those computers [6].

Which Package Managers Are Being Secured

The more popular package managers have begun planning and implementing solutions to these vulnerabilities, but currently only a limited amount has been done.

The most vulnerable of the popular package managers, YUM, has added the ability to use signed root metadata, thus securing it from the various metadata manipulation attacks [7]. YUM developers have stated that they plan to warn users if metadata the client receives is too old (that is, a possible replay attack), but this has not yet been implemented. YUM also plans to correct its SSL implementation so that server certificate validity is checked (a similar problem to what RHEL had and fixed) [8].

The ability to sign root metadata will make it into Fedora 10, but it is uncertain whether Fedora 10 will begin making use of this feature. Even more uncertain is when RHEL clone distros such as CentOS might begin using root metadata signatures. The Fedora Project also has stated that it intends to have YUM's initial requests for lists of mirrors be done through SSL. In general, Fedora and YUM developers have expressed their intent to address most issues besides endless data attacks [7].

Debian's APT developers have begun planning the necessary changes to protect APT users from replay attacks. Some of this has made it into Debian's testing branch, but it is not usable yet [9]. Like YUM, APT has also not yet addressed endless data attacks. Ubuntu, being derived from Debian, may follow its lead, although Ubuntu does have its own open bug report for the replay and endless data attacks [10].

With the release of openSUSE 11.1, the openSUSE developers say replay and freeze attacks will be protected against in YaST. They also have said that they are working on protecting against endless data attacks but those changes won't be ready until the following release [11].

Gentoo's Portage developers have begun to address its vulnerabilities. They are in the planning stage for adding a signed root metadata file to Portage, as well as using it to protect against replay and freeze attacks [12]. Additionally, they have already written a patch to protect against endless data attacks.

We implemented protections against all of the attacks mentioned for Stork, our research package management system [13]. We have shared our experiences implementing these mechanisms with other package manager developers to help them implement protections in their package managers.

How to Protect Your Systems

In the long term, the best way you can stay secure against the attacks we've discussed is to choose a distribution that has devoted the necessary time and energy to secure its users against these attacks.

Our findings show that there really is a security advantage to using one of the enterprise distributions, either SUSE Linux Enterprise or Red Hat Enterprise Linux. They fared so well not because they had specifically protected against any of these attacks but, rather, because of their use of SSL for communication and not exposing clients to public mirrors. Among the free distributions, openSUSE should soon offer the same level of protection against these attacks as the enterprise distributions.

In general, the most significant criterion with regard to the vulnerabilities discussed is whether root metadata obtained from the repository is signed. Next, assuming the root metadata is signed, it is important to determine whether the package manager is able to recognize when the metadata it has obtained is out of date (because of either a stale mirror or a replay attack) and alert you to this without proceeding automatically.

Worth noting is the fact that it's not just a matter of your package manager having this functionality, but also whether your distribution uses it. For example, distributions using YUM do not yet sign root metadata even though the YUM developers acted quickly in adding support for signed metadata. In such cases, you aren't safe until your distribution uses the security options available with the package manager.

In the meantime, if your package manager or distribution is not safe against replay or metadata manipulation attacks, your only option for a high level of security requires additional manual work on your part. You can stay aware of which packages should be updated or installed on your system, invoke your package manager manually, and ensure that the packages you expected to be installed are in fact the ones installed. Organizations running multiple machines would benefit from running their own internal mirror that syncs from a single upstream mirror. All systems on the organization's network can then use the internal mirror for updates. This then requires that only one machine, the internal mirror, be verified to have accurate and updated content (if your sync method or the mirror you sync from is insecure, your network would again be at risk). Keep in mind that if your organization's internal mirror is accessed by machines over a WAN, your machines will still be vulnerable to MITM attacks.

Endless data attacks are important to protect against, as well. This is especially true for mission-critical systems where uptime is of primary importance. If you are using a package manager that is vulnerable to this attack, there are things you can do to protect your systems until your package manager is secured. Most package managers dump the endless data they receive to a file on the file system. Thus, one way to protect the rest of your system against this type of attack is to mount the directory in which the package manager stores downloaded files as its own file system. However, some package managers allow a memory exhaustion DoS through this attack [14]. In those cases, it is probably best to monitor the running of the package manager either manually or by scripting the monitoring of the processes' memory consumption to kill it if necessary. Using an internal mirror accessed only from within your network (as mentioned above) also mitigates this attack.

If your distribution wasn't mentioned or if you are a BSD user, it is important to look into the security of your package management or software update system. If you are a FreeBSD user, for example, we suggest you look into using portsnap. For other distributions, inquire into the security of the update system you are currently using and other more secure options that may exist. The issues we've discussed in this article should provide you with a solid set of security issues to be aware of.

Conclusion and Resources

Securing these systems is a work in progress for all package managers and distributions. We intend to keep a Web site updated with the progress of the various package managers and distributions [15].

If in doubt about the current state of any of them, please take a look at our Web site and don't hesitate to contact us with your questions.

ACKNOWLEDGMENTS

We would especially like to thank Ryan Giobbi at CERT/CC for helping us to responsibly disclose these issues to the package manager maintainers.

Our wholehearted thanks are owed to the many distribution and package manager developers who have discussed these issues with us and who are working on making their systems more secure. In many cases, their openness has helped further our understanding of these issues and the extent of vulnerability of their package managers.

Specifically, we would like to thank Jake Edge, Dag Wieers, Kees Cook, Jan iankko Lieskovsky, Robin Johnson, Ludwig Nussel, Peter Poeml, Marcus Meissner, Josh Bressers, Tomas Hoger, Jason Gunthorpe, Kyricos Pavlou, Mike Piatek, Chris Gniady, Wenjun Hu, and Tadayoshi Kohno for their comments on our research. We also thank the members of the Stork project for their help in creating Stork and their input and work on this research, especially Scott Baker and John Hartman.

REFERENCES

[1] J. Cappos, J. Samuel, S. Baker, and J. Hartman, "A Look in the Mirror: Attacks on Package Managers," *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS '08)*, October 2008.

[2] "Ports Collection": http://en.wikipedia.org/wiki/Ports_collection.

[3] "Secure FreeBSD Ports Tree Updating": http://www.daemonology.net/portsnap/.

[4] US-CERT, "Vulnerability Note VU#800113": http://www.kb.cert.org/vuls/id/800113.

[5] O. Nordström and C. Dovrolis, "Beware of BGP Attacks," *ACM SIGCOMM Computer Communication Review,* 34(2):1–8 (2004).

[6] J. Bressers, "Re: YUM Security Issues...": http://www.redhat.com/archives/fedora-infrastructure-list/2008-July/msg00140.html.

[7] S. Vidal, "[Yum-devel] 3.2.18 Released": http://lists.baseurl.org/pipermail/yum-devel/2008-August/005350.html.

[8] M. Domsch, "Re: YUM Security Issues...": https://www.redhat.com/archives/fedora-infrastructure-list/2008-July/msg00114.html.

[9] "#499897 Preventing Replay Attacks against the Security Archive": http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=499897.

[10] "Bug #247445 in apt (Ubuntu): 'Package managers vulnerable to replay and endless data attacks' ": https://bugs.launchpad.net/ubuntu/+source/apt/+bug/247445.

[11] "Package Management Security on openSUSE": http://lizards.opensuse.org/2008/07/16/package-management-security-on-opensuse/.

[12] http://viewcvs.gentoo.org/viewcvs.py/gentoo/users/robbat2/tree-signing-gleps/.

[13] http://www.cs.arizona.edu/stork/.

[14] J. Cappos, J. Samuel, S. Baker, and J. Hartman, "Package Management Security," University of Arizona Technical Report 08-02, 2008.

[15] http://www.cs.arizona.edu/people/justin/packagemanagersecurity/.

MARK DEHUS AND DIRK GRUNWALD

STORM: simple tools for resource management

Mark Dehus recently graduated with his M.S. in Computer Science from the University of Colorado at Boulder. His focus is virtualization and large-scale systems administration, and his current research includes virtualization management systems, content delivery/distribution networks, and applications of virtualization toward computer science education.

mark.dehus@colorado.edu

Dirk Grunwald received his Ph.D. from the University of Illinois in 1989 and has been a faculty member at the University of Colorado since that time. He is interested in the design of digital computer systems, including aspects of computer architecture, runtime systems, operating systems, networking, and storage. His current research addresses resource and power control in microprocessor systems, power-efficient wireless networking, and managing very large storage systems.

dirk.grunwald@colorado.edu

CLOUD COMPUTING HAS BECOME ALL the rage, because it allows an organization to forget about the added management associated with having a single operating system tied to a single physical piece of hardware. In this article we discuss a cloud system built using standard tools to further reduce the amount of overhead in computer administration by simplifying software configuration. We also introduce the notion of layered virtual appliances, an approach we consider to be critical for the success of any large-scale cloud management software.

STORM [1] is a system built with the overworked system administrator in mind. It is designed to simplify the configuration and management involved with running applications as much as possible. Using the provided Web interface, an administrator can quickly provision desired software without having to worry about the middle layers such as Apache installation and configuration.

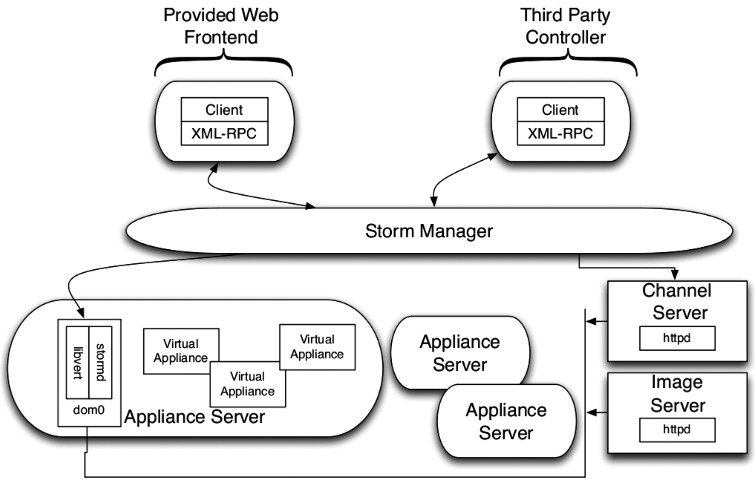


FIGURE 1: THE LOGICAL ARRANGEMENT OF THE FOUR PRIMARY ENTITIES IN STORM

The overall STORM system, illustrated in Figure 1, consists of four primary entities:

- The STORM Manager: The control subsystem for the cloud, which is itself an instance of a virtual appliance. This allows for the management framework to receive the benefits that come along with being an appliance.
- The Virtual Appliance server: The physical hardware running any hypervisor supported by our underlying API, libvirt [2].
- A Channel Server: The server responsible for

serving RSS to the STORM manager. The RSS feed contains information about available appliances.
- An Image Server: The server that strictly serves up disk images. It may be integrated with the channel server if scalability is not required.

The STORM manager, being the central control system for the cloud, is responsible for handling incoming requests from either the administrator via the provided Web front end or from an authorized appliance instance (i.e., a virtual machine). These requests can range from creating new appliance instances to registering MX records.

Several types of requests require the STORM manager to communicate with any given appliance server; this communication is accepted by libvirtd or a custom daemon called stormd, depending on the request. To authenticate the STORM manager, a signed certificate is presented by the manager upon connection. The daemons then verify this signature against a locally stored CA certificate. Libvirt does this authentication internally, and for consistency reasons we reproduced this same scheme for stormd using M2Crypto.

To provide a complete solution, we implemented services such as DNS, DHCP, LDAP, and Kerberos, all of which are handled by the STORM manager and can be controlled through the provided Web interface. Appliances can reach the authentication and authorization mechanisms through the reserved internal DNS record “storm.local” and required configuration items can be obtained through the secure XML-RPC interface.

Traditional and layered appliances are distributed to individual cloud systems by the appliance developers. Each appliance developer should have at least one server available to feed RSS, disk images, and associated XML. For scalability, disk images should be provided from separate servers, or some kind of HTTP load balancer should be used.

The RSS supplied by the channel server is very generic and does not require many attributes to define a channel (see Figure 1). This same principle also applies to the XML associated with an appliance. We do not currently have any mechanism for developers to automatically generate the required RSS and XML. However, given the simplicity of the RSS and XML, it would be trivial to develop a tool to do this.

Layered Appliances

The traditional approach to virtual appliances replicates data by having a single disk image for each given application. When one obtains two traditional appliances that have software in common, this software is stored on the system twice.

For example, take an appliance that provides Wordpress and another that provides MediaWiki. Each requires an underlying substrate of software, such as MySQL, Apache, and PHP. With a traditional appliance the underlying software ends up being stored twice on separate disk images. To address this issue, we invented the concept of a layered appliance.

Layered appliances reduce redundancy by sharing common substrate. This provides several other benefits as well, including the ability to simultaneously apply updates across multiple instances, caching of data across multiple virtual machines, and the capability of taking snapshots of individual layers.

Our virtual appliance approach consists of four layers:

- A common operating system substrate: The substrate contains the basic components needed by all virtual appliances; the Ubuntu “Just Enough OS” (JEOS) platform is a representative example of this.

- An appliance-specific component: This component provides the application and necessary libraries; an example might be the Postfix program, LDAP and MySQL libraries for remote mail delivery, and other necessary libraries.
- A deployment-specific component: This customizes the combination of the operating system substrate and the appliance-specific component; an example might be the configuration files for Postfix, MySQL, NFS, and LDAP. The deployment-specific component essentially captures changes to the underlying appliance component (e.g., the appliance component would typically include off-the-shelf configurations provided by an OS distribution). The deployment-specific component would be the result of an appliance maintainer editing the specific configuration files to customize those files for the local environment.
- An instance-specific component: This uses information provided by the STORM server to configure a specific instance of a more general appliance. For example, that instance-specific information may configure the domain name to be "mail.foo.com" instead of "mail.bar.com."

STORM and the UnionFS

In STORM we used a project called UnionFS [3] to provide the desired functionality for layering. UnionFS allows one to specify a series of directories and have them presented as a single virtual directory. We chose to implement the layering within the base substrate (operating system layer), allowing developers to build nonlayered appliances if so desired.

When a layered virtual appliance is deployed, several disk images are presented as devices and set as writable or read-only devices depending on the layer they provide. The init script within the initrd image mounts each device to a directory and then unions all of the directories appropriately to a single root. It will do its best to detect the write/read-only status of all devices; however, if the detection fails, then it relies on the following logic:

- /dev/sdaX mounts as read-only.
- /dev/sdbX mounts as read-write.
- /dev/sdcX mounts as read-only.
- /dev/sddX mounts as read-write, excluding /dev/sdd4, which is always reserved for swap.

An important thing to note is that UnionFS is not designed for I/O-intensive applications, and it adds significant overhead in these applications. These types of applications already suffer a substantial impact when being virtualized [4, 5]; therefore we suggest considering alternatives for any I/O-intensive application.

So Why Use UnionFS?

An alternative to UnionFS that immediately comes to mind is directly mounting disk images to the proper locations within the filesystem. We decided against doing this for several reasons, but the most important ones were:

- When using directly mounted disk images the developer would have to specify where the disk images should be mounted. The STORM manager would then have to get mapping information (i.e., specifying which disk image is mapped to what device) for every instance. Furthermore, conflicts can occur when using direct mounts. An example of this would be multiple disk images both needing to be mapped to /usr/bin.
- To share the operating system between multiple instances, the root directory would have to be marked as read-only. This can potentially lead to

data loss if an application attempts to write to a location that has no writable disk image mounted to it. This is somewhat of a limitation for current hypervisors, as they do not currently have locking mechanisms that would allow for better sharing.

- Developers and system administrators would have to learn yet another custom configuration management tool.

Overall, using UnionFS allowed for a much cleaner solution, because we didn't have to build the extra infrastructure to maintain and distribute directory-mapping information. It allowed us to have a single instance layer that sat on top of all the others with write capabilities only given to that specific instance.

Going Forward

The resurgence of virtualization and the construction of fiber networks have greatly impacted the information technology landscape. These two advances have made cloud computing competitive and reliable. Corporations, universities, and institutions that lack sufficient knowledge to capitalize on the advantages provided by virtualization are unable to move toward it. The STORM system successfully allows these entities to capitalize on virtualization without requiring large expenditures in virtual machine manager expertise.

One may ask, what things will cloud computing allow us to do next? Once things become more established, I imagine we will see capabilities for cross-cloud computing. Administrators will watch their virtual machines "pack their bags" and move out west for more sun (assuming data centers become solar powered [6]). Sophisticated and yet simple tools will be required to manage virtual machines in cross-cloud computing. It is our plan to continue STORM development as one of these management tools.

For more information and details about STORM, we suggest reading our paper published in the LISA '08 Proceedings [1].

REFERENCES

[1] Mark Dehus and Dirk Grunwald, "STORM: Simple Tools for Resource Management," *Proceedings of LISA '08: 22nd Large Installation System Administration Conference* (USENIX Association, 2008).

[2] The virtualization API: http://libvirt.org.

[3] David Quigley, Josef Sipek, Charles P. Wright, and Erez Zadok, "UnionFS: User- and Community-Oriented Development of a Unification File System," *Proceedings of the 2006 Ottawa Linux Symposium*, pp. 349–362, 2006.

[4] D.I. Wolinsky, A. Agrawal, P.O. Boykin, J.R. Davis, A. Ganguly, V. Paramygin, Y.P. Sheng, and R.J. Figueiredo, "On the Design of Virtual Machine Sandboxes for Distributed Computing in Wide-Area Overlays of Virtual Workstations," *Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing*, p. 8.

[5] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, "Xen and the Art of Virtualization," *SOSP '03: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, pp. 164–177, New York, 2003.

[6] Stacey Higginbotham, "Data Centers Will Follow the Sun and Chase the Wind": http://earth2tech.com/2008/07/25/data-centers-will-follow-the-sun-and-chase-the-wind.

PAUL ANDERSON

programming the virtual infrastructure

Paul Anderson is a researcher at the University of Edinburgh. He has a background in practical system administration and he is particularly interested in bridging the gap between theory and practice. His homepage is http://homepages.inf.ed.ac.uk/dcspaul.

dcspaul@ed.ac.uk

OVER THE PAST FEW YEARS, THERE seems to have been a growing demand for practical system configuration tools but depressingly little progress. The recent arrival of virtual machines seems only to have increased the difficulties, as well as the need. However, the “programmable” nature of these “virtual infrastructures” made me wonder whether there was anything to be learned from the corresponding development of programming languages for the early computers. It seems that there are some very interesting analogies, but in the end, the problem might be sufficiently different that a good solution is not likely to be available for some time.

In my early teens, I was fascinated by electronics. I’ve always liked creating things, and I could build devices that seemed to have a life of their own—even if they just played random musical notes or flashed colored lights. I met my first computer when I was around fifteen—we had a school trip to a local engineering firm that ran an Elliot 905, and I was hooked. We had a chance to write and run our own programs, and I remember writing Fortran code to do some numerical integration. The Fortran compiler had to be loaded off paper tape and followed by the source code. This then produced a paper tape of the object code, which you could run. The machine flashed its lights and hummed a tune as it moved the data around, then the answer came out on a teletype roll. Sometimes you could fix small errors by manually punching extra holes in the tape.

I went on to work for ICL, a UK computer company formed in 1968, during the design of the 2900 series, learning more about hardware and how to program in machine code. But it was around 1979 before I actually got my hands on my own computer hardware—I built an Acorn System I from a kit, soldering in the chips and programming in hex. Then I started working with larger microprocessor systems that supported higher-level languages and more sophisticated software. This was an interesting time because the hardware was still simple enough that it was possible to understand the whole process. I designed and built a video framegrabber, including the analog electronics—but I also wrote the LISP interpreter that processed the images.

As things became more complex, this kind of generalization got a lot harder. I was seduced by the possibilities of the first Apple Macintosh and the Sun 3/50. I wrote software in high-level languages that ran at a layer once removed from the hardware. Even the hardware design people were using software and simulations and programmable chips.

Eventually, I became interested in managing lots of machines. All of the associated system administration problems are now well documented, but at the time this was uncharted territory—how do we get the software onto all these machines without loading each one individually? How do we set up the configuration files so that the clients talk to the right servers? How do we make sure it all stays up-to-date and correct? How do we make sure that the whole installation actually does what it is intended to do?—if we even have a clear idea of what it is supposed to do in the first place!

For the past few years I've been working on "autonomic systems" and trying to create infrastructures that can reconfigure automatically in response to failures or problems with loading. But the recent explosion of interest in virtualization has increased the complexity by another order of magnitude, with modern datacenters supporting virtual machines that migrate around the physical hardware. The network connections are established by programming VLANs, and the storage comes from network-attached devices. This means that the entire datacenter is now programmable—we can rewire network connections, change storage sizes, and replace failed hardware, all by "reprogramming the virtual infrastructure."

This all begins to sound very like the early days of computer programming when software replaced the hard-wired connections. So I started to think about the analogy, and I wondered what we could learn from the way in which computer programming has evolved.

> The charm of history and its enigmatic lesson consist in the fact that, from age to age, nothing changes and yet everything is completely different.
>
> *Aldous Huxley*

It seems clear that there have been definite steps in the evolutionary process. When a particular level becomes complex enough, a new layer of abstraction develops to advance the technology to the next stage—transistors, chips, assembler code, high-level languages, operating systems, etc.—and each stage comes with new techniques, new theories, and a new generation of specialists. Yet, over the past 10–15 years, I've spent a lot of time thinking about the problems of configuring large computing installations, and I am often frustrated by how little progress there seems to be. Compared to the rigor in designing a new chip, most computing installations are set up in a very ad hoc way—there is no systematic process nor a way of demonstrating the correctness, nor is there even a clear idea of the overall specification. Why is this?

Some History

Wikipedia has a good description of some of the early computers. EDVAC was one of the first computers to support "stored programs." Before that, machines such as ENIAC had used switches and patch leads to set up the instructions. This sounds familiar to me—there was a time when I could change the network port in my office by walking down the hall and changing the patch panel. Now I have to find someone who knows how to make the right incantation to the switches! But these early computers were still programmed by the engineers with an intimate knowledge of the hardware.

In 1953, John Backus proposed the idea of a what we would now call a "higher-level" language for the IBM 704. His team created the Fortran language and the first compiler. It's fascinating to read about this process and think about the analogy with today's system configuration tools. The initial motivations were very similar—for example, efficiency. The cost of the programmers associated with a computer installation was higher than the cost of the computer itself:

> The programmer attended a one-day course on Fortran and spent some more time referring to the manual. He then programmed the job in four hours, using 47 Fortran statements. These were compiled by the 704 in six minutes, producing about 1000 instructions. He estimated that it might have taken three days to code this job by hand. [1]

Correctness (hence reliability) was also a manual process:

> He studied the output (no tracing or memory dumps were used) and was able to localise his error in a Fortran statement he had written. He rewrote the offending statement, recompiled and found that the resulting program was correct. He estimated that it might have taken three days to code this job by hand, plus an unknown time to debug it. [1]

Of course, there were other benefits too; programs were now portable between different machines, and the language was much closer to the statement of the problem to be solved. This meant that users themselves could learn one language and their programs would run on almost every computer created since that time. However, this new concept of "automatic programming" wasn't universally accepted; many people were concerned about the efficiency of the code. Backus and Heising emphasized how much the fear of not being able to compete with hand-coded assembler code influenced the design of the first Fortran compiler. And my favorite quote comes from Irvine Ziller—one of the original Fortran team:

> And in the background was the scepticism, the entrenchment of many of the people who did programming in this way at that time; what was called "hand-to-hand combat" with the machine. [2]

This definitely reminds me of the time I have spent trying to convince people to configure systems by using the tools, rather than simply hand-editing some configuration file because "it is an emergency" or "it is only a one-off."

Languages

Most practical configuration languages have not been specifically "designed"—their functions are often related closely to the operations provided by a particular tool, and their syntax and semantics have not been carefully thought out. It is interesting that the same was true of the early programming languages. John Backus writes:

> We simply made up the language as we went along. We did not regard language design as a difficult problem, merely a simple prelude to the real problem: designing a compiler which could produce efficient programs. [3]

Of course, this was all to change—people soon realized that translating the problem description into a usable program was the biggest source of effort and errors. In the fifty years since then, there has been a huge amount of work on programming languages and their related theory. (See Figure 1 for a timeline.) Backus himself gave his initial to the BNF notation, which he co-developed for describing the formal syntax of programming languages. Periodically, different paradigms have appeared. Over the years, some of these have become accepted, and others have faded away.

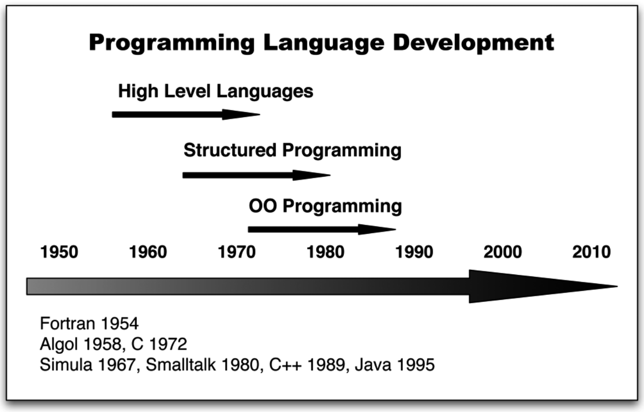


FIGURE 1: TIMELINE FOR DEVELOPMENT OF PROGRAMMING LANGUAGES, SHOWING APPROXIMATE DATES

I was quite surprised when I looked into this to see the amount of time between the "invention" of a language and its acceptance as a common production tool. Something like 10–15 years doesn't seem to be atypical. It appears to take this long for people to become comfortable with a new approach, for the features of the language to be refined, and for implementations to be accepted as stable. Of course, the increasing power of the machines along with the increasing complexity of typical code also changes the balance. If we look at the development of configuration languages on the same scale (Fig. 2), perhaps we shouldn't be surprised at the apparent lack of progress.

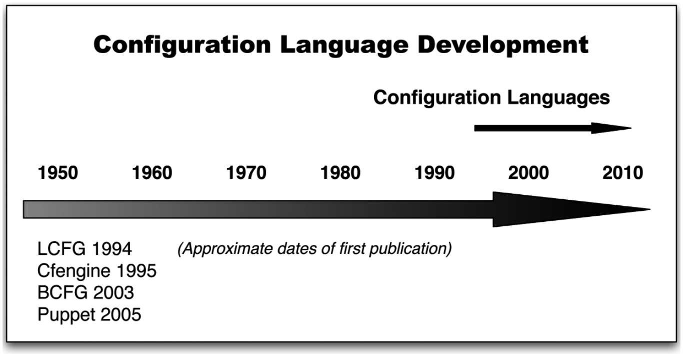


FIGURE 2: TIMELINE FOR DEVELOPMENT OF CONFIGURATION LANGUAGES, SHOWING APPROXIMATE DATES

It would be a mistake to try and draw too much of an analogy between programming and configuration languages, but it is interesting to look briefly at a few ideas and see what we can learn—both from the features that evolved and from the process itself.

Raising the Level of Abstraction

Early programming languages were based on a model that was close to the operation of the hardware. Large arithmetic expressions could be translated into multiple instructions, but the flow of control had to be specified explicitly (using conditionals and branches) in a way that mapped fairly directly onto the underlying hardware. The next significant step was the introduction of structured programming. Here, the explicit control flow was replaced with control structures; these mapped more closely onto the kind of operations that people wanted to model. Independent routines with local variables also made it easier to reuse code and for multiple programmers to work on the same project. Modern programs are supported by frameworks and operating systems that deal with entities at a much higher level of abstraction, such as files and windows.

Current configuration languages still seem to operate at a level close to the hardware—manipulating files and processes, for example. There is a big gap between this and the level at which a system administrator normally wants to talk about the infrastructure—in terms of services (mail, Web, database, etc.) and the migration strategies for the virtual machines, for example. It is certainly possible to configure multiple systems with a single statement, and it is common to have constructs that encapsulate concepts such as "Web server." But even the ability to exchange a "Web server" configuration between two sites is rare—certainly if we include all the associated consequences, such as DNS entries and firewall holes.

It is not entirely clear why there has been so little progress in raising the abstraction level. The virtual infrastructure certainly presents a few problems, such as the distributed and unreliable nature of the underlying system, that are not present when programming a single machine. However, CIM, for example, provides one possible way of modeling entities at a much higher level. Perhaps one difficulty is the relative complexity of the underlying "machine"—the infrastructure is complex and it changes rapidly as new software and services are added. System administrators tend to need a more agile approach, preferring Perl to Java and Cfengine to CIM. Or perhaps it will simply take a few more years for the appropriate paradigms to emerge.

Declarative Programming

Existing configuration languages are often "declarative" (to varying degrees). This means that the user specifies the desired configuration, and the tool works out what it needs to change to make this true. For example, you might specify that a configuration file should contain a certain line. The tool will then add that line, only if it is not already present. There isn't space here to go into detail, but declarative configuration languages have a lot of practical advantages. The problem, though, is that the tool has to work out the necessary steps by itself. This is fine when things are simple (as in the example here), but if we are specifying, say, the relationship between a set of virtual services, then working out the deployment steps can be much more complex; the placement of the virtual machines, their configuration, and the order in which we move things are all important. This may be too complex or too critical to leave completely to some automatic process.

General-purpose declarative languages such as Prolog have been around since the 1970s, but they remain confined to a comparatively small number of applications for similar reasons. Indeed, the configuration situation is actually more difficult, because the intermediate states of a configuration change may be important, whereas the intermediate states of a computation are purely internal.

Conclusion

"Automatic programming" of the virtual infrastructure is hard. It is not easy to specify correctly what is required. Translating high-level requirements into implementable specifications is hard. The languages are immature and contain considerable accidental complexity. The solutions can be difficult to compute, and automatic solutions may be difficult to understand and trust. I suspect that the idea of fully "automatic configuration," from a declarative service description, is really a myth. At several points in the history of programming, new approaches have led to talk of the "death of the programmer." Certainly the programming problems change, but they don't really become easier—they just enable a new level of abstraction.

I am still interested in languages and ways of specifying configurations. In the future, there may well be completely new approaches that provide a new degree of automation. But recently I've become interested in frameworks that might support a better integration of manual and automatic processes—this is inspired by a similar approach in AI research [4], and it may provide a smoother transition toward more automation, as the techniques become available and accepted.

REFERENCES

[1] J.W. Backus et al., "The FORTRAN Automatic Coding System": http://archive.computerhistory.org/resources/text/Fortran/102663113.05.01.acc.pdf.

[2] *The Bulletin of the Computer Conservation Society*, Number 41, Autumn 2007: http://www.cs.man.ac.uk/CCS/res/res41.htm.

[3] Computer History Museum, Fellow Awards, 1997—John Backus: http://www.computerhistory.org/fellowawards/index.php?id=70.

[4] I-X: Technology for Intelligent Systems: http://www.aiai.ed.ac.uk/project/ix/.

COREY BRUNE

Python: an untapped resource in system administration

Corey Brune is currently a Master Consultant in Plano, Texas. He has been a UNIX Engineer for over 13 years, specializing in systems development, networking, architecture, and performance and tuning. His latest project is the development of parsers for performance tuning and capacity planning.

cbrune@sim10tech.com

HISTORICALLY, SYSTEM ADMINISTRATORS chose Perl as their preferred scripting language. However, the functionality of Python may surprise those not familiar with the language. I hope to illustrate the benefits of Python within system administration while highlighting script functionality within a password-file-to-LDAP conversion script.

Python is an ideal language for beginning and expert programmers. Organizations such as Google, the New York Stock Exchange, and NASA have benefited from Python. Furthermore, Python is used behind Red Hat menu systems as well as BitTorrent clients. As a system administrator, I find Python to be an exciting and rewarding open-source language. Many first-time users are surprised at the speed at which the code falls into place when beginning to program. However, it is in the large and demanding projects that you will find Python most beneficial. This is where you will find increased manageability and time savings as opposed to other languages. Not only does Python aid in rapid deployment, but its functionality, ease of use, portability, and dependability result in hassle-free administration. Furthermore, it is compatible with all standard operating systems.

Python's ease of use is achieved primarily in the language's maintainability and elimination of code redundancy. Elimination of code redundancy is important, since duplicating code consumes time and resources. In regard to maintainability, you will notice that reading scripts is made easy. For example, Perl can be time-consuming and difficult to maintain, primarily with large programs. You want a script that can be easily read and supported, especially when modifying past or unfamiliar programs. Syntax is easy to use, read, and understand. This is primarily achieved with the language's straightforward code and similarity to the English language. Since it is object-oriented, it lends itself to easily creating modules and reducing code duplication. Python also supports Functional Programming (FP), leaving it up to the programmer to use Object Oriented Programming (OOP) or FP. Furthermore, a user can easily reuse previously created code, and specific modules can be tested rather than the entire program.

My goal in highlighting this script is to illustrate the ease in which the code falls together while spotlighting process interactions. Furthermore, I hope to demonstrate code simplicity while defining methodology.

Password-to-LDAP Conversion Script

You can find the entire listing for this script online [1]. In this article, I will just cover the highlights of the script as a way of describing Python syntax. I will also show how modules make it easy to perform system administration tasks.

Similarly to Perl and Java, Python's extensive library contains built-in modules that may be used to simplify and reduce development time. These include import pwd, sys, os, csv, and subprocess.

Note that Python statements do not end in a terminating semicolon; rather, the terminator is the end of the logical line itself.

The def keyword is used to declare a method:

```
def main(argv):
```

The main declaration accepts a list named argv. argv is similar to Perl's @ARGV or C language's **argv and contains the command-line arguments passed to the program.

Exception handling is how errors are managed within a program. For example, consider the following:

```
try:

    <code>
except IOError, (errno, strerror):
    sys.exit('I/O error (%s): %s' % (errno, strerror))

except ValueError:
    sys.exit('Could not find x in string %s: %s' % (pwLine, sys.exc_info()[0]))
```

When an exception is thrown, the programmer decides whether the script will exit, call another method, or prompt for user action. Multiple exceptions may be nested in a try block. The sys.exc_info() method returns the last exception caught.

Parsing Comma Separated Value (csv) files is simplified with the csv module. Instead of using methods such as split() to parse these files, the csv.reader() or csv.writer() methods allow for a standard mechanism for reading or writing csv files:

```
fd = csv.reader(open('shadow','r'), delimiter=':')
```

The delimiter argument allows reading or writing different csv files. Notice that the results of open() are used as the argument to csv.reader(). This syntax is common usage throughout Python.

Here is an example of file IO with Python:

```
ldifOut = open(ldifFile, 'w+')
ldifOut.close()
```

The first argument is the filename, and the second argument is the mode. There are different modes available depending on the type of operation required: read (r), read-append(r+), write (w), write-append (w+), or append (a+). The return value is a file object assigned to the variable ldifOut. The close() method closes the file object.

Lists are one of the most dynamic data types in Python. Open and close brackets with comma-delimited values declare a list. The example here declares an empty list, pwLine:

```
pwLine = []
```

Lists may be indexed, split, and searched. Furthermore, they may be used as a stack or queue and may contain other data types.

In the following code:

```
for row in fd:
    if row[1] in '*LK*' 'NP' '*' '!!':
        continue
```

the for loop iterates through the file object fd until the end of file. Conditionals and loops are terminated with a colon. Unlike other languages, loops, conditionals, and other statements are "grouped by using indentation" (python.org). The if statement says if row*[1]* matches *LK*, NP, **, or !!, to continue to the next line in the file, since these are local accounts such as root or nobody.

Python contains many of the C standard UNIX/Linux system functions and usage is similar to C functions. The pwd.getpwnam() method generates the list of users to be converted to the LDAP script:

```
String = pwd.getpwnam(line[0])
```

The argument passed is line[0], which is the username.

The pwd.getpwnam() method returns a tuple. Tuples, like strings, are read-only or immutable data types. To modify the tuple, we convert the tuple to a list:

```
pwLine = list(String)
```

Python offers many types of conversion methods, such as int(), float(), and str().

The code:

```
index = pwLine.index('x')
pwLine.pop(index)
pwLine.insert(index, line[1])
```

illustrates some of the methods available for list manipulation. pwLine.index('x') returns an integer value to where the value was found. If the value is not found, a ValueError exception is thrown. pwLine.pop(index) removes and returns the value at index. pwLine.insert(index, line[1]) inserts the encrypted password (line[1]) into the list at index.

Using the write() method allows for updating or writing files:

```
ldifOut.write('dn: cn=' + pwLine[0] + ',' + fullDN + '\n')
```

The arguments to write() illustrate how to use string concatenation with the + sign. You can access individual elements in a list or string by using brackets []. In the example here, pwLine[0] accesses the first data element. Note that lists and strings start at index number 0.

The subprocess module is the preferred mechanism when interacting with processes:

```
output = subprocess.Popen(ldapStr, shell=True, stdout=subprocess.PIPE)
```

```
output.wait()
stdout_value = output.communicate()[0]
```

subprocess.Popen() is used to invoke ldapadd and the associated arguments. shell=True indicates that the command will be passed to the shell; otherwise os.execvp() is executed. stdout=subproces.PIPE contains a pipe to control the output. Other pipes may be created for stderr and stdin. The variable output is assigned a Popen object. wait() is called to allow for the process to finish. Process output is then retrieved with the communicate()[0] method.

Every module or method has many convenient built-in methods. These are denoted by underscores on either side of the name, for example:

```
if __name__ == '__main__':
   main(sys.argv[1:])
```

The build-in method __name__ defines the module's name. The next line passes sys.argv[1:] to main(). List elements may be accessed by listname[start index:end index]. In this example, the list sys.argv[1:] will pass elements starting at index 1 through the last element.

General Notes

Python uses indentation for code blocks, such as loops and conditionals, rather than semicolons, braces, or parentheses. Statements do not require semicolons for termination.

Python contains built-in data types referred to as *numbers*, *strings, lists, tuples*, and *dictionaries*. These data types are represented by characters such as parentheses, brackets, and braces. Every data type is an object and has associated methods for manipulation.

File parsing is made simple in Python with the module re. If you are familiar with Perl, you will notice that Regular Expression Syntax is similar. Python has the capability to handle large files such as XML, CSV, binary, and text files.

Conclusion

Although I have only skimmed the surface of Python's functionality and syntax, I hope to have provided a foundation for further exploration. The application range for Python crosses over many domains such as system administration, game programming, Web programming, and research and development. The extensive library and maintainability of code make this a versatile language. Examples of functionality are highlighted in the interaction of processes, file parsing, and exceptions. If you have dabbled with Python in the past, this is an opportunity to revisit the language. Soon after programming in Python, you may find its range spreading into all aspects of administration. For further information and resources visit www.python.org/.

REFERENCE

[1] http://www.sim10tech.com/code/python/passwd2ldap.py.

DAVID N. BLANK-EDELMAN

practical Perl tools: be the browser

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the O'Reilly book *Perl for System Administration*. He has spent the past 24+ years as a system/network administrator in large multi-platform environments, including Brandeis University, Cambridge Technology Group, and the MIT Media Laboratory. He was the program chair of the LISA '05 conference and one of the LISA '06 Invited Talks co-chairs.

dnb@ccs.neu.edu

BELIEVE IT OR NOT, MY SYSADMIN-themed column for this issue is about Web crawling, automation, scraping, browsing, circumnavigating—whatever you'd like to call it. Why is this something a sysadmin would want to automate, versus, say, a Web developer? Besides your everyday sysadmin-related surfing (searching for reference material or answers to questions, participating in the community, etc.), you may have noticed the increasing number of Web sites to which you need to connect strictly to get your job done. Maybe you need to work with mailing lists, submit requests to a certificate authority, interact with a trouble-ticket system, or deal with any number of Web applications. Surely it would be pleasant to reduce the amount of menial pointing and clicking you do on a daily basis. This column can help.

One quick caveat before we dive in, mostly for the regular readers of the column. In most of my columns I've tried to present a number of tools to perform a task. I like the idea that my readers can assemble their own tool chest from which to choose the best utensil or approach. Well, in today's column I'm going to hand you a single spanner wrench and say, "Here, here's the best tool."

In this column we're going to focus on using just the module WWW::Mechanize and the modules in its orbit. There are other modules out there for performing tasks like the ones we're going to explore, but I've not found one to date that I've liked better than WWW::Mechanize. I should note that I'm not the only person enamored of this module. Mech, as it is affectionately called, has been reimplemented in several other of your favorite languages: py-mechanize (in Python) and WWW::Mechanize (in Ruby). The boundary between Perl and Ruby is actually porous in both directions. We're not going to look at it in this column, but WWW::Mechanize::Plugin::Web::Scraper lets you use the module Web::Scraper from WWW::Mechanize. Web::Scraper takes its design from the excellent Ruby-based scrAPI toolkit.

First Steps with WWW::Mechanize

Almost all WWW::Mechanize scripts start out like this:

```
use WWW::Mechanize;

my $mech = WWW::Mechanize->new();
$mech->get($url); # get can also take a :content_file param to save to a file
```

We initialize a new Mech object and ask it to go fetch some Web page. If we want the contents of the page we just fetched, we call:

```
my $pagecontents = $mech->content();
```

It's not uncommon to hand the results of the content() method off to some other module (for example, we could feed it to HTML::TableExtract, one of my favorite modules) to do more sophisticated parsing. We won't see an example of that handoff in this column, but I wouldn't be surprised if it didn't find its way into a future column.

OK, so far the code has been really simple. So simple LWP::Simple could have basically handled it. Let's take things to the next level:

```
use WWW::Mechanize;

my $mech = WWW::Mechanize->new();

$mech->get( 'http://www.amazon.com' );
$mech->follow_link( text => 'Help' );
print $mech->uri . "\n";

# prints out something like:
# http://www.amazon.com/gp/help/customer/display.html?ie=UTF8&nodeId
# =508510
```

So what happened here? WWW::Mechanize retrieved the home page for Amazon and then found the link on the page with the text 'Help'. It followed the link in the same way you would on a browser and retrieved the contents of the URL specified in the link. If we called `$mech- >content()` at this poinr, we'd get back the contents of the new page found by browsing to the selected link.

If we wanted to, we could use an even cooler feature and write something like:

```
$mech->follow_link ( text_regex => qr/rates.*policies/ );
# or
$mech->follow_link ( url_regex => qr/gourmet.*food/ );
```

The first line of code will find and then follow the first link whose text matches the given regular expression. This means we can follow links in a page without knowing the precise text used (e.g., if each page was generated dynamically and had unique links). The second line of code performs a similar find and follow, this time based on the URL in the link.

`follow_link()` has a number of other options available. There's a related `url => 'http://...'` option equivalent to the text => 'text' option that will take a fully specified URL to follow. Though this is more fragile, `follow_link` can also take an `n =>` option to allow you to choose the *n*th link on the page. All of the options mentioned so far can be compounded. If you want the third 'help' related link on a page with a URL that included the path 'forum' in its name you could write:

```
$mech->follow_link( text => 'help', url_regex => 'forum', n => 3 );
```

If for some reason you want to just find the links on a page without navigating to their target, WWW::Mechanize provides `find_link` and `find_all_links`, which take the same selector arguments as `follow_link`. WWW::Mechanize can also find images on a page via `find_images` and `find_all_images`, which use similar arguments.

WWW::Mechanize Tip #1: mech-dump

Now that we've seen some of the basic stuff, I'd like to show you a couple of tips that will make the more complex WWW::Mechanize work we're going to explore considerably easier.

The first tip involves a utility that ships with WWW::Mechanize and optionally gets installed during the module's installation: mech-dump. mech-dump calls WWW::Mechanize and gives you a little bit of insight into how WWW::Mechanize is parsing a particular page. It offers four choices:

- 1. Display all forms found on a page.
- 2. Display all links found on a page.
- 3. Display all images found on a page.
- 4. Display all of the above.

Let's see it in action:

```
$ mech-dump --links http://www.amazon.com
http://www.amazon.com/access
/
/gp/yourstore/ref=pd_irl_gw?ie=UTF8&signIn=1
/gp/yourstore/home/ref=topnav_ys_gw
  ...
```

I cut that list off quickly, because:

```
$ mech-dump --links http://www.amazon.com|wc -l
  247
```

Finding links can be helpful, but this command really shines when it comes time to interact with forms, something we're going to do in just a moment:

```
$ mech-dump --forms http://www.boingboing.net

GET http://www.google.com/search
  ie=UTF-8                                 (hidden readonly)
  oe=UTF-8                                 (hidden readonly)
  domains=boingboing.net                   (hidden readonly)
  sitesearch=boingboing.net                (hidden readonly)
  q=                                       (text)
  btnG=Search                              (submit)

POST http://www.feedburner.com/fb/a/emailverify
  email=                                   (text)
  url=http://feeds.feedburner.com/~e?ffid=18399 (hidden readonly)
  title=Boing Boing                        (hidden readonly)
  loc=en_US                                (hidden readonly)
  <NONAME>=Subscribe                       (submit)
```

The output shows us that each form has a number of fields. Some are hidden fields set in the form by the form's author, but the useful information in the output is the fields that someone sitting at a browser would need to fill and select. For example, the blog BoingBoing has an option to allow people to subscribe via email using a Feedburner service. The output of mech-dump lets us know that we'll be filling in a field called "email" (versus something else such as "address" or "user_email" or any number of possibilities).

Let's put this utility into practice to help with our WWW::Mechanize programming. I recently had a need to scrape some information from our internal wiki. We use a commercial product that has its own login screen to control access (versus depending on external authentication performed by the Web server). To get past the login screen to the content I needed, my program had to fill in the Web form on the login screen. The first step toward figuring out how to do so was to run a `mech-dump --forms` on that page. (A quick aside: Mech can talk to anything LWP::UserAgent can talk to. This means that pages served over HTTPS are not a problem as long as you have Crypt::SSLeay or IO::Socket::SSL installed.)

`mech-dump --forms` returned something like the following:

```
GET https://wiki.example.edu/dosearchsite.action
where=conf_all (hidden readonly)
queryString= (text)
<NONAME>=Search (submit)
<NONAME>=Search (hidden readonly)

POST https://wiki.example.edu/login.action [loginform]
os_username= (text)
os_password= (password)
os_cookie=<UNDEF> (checkbox) [*<UNDEF>/off|true]
login=Log In (submit)
os_destination= (hidden readonly)
```

The first form on the page is a search box; the second is exactly the one I needed. The output for the second form told me that I needed to provide two fields to log in: `os_username` and `os_password`. In WWW::Mechanize you can use the `submit_form` method to fill in a form, like so:

```
use WWW::Mechanize;

my $mech = WWW::Mechanize->new();
$mech->get( $loginurl );
$mech->submit_form(
form_number => 2,
fields => { os_username => $user, os_password => $pass },
);
```

`submit_form` chooses the form to use, fills in the given fields, and performs the "submit" action (the equivalent of selecting the 'Log In' element on the page). Now the script is "logged in" to the wiki and can proceed to operate on the protected pages.

WWW::Mechanize Tip #2: WWW::Mechanize::Shell

The second tip I want to pass on is about a companion module called WWW::Mechanize::Shell. WWW::Mechanize::Shell calls itself "an interactive shell for WWW::Mechanize." If you type the following slightly unwieldy command:

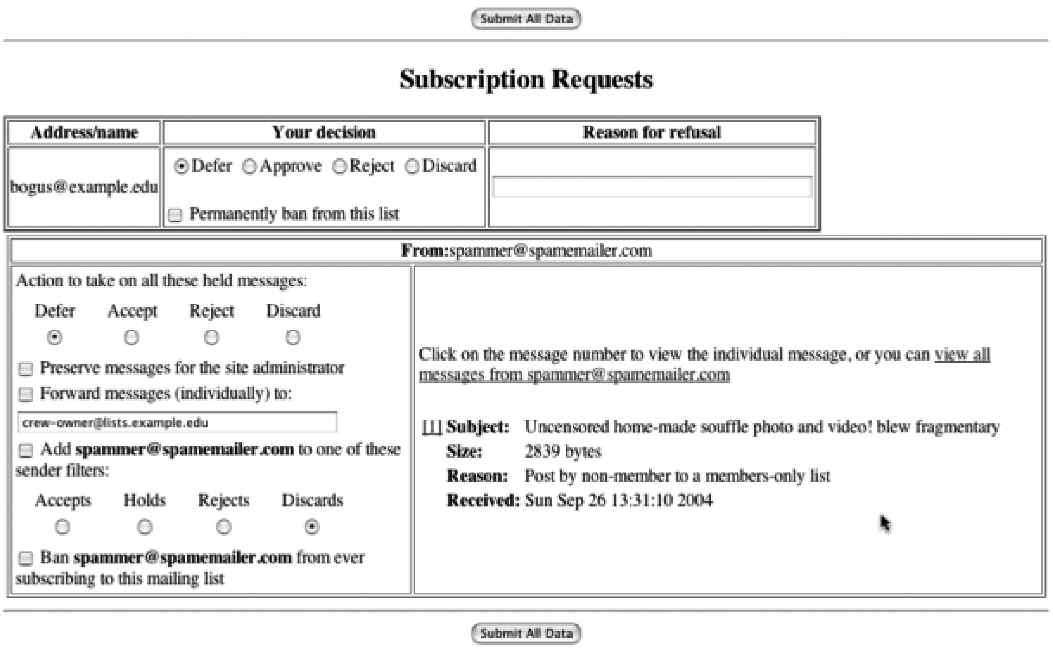
```
$ perl -MWWW::Mechanize::Shell -eshell
```

you get an interactive shell with the following commands:

```
(no url)>help
Type 'help command' for more detailed help on a command.
   Commands:
   auth:      Sets basic authentication credentials.
   autofill:  Defines an automatic value.
   back:      Goes back one page in the browser page history.
   browse:    Opens the Web browser with the current page.
   click:     Clicks on the button named NAME.
   comment:   Adds a comment to the script and the history.
   content:   Displays the content for the current page.
   cookies:   Sets the cookie file name.
   ct:        Prints the content type of the most current response.
   dump:      Dumps the values of the current form.
   eval:      Evaluates Perl code and print the result.
   exit:      Exits the program.
   fillout:   Fills out the current form.
   form:      Selects the form named NAME.
   forms:     Displays all forms on the current page.
   get:       Downloads a specific URL.
   headers:   Prints all C<< <H1> >> through C<< <H5> >> strings found in
              the content.
   help:      Prints this screen, or help on 'command'.
   history:   Displays your current session history as the relevant commands.
   links:     Displays all links on a page.
   open:      <open> accepts one argument, which can be a regular
              expression or the number.
   parse:     Dumps the output of HTML::TokeParser of the current content.
   quit:      Exits the program.
   referer:   Alias for referrer.
   referrer:  Sets the value of the Referer: header.
   reload:    Repeats the last request, thus reloading the current page.
   response:  Displays the last server response.
   restart:   Restarts the shell.
   save:      Downloads a link into a file.
   script:    Displays your current session history as a Perl script using
              WWW::Mechanize.
   set:       Sets a shell option.
   source:    Executes a batch of commands from a file.
   submit:    Submits the form without clicking on any button.
   table:     Displays a table described by the columns COLUMNS.
   tables:    Displays a list of tables.
   tick:      Sets checkbox marks.
   timeout:   Sets new timeout value for the agent. Affects all subsequent.
   title:     Displays the current page title as found.
   ua:        Gets/sets the current user agent.
   untick:    Removes checkbox marks.
   value:     Sets a form value.
   versions:  Prints the version numbers of important modules.
```

Seeing the whole list is a little intimidating, so I'll pick out a few choice commands for an example that will get us back into the main discussion of WWW::Mechanize use again. Let's look at a more complex forms example from a real-life problem.

One common Web-based administrative interface is the one used to configure and administer Mailman (http://www.list.org/) mailing lists. This interface mostly makes working with the server easier, but there have been some issues. One issue that used to exist (it has been fixed in later versions) was its handling of large batches of spam. If a spammer sent a large batch of mail that was intercepted by Mailman and queued for an administrator to triage, that admin had to click the buttons to delete each message individually in a form that looks like that in Figure 1.



Subscription Requests

Address/name	Your decision	Reason for refusal
bogus@example.edu	[✓ Defer] Approve Reject Discard ☐ Permanently ban from this list	

From:spammer@spamemailer.com

Action to take on all these held messages:
[✓ Defer] Accept Reject Discard
☐ Preserve messages for the site administrator
☐ Forward messages (individually) to:
crew-owner@lists.example.edu
☐ Add **spammer@spamemailer.com** to one of these sender filters:
Accepts Holds Rejects [✓ Discards]
☐ Ban **spammer@spamemailer.com** from ever subscribing to this mailing list

Click on the message number to view the individual message, or you can view all messages from spammer@spamemailer.com

[1] **Subject:** Uncensored home-made souffle photo and video! blew fragmentary
Size: 2839 bytes
Reason: Post by non-member to a members-only list
Received: Sun Sep 26 13:31:10 2004

FIGURE 1: MAILMAN FORM

If you get 800 messages in the queue, this gets old quickly. Let's see how we can use WWW::Mechanize::Shell and WWW::Mechanize to make that pain go away.

The easiest way to start is to poke at the Mailman server with WWW::Mechanize::Shell:

```
$ perl -MWWW::Mechanize::Shell -eshell
   > get https://lists.example.edu/bin/admindb/mylist
   > form
   Form [1] (<no name>)
   POST https://lists.example.edu/bin/admindb/mylist
           adminpw= (password)
           admlogin=Let me in... (submit)
```

(where the output has been slightly edited for readability). OK, so this means that we will need to fill in adminpw to get access to the administration page. We set that value and submit the form. Let's do so:

```
> value adminpw listpw
> click admlogin
```

If we were to use the `form` command now we'd see a reasonably complex form dump that included lines such as:

```
senderaction-spammer%40spamemailer.com=0 (radio) [*0|1|2|3]
senderpreserve-spammer%40spamemailer.com=<UNDEF> (checkbox)
   [*<UNDEF>/off|1/?Preserve messages for the site administrator]
senderforward-spammer%40spamemailer.com=<UNDEF> (checkbox)
   [*<UNDEF>/off|1/?Forward messages (individually) to:]
senderforwardto-spammer%40spamemailer.com=mylist-owner@
lists.example.edu (text)
```

```
senderfilter-spammer%40spamemailer.com=3 (radio) [6|7|2|*3]
senderbanp-spammer%40spamemailer.com=<UNDEF> (checkbox)
[*<UNDEF>
```

These are all form elements we'd like to set without a major point-and-click fest. The tricky thing is that those elements are all message-sender specific. It says "senderaction-{some email address}" not just "senderaction." Here's where WWW::Mechanize-related options that take regular expressions come to our rescue. We can tell WWW::Mechanize::Shell to set elements that match a regular expression. Let's set all of those fields:

```
> autofill /senderfilter/ Keep
> autofill /senderfilterp/ Keep
> autofill /senderbanp/ Keep
> autofill /senderpreserve/ Keep
> autofill /senderforward/ Keep
> autofill /senderforwardto/ Keep
> autofill /senderaction/ Fixed 3
```

That last one might look a bit strange. It comes from the senderaction form definition. If you look carefully at the HTML source for the Mailman page (not reproduced here), you'll see that "3" in that case is the "Discard" option. In case you are curious about why we're setting all of the options in the form and not just the senderaction one, it's because the next command we're going to use will interactively prompt for any elements that don't have values set already:

```
> fillout
> click submit
```

Congrats! You've just avoided clicking on many hundreds of form elements, because they were set programmatically and the form has been submitted.

Now, here's an even cooler trick: If you type `script` at this point, it will spit out a Perl script that essentially reproduces your interactive session. If we were to run this command after typing in the session above (and edit it to remove some of the initial probing of the forms), we'd get output that looks like this:

```
#!perl -w
use strict;
use WWW::Mechanize;
use WWW::Mechanize::FormFiller;
use URI::URL;

my $agent = WWW::Mechanize->new( autocheck => 1 );
my $formfiller = WWW::Mechanize::FormFiller->new();
$agent->env_proxy();

$agent->get('https://lists.example.edu/bin/admindb/mylist');
$agent->form(1) if $agent->forms and scalar @{$agent->forms};
{ local $^W; $agent->current_form->value('adminpw', 'listpw'); };
$agent->click('admlogin');
$formfiller->add_filler( qr((?-xism:senderfilter)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderfilterp)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderbanp)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderpreserve)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderforward)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderforwardto)) => "Keep" => );
$formfiller->add_filler( qr((?-xism:senderaction)) => "Fixed" => '3');
$formfiller->fill_form($agent->current_form);
$agent->click('submit');
```

This code has some components (e.g., the use of the WWW::Mechanize::FormFiller module) that we haven't explored in this column, but hopefully you get the basic idea that an interactive session can be the basis for automatically generated code that can be further customized.

In this column we've gotten a taste of some of the major features of WWW::Mechanize. But that's only the core of this subject. There's a whole ecosystem of additional add-on modules that has grown up around WWW::Mechanize that I'd encourage you to explore. These modules add features such as timed requests and human-like requests with delays between them (should you want to test your Web site), and experimental JavaScript support. Web work with WWW::Mechanize can be both time-saving and fun, so enjoy! Take care, and I'll see you next time.

PETER BAER GALVIN

Pete's all things Sun: analytics for the masses (of storage administrators)

Peter Baer Galvin is the Chief Technologist for Corporate Technologies, a premier systems integrator and VAR (www.cptech.com). Before that, Peter was the systems manager for Brown University's Computer Science Department. He has written articles and columns for many publications and is co-author of the *Operating Systems Concepts* and *Applied Operating Systems Concepts* textbooks. As a consultant and trainer, Peter teaches tutorials and gives talks on security and system administration worldwide. Peter blogs at http://pbgalvin.wordpress.com and twitters as "PeterGalvin." His All Things Sun Wiki is http://wiki.sage.org/bin/view/Main/AllThingsSun.

pbg@cptech.com

WHILE WORKING WITH A CLIENT ON deploying a new server that was attached to its SAN, the performance was supposed to improve. Unfortunately, even though we theoretically doubled the CPU and memory being used to run the database, performance was about the same. After digging into the problem (with DTrace, of course), we determined that the SAN was providing disk I/O at very low rates. Unfortunately, the SAN had been having performance problems for quite some time but the vendor had been unable to determine the cause of the problems or, better yet, a solution to the problems. This situation is all too common. Storage is very difficult to performance-analyze and is frequently blamed (sometimes wrongly) for sitewide performance issues. Wouldn't it be great to have the convenience of centralized storage, but with the power of DTrace? Welcome to the Sun Storage 7000.

When Sun introduced DTrace [1] as part of Solaris 10, it provided an unprecedented new tool for system administrators, systems programmers, and developers. DTrace won awards, but more importantly it won the hearts and minds of those trying to analyze the performance of their applications and systems. Almost literally, systems performance analysis moved from a world where performance problems were ignored, had best-guess logic applied, or were tested via trial and error and where problems needed to be recreated in a nonproduction environment for exploration, to a new world where a problem could be dissected astonishingly quickly, in production, with no special development or compilation steps needed. In summary, DTrace moved systems analysis from the Stone Age to the Iron Age.

With the release of the Sun Storage 7000 line of storage appliances, Sun has included an "Analytics" toolkit that once again moves performance analysis forward, this time to the Modern Age. These analytics are based on DTrace but essentially hide the DTrace complexity under a cloak of Ajax-based browser graphics. With a few clicks of the mouse a storage administrator can determine which clients are causing which files on the server to be "hot," or resource-use–intensive. A few more clicks and that administrator can see the latency of each request to the blocks of that file, or how many requests of each protocol are being processed, or how many cache hits a file had. The list is almost endless. In this episode of PATS, I'll explore the Sun Storage 7000 analytics tool, the Modern Age of storage performance analysis.

OVERVIEW

Because the new analytics are based on DTrace, we should start there. The first public unveiling of DTrace was as a paper at the 2004 USENIX Annual Technical Conference. From that auspicious start, DTrace has been the talk of the town. It has been added to other operating systems, including FreeBSD and Mac OS X, and has led to much discussion in the Linux community about adding similar features there. DTrace, along with ZFS and Containers, provides Solaris with a world-class set of capabilities that is causing even some long-term adversaries, including IBM and Dell, to support Solaris on their hardware.

DTrace is a scalpel of a system analysis tool, given its depth and breadth of abilities, but like a scalpel is best used only by surgeons. DTrace has its own programming language, and it started as command-line only. Since then, DTrace has been included in three GUI programming environments. Sun Studio Express [2], the latest version of Sun's IDE, includes project D-Light for DTrace visualization. Also, the NetBeans DTrace GUI Plugin can be installed into the Sun Studio IDE using the NetBeans Plugin Portal for Java debugging [3]. For those using Mac OS X, the free XCode IDE includes the "instruments" feature, which is a DTrace-based GUI [4]. Given their IDE flavor, these tools are mostly useful for developers, but they can be stretched for general-purpose use. Chime [5] is an open-source project that provides a visualization wrapper around DTrace. Although it is a step in the right direction, it is far from being a general-purpose DTrace GUI for system administration use.

This leads us to the Sun Storage 7000, which was announced and started shipping in November 2008 [6]. The 7000 is a line of NAS storage appliances which includes the 7110, 7210, and 7410 products. It was developed by the Fishworks team at Sun as a fully integrated software and hardware (fish) solution. It is Sun's first OpenStorage product, based on commodity hardware and open-source software (OpenSolaris), but it is far more than the sum of its parts. The 7000 line is a full appliance, with no sign of Solaris (or DTrace, or ZFS, for that matter). I won't review the full feature set here, but it includes clustering, phone home support, snapshots, clones, replication, compression, NFS, CIFS, FTP, HTTP, WebDav and iSCSI protocols, and GUI and CLI management interfaces. The 7000 line uses read-oriented and write-oriented SSDs to increase performance, while using SATA and SAS drives for density, power conservation, and cost-effectiveness. All this comes in addition to the analytics, which will be covered in the remainder of this column.

Try It, You'll Like It

There are two great ways to explore the Sun Storage 7000. Sun has a try-and-buy program for many of its products, including some of the 7000 line [7]. Sun pays to ship the system to you, and it will even pay to ship it back if you decide not to keep it. This is a very hassle-free way to be sure the Sun products meet your needs before committing to their purchase. If you prefer instant gratification and want to try out the 7000 appliance software, simply download the Sun Unified Storage Simulator, provided by Sun as a VMware image. On Windows or Linux you could use VMware player [8], for free, to run the virtual machine. On Mac OS X there is no free version of VMware Fusion [9], but it does have a trial period. Also, virtualbox, the open-source virtual machine package now owned by Sun, should be able to play VMware images.

For this column I used the simulator, which is a full implementation of the software that runs the 7000 appliance. Although the simulator cannot be used for performance testing (or production storage use), it is as close to the real thing as is needed for evaluation, planning, and experimentation.

Analytics

The analytics component of the Sun Storage 7000 line is feature-rich. Most important, it can provide an astonishing amount of useful information to a storage administrator who is trying to manage and monitor the appliance and the files and blocks stored there. Just like DTrace, the analytics run in real time, and they allow quick progression from hypothesis through data gathering to new hypothesis, data, and conclusions. Unlike DTrace, the analytics component has a very complete and useful graphical interface and visualization engine.

Joerg Moellenkamp has posted a nice blog entry proving a walkthrough of setting up the 7000 software, configuring it to a point that it is ready to be managed by the GUI [10]. After setting up my virtual machine, I configured the virtual disks that are included in that machine to be RAID double parity. NFS service is enabled by default, so nothing was needed there. I then created the userid "pbg" for myself and created a share called "test" owned by "pbg." The share was automatically exported as /export/test. I mounted that share from my Mac and used the analytics to watch the virtual appliance. The GUI is accessed by browsing via https to its IP address at port 215.

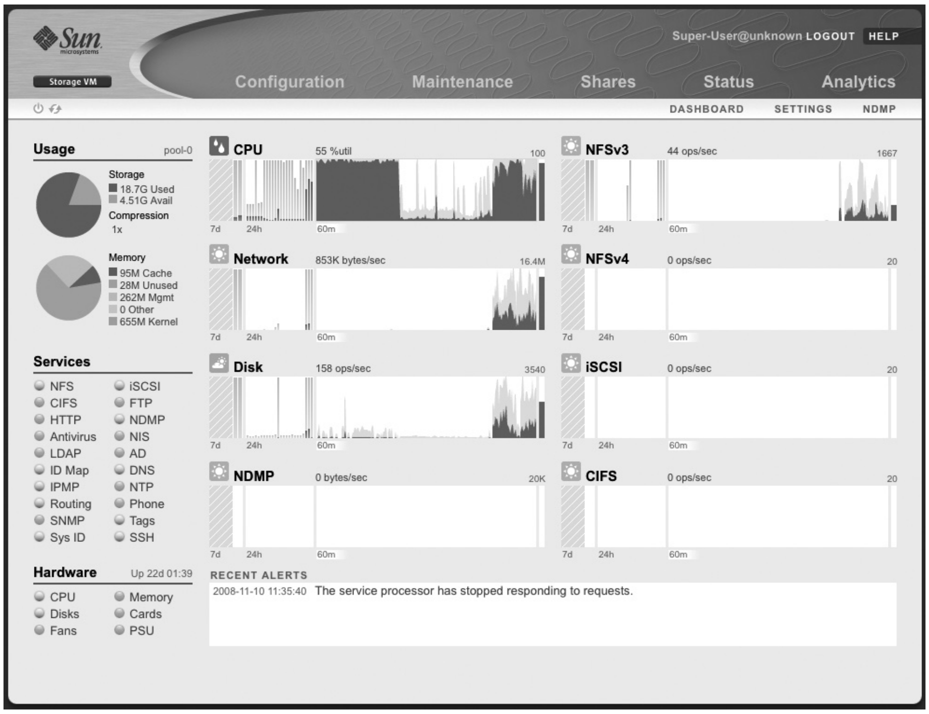
For more details on how analytics work, take a look at the presentation put together by members of the Fishworks team that implemented them [11]. All things Fishworks-centric (videos, blogs, white papers, and more) are also available online [12].

Some examples of what Sun Storage 7000 analytics can do should go a long way toward understanding the power and flexibility of the tool. The analytics can show:

- What clients are making CFS requests
- What NFS files are currently being accessed
- How long NFS operations are taking
- What LUNS are being written to
- What files are being accessed from a specific client
- The read/write mix for a specific file by a specific client
- How long writes are taking to a specific file at a specific offset in that file

All of these metrics, and many more, are shown and optionally recorded on a per-second basis. Recorded data can be examined historically for event correlation or trend analysis. Generally, instrumentation is done at a level of abstraction above the implementation, at a level of detail that system administrators care about. The system conveys both the load placed on the appliance and how the appliance is reacting to the load. For example, a problem could be too much load or not enough appliance resources, and the details are available to make that determination. The 7000's analytics allow ad hoc instrumentation, not just precanned or predetermined diagnostics, for site-specific or problem-specific debugging. The standard UNIX tables-and-numbers diagnostic output is frequently not easy to interpret and slow for the brain to understand, so the GUI manages visualization as a first-class aspect of storage analytics.

Let's have a look at the appliance management screen. The main "Status" screen gives an overview of the entire appliance, including space used and free, protocols enabled, and basic performance metrics (Figure 1). From there, clicking on a metric brings that metric into a "worksheet" on the Analytics page. An administrator may create many worksheets, store them, and switch among them to quickly look at various custom views of the activity of the appliance. This is done by selecting the "Saved Worksheets" subpage from Analytics. Many performance aspects are constantly sampled and made available for archiving, deletion, or adding to a worksheet from the "Datasets" subpage. Further, the administrator may have many open worksheets and can clone a worksheet to make another copy from the current worksheet. There seems to be no limit to the flexibility for viewing various system performance aspects, both current and past.



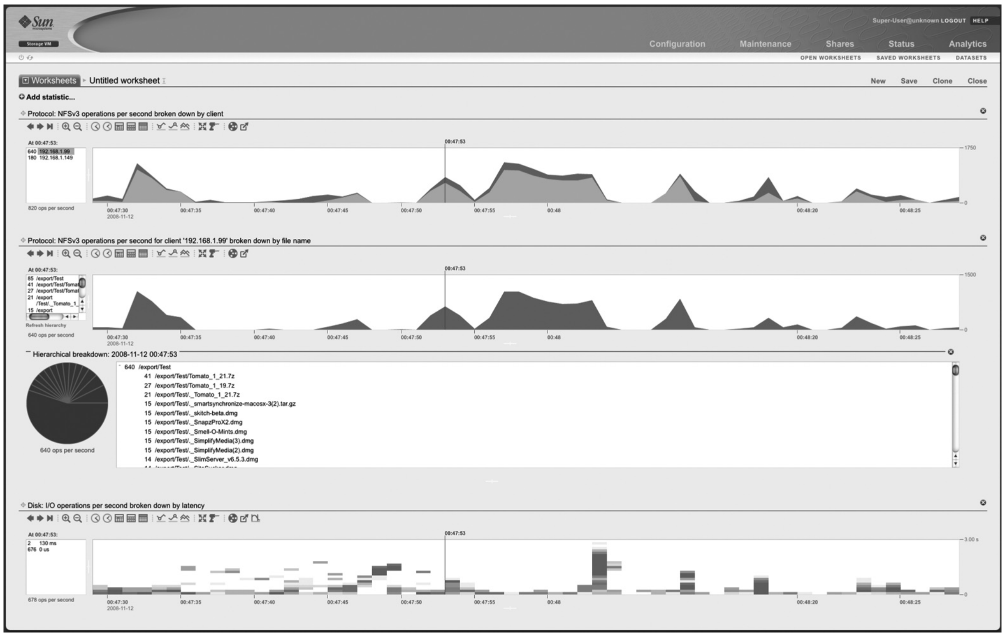
2008-11-10 11:35:40 The service processor has stopped responding to requests.

FIGURE 1: SUN STORAGE 7000 MAIN "STATUS" SCREEN

On a worksheet, the "Add Statistic..." button brings up a menu of "statistics," or system metrics, that can be added and manipulated. These statistics can in turn be broken down into constituent elements. Adding a statistic creates a new panel containing a graph of that statistic updated in real time. Averages and other "breakdown" details (for example, when a specific sample or time is clicked on within the graph) are shown to the left of the graph. Likewise, selecting an item (or multiples via shift-click) in the breakdown table highlights the corresponding data in the graph. For files and devices, a "show hierarchy" option creates a subpanel to visualize the details of that item in a pie chart and with file and device names enumerated. Again, selecting any item in any part of the panel highlights that item in the other parts of the panel. Another type of statistic is a quantified breakdown, displayed as a heat map (color-coded histogram) of the data. This is used for latency and size offsets, where scalar representation does not make sense.

Consider data where a value of zero must be distinguished from no data, such as the response time of a request.

It is difficult to describe in words the power of analytics, but the use is intuitive and the power really must be tried to be believed. A screenshot of a few graphs on one worksheet with Analytics is shown in Figure 2. There are many controls to manage a panel. Some of these are time controls. These include moving backward and forward in time, pausing the display (but not the data capture), zooming in and out (showing more or less time in the chart), and going to minute, hour, day, week, and month views of the data. Other controls manage viewing specific data, such as going to the smallest recorded sample or the largest, or directly comparing samples by showing them as line graphs rather than a stacked graph. If there is a specific time of interest in one graph, pressing the "synchronize" icon changes all graphs in the worksheet to show that time and the same time scale and to stay synchronized.



Protocol: NFSv3 operations per second broken down by client

At 00:47:53:
640 192.168.1.99
180 192.168.1.149

820 ops per second

Protocol: NFSv3 operations per second for client '192.168.1.99' broken down by file name

At 00:47:53:
85 /export/Test
41 /export/Test/Tomat
27 /export/Test/Tomat
21 /export /Test/._Tomato_1_
15 /export

640 ops per second

Hierarchical breakdown: 2008-11-12 00:47:53

```
640 /export/Test
    41 /export/Test/Tomato_1_21.7z
    27 /export/Test/Tomato_1_19.7z
    21 /export/Test/._Tomato_1_21.7z
    15 /export/Test/._smartsynchronize-macosx-3(2).tar.gz
    15 /export/Test/._skitch-beta.dmg
    15 /export/Test/._SnapzProX2.dmg
    15 /export/Test/._Smell-O-Mints.dmg
    15 /export/Test/._SimplifyMedia(3).dmg
    15 /export/Test/._SimplifyMedia(2).dmg
    14 /export/Test/._SlimServer_v6.5.3.dmg
```

640 ops per second

Disk: I/O operations per second broken down by latency

At 00:47:53:
2 130 ms
676 0 us

678 ops per second

FIGURE 2: SCREENSHOT OF GRAPHS ON A WORKSHEET WITH ANALYTICS

The "drilldown" icon starts from the current graph, allows selection of a specific attribute, and creates a new graph of just that attribute of the current graph. For example, from the "CPU: percent utilization" graph, choosing "drilldown" allows selection of "by user name" or "by application name," among other choices. The drilldown choices are specific to the graph being controlled. For example, in the graph "Disk: I/O operations per second broken down by type of operation," selecting a point in time shows the types and number of each type of operation that occurred at that time. Selecting a type of operation from the list left of the graph and choosing "drilldown" allows creation of a new graph based on that operation type, but further broken down by disk, size, latency, or offset of the operation. Shift-clicking on the drilldown icon highlights every breakdown, creating a "rainbow" differentiating every breakdown on the graph.

The penultimate per-graph control saves the current graph to the Datasets collection. On the Datasets page, each graph has its data being constantly collected. Uninteresting datasets can be deleted to save space and increase performance a bit. Interesting new graphs can be saved as a dataset and the pertinent data will then be continually collected for later examination. The last control exports the data shown in the graph to a comma-separated .csv file for importation into a spreadsheet, for example. Even more analytic options are available, and these can be enabled by selecting the "Make available advanced analytics statistics" checkbox on the Configuration->Preferences page.

Overall, the use of the analytics component of the Sun Storage 7000 NAS appliance is amazing, allowing exploration of every aspect of the performance, load, and status of the system. Sometimes the response of the GUI lagged, but the appliance's virtual machine was limited to 1 CPU and 1 GB of memory within VMware Fusion, and I was putting quite a load on the virtual machine to test out the analytics, so I expect that was an aberration. On the whole I'm glad to leave behind the Middle Ages of performance analysis and look forward to the new tools of the modern age (except for the violence inherent in the system).

Random Tidbits

This month I'm adding a new section to my column, called "Random Tidbits." This is a perfect spot for that important command, technique, or news bit that, well, doesn't really fit with the column but is important enough to talk about.

Jeff Victor has written a free and very useful new stat tool, zonestat. Jeff is a Sun employee who spends a lot of time in and around zones, even teaching tutorials about how to build and use them. One aspect of zones that was difficult to grapple with was the performance of the zones. Read Jeff's blog for more details and to download information about this very useful tool [13].

Solaris 10 11/08 (a.k.a. Update 6) shipped in November. The biggest news is a feature that I've been waiting for, for a very long time: ZFS boot / root. ZFS can now be the root file system on both x86 and SPARC systems. With this change comes the power of ZFS (snapshots, clones, replication, checksumming, and practically infinite everything) for the root disk and all system files. It is also very nice to be able to mirror the root disk with one command: `zpool attach rpool <rootdiskname> <newmirror diskname>`.

My company has started blogging about all things IT strategy. The topics there [14] will run the gamut from servers through storage, technologies, and trends. For example, my colleague Jesse St. Laurent has posted recent entries about HSM without headaches and the role of SSDs in storage. I've posted the slides I use when I teach my Solaris 10 tutorials. We hope you enjoy the blog and look forward to your comments.

A new edition, *Operating System Concepts,* 8th edition, the textbook I co-author with Avi Silberschatz and Greg Gagne, was published in the fall but I failed to mention it here. It is the best-selling undergraduate textbook on the topic. Have a look if operating systems theory and implementation interest you.

For an interesting real-world example of the use of analytics you should check out a blog entry by Brendan Gregg in which he shows that yelling at disk drives decreases their performance: http://blogs.sun.com/brendan/entry/unusual_disk_latency. This includes the YouTube video http://www.youtube.com/watch?v=tDacjrSCeq4.

RESOURCES

[1] http://en.wikipedia.org/wiki/DTrace.

[2] http://developers.sun.com/sunstudio/index.jsp.

[3] http://docs.sun.com/source/820-4221/index.html.

[4] http://developer.apple.com/technology/tools.html.

[5] http://opensolaris.org/os/project/dtrace-chime/.

[6] http://www.sun.com/storage/disk_systems/unified_storage/.

[7] http://www.sun.com/tryandbuy/.

[8] https://www.vmware.com/products/player/.

[9] https://www.vmware.com/products/fusion.

[10] http://www.c0t0d0s0.eu/permalink/A-walkthrough-to-the-Sun-Storage-Simulator-Part-1-Initial-Config.html.

[11] http://blogs.sun.com/bmc/resource/cec_analytics.pdf.

[12] http://blogs.sun.com/fishworks/.

[13] http://blogs.sun.com/JeffV/.

[14] http://ctistrategy.com/.

DAVE JOSEPHSEN

iVoyeur: message in a bottle

REPLACING EMAIL WARNINGS WITH SMS

David Josephsen is the author of *Building a Monitoring Infrastructure with Nagios* (Prentice Hall PTR, 2007) and Senior Systems Engineer at DBG, Inc., where he maintains a gaggle of geographically dispersed server farms. He won LISA '04's Best Paper Award for his co-authored work on spam mitigation, and he donates his spare time to the SourceMage GNU Linux Project.

dave-usenix@skeptech.org

WE FINALLY DID IT. DEGREE BY TINY degree, feature by "unsafe" feature, we killed email. This perhaps comes as no surprise to you, but I arrived at the realization only recently. Somewhere between SPF, DKIM, and the four-millionth RBL a critical mass was reached. It might be that one too many of the engineers who believed in the robustness principle [1] is now retired or dead, or perhaps it's the result of one too many overzealous vendors readjusting the corporate perception of "normal"; it might just be a question of scale. Whatever the reason, there's no question that today the onus is on you to convince your recipients' hostile, unwilling MX that your message is worth delivering, and attempting to do so makes you a spammer. These days any means justifies the claim of a little less spam.

Sending too many mails or not enough; having SPF or not; using domain keys, DKIM, both, or none; having reverse lookups; using text or HTML: all of these are indicators of spam and all of them aren't; there is nothing you can do in 2008 to appear to be a legitimate sender to every recipient. That after decades of flawless operation email has finally become an untrustworthy means of communication is as shocking as the fact that the overtly lazy, idiotic, wholesale destruction of SMTP was carried out in the name of "security"—by us, no less—in a vain attempt to outsmart Viagra peddlers one lazy quick fix at a time. Today the only way you can reliably get your messages delivered is to personally know the sysadmin at your recipients' organizations and get on their white-lists. There are secret invitation-only mailing lists for this purpose. You know who you are.

So what do you do when the quick fix of the week decides your monitoring server is questionable and you can no longer deliver email alerts to your pager or that of your peers? Move to a new messaging protocol? Maybe one such as SMS, that is younger and not yet broken? It'll work for a while, and hopefully something will be there to replace it when we break it too. You could move backward to an older, more trustworthy system such as the PSTN, a retro yet dependable option. A third option might be to use both, a gsm/gprs modem for SMS with the PSTN used as a backup, for example.

In this article we'll explore the third option, using a Nokia cell phone tethered to our Nagios server with a serial cable to send SMS, and alternately calling people on the phone with Asterisk if our Nokia is not up to the task.

I chose the Nokia 6170 for my own SMS implementation because it is readily available for about $100, is compatible with several U.S. carriers (we use T-Mobile), and has excellent compatibility with gnokii [2], the software we'll be using to interact with the phone. You'll also need a Nokia USB DKU-2 cable to connect the phone to the Nagios server. Beware of after-market cables, for they sometimes don't work as expected, and spend the extra $5 for the Nokia cable. I got mine on ebay for $20.

gnokii is a user-space driver and tool suite for communicating with mobile phones. It supports the usual free UNIX OSes and is scriptable via the command line and via a C library called libgnokii, for which the usual scripting language wrappers exist. Originally the intent of the gnokii authors was to operate with Nokia phones, but the tool suite can be made to work with any AT-compatible phone using serial, USB, IRDA, or Bluetooth. I can't speak to how hard this is, since I took the easy road and just got a Nokia, but the Wiki [3] lists a few non-Nokia phones that folks have gotten to work.

Gnokii supports a litany of features including getting/putting calendar events, editing phone-book entries, and dialing voice calls (useful for pranks and, I imagine, for war-dialing). This article will only use the `--send-sms` feature, although gnokii is capable of receiving SMS as well. Gnokii installs with the typical automake commands (`configure/make/make install`) and requires libusb for USB support and bluez for bluetooth. When gnokii starts up it checks the current user's home directory for .gnokiirc and then /etc/gnokiirc. You can specify a custom config file with the `--config` option.

My configuration file for the 6170 looks like this:

```
model = 6170
port = /dev/ttyS0
connection = dku2libusb
```

Once gnokii is installed and the phone is connected to the USB port of the Nagios server, issuing a `gnokii --identify` command should return some information on the phone. If it doesn't, adding `debug = on` to the config file might print some helpful error messages.

Integration with Nagios, as you might guess, involves defining a notification command. Mine is this:

```
define command{
   command_name  notify-by-sms
   command_line  /usr/bin/printf "%b" "`echo $NOTIFICATIONTYPE$
| /usr/bin/cut -c '1-3'`: $HOSTNAME$/$SERVICEDESC$ $SERVICESTATE$"
| /usr/bin/gnokii --sendsms $CONTACTPAGER$ 2>&1 | /usr/bin/logger -t
Nagios -p local4.info
}
```

That command might be a tad difficult to parse because of the nested system call to echo. This is intended to abbreviate the value of Nagios's `$NOTIFICATIONTYPE$` macro (either `PROBLEM` or `RECOVERY`) to a three-letter word (PRO or REC). The message also doesn't contain many of the details you might expect in a Nagios notification, such as the date or plug-in output, because the notification is designed to always fit within the 140 bytes allowed in an SMS message. The logger command at the end is intended to catch any error output from gnokii and send it to syslog.

Like any well-behaved UNIX program, gnokii exits 0 if everything went OK. This means that instead of piping to logger after gnokii, we could use a logical or operator (||) to launch a different command if gnokii is unsuccessful. This is a good place to put our Asterisk script.

Asterisk, as you probably know, is an open source PBX system. It is packed to the brim with features and is the subject of at least one article in just about every issue of *;login:* in recent memory. Asterisk is so featureful, in fact, that it feels silly to be using it for something as small as Nagios notifications, but it works excellently in this regard and is especially worth thinking about if you already have an Asterisk implementation of some kind.

The general strategy here is to use the festival-lite text-to-speech engine to create an error report, and pass this to Asterisk, which will call people and recite it to them over the phone. To do this you need an existing Asterisk system, or a telephony card of some type in the Nagios box. We use the TDM410 from Digium [4]. Installing Asterisk is a snap; I recommend using the packages from your distro, as several drivers need to be built and there are kernel dependencies involved.

Asterisk is normally a beast to configure, but in this context there isn't much to do. Normally you'd spend a lot of time configuring dialplans in extensions.conf, but since nobody will be calling this Asterisk server, all there really is to do is set up your hardware. For the TDM410 this means editing zapata.conf. The relevant section of mine looks like this:

```
context=incoming
signalling=fxs_ks
include => [default]
channel => 2
```

The easiest way to make Asterisk call people from the shell is to use the call files interface. Simply create a text file with the relevant data and drop it in /var/spool/asterisk/outgoing: Asterisk will immediately make the phone call. Here's what a typical call file looks like:

```
Channel: Zap/g2/15558675309
WaitTime: 15
Application: Playback()
Data: /var/spool/nagios/alerts/tmp.ihbgAO3751.gsm
```

I tend to use shorter wait times than the default (45 seconds) so that voicemail doesn't have a chance to answer. Reliably leaving voicemail gets more complicated, so I prefer to just not deal with it. If I have a missed call from the Nagios box, I'll log in and see what's up. The data file is created by simply echoing the alert text to festival, like so:

```
echo "Nagios ${NOTIFICATIONTYPE}, Host ${HOSTNAME}, Service
${SERVICEDESC} is in state ${SERVICESTATE}" | flite -o somefile.wav
```

We can refer to these variables inside a shell script called by the notify-by-sms Nagios command because any script called by Nagios is given global variables that correspond to Nagios macros relating to the service outage. So many times I see people using macros as arguments to shell scripts called from within Nagios notification commands when it's always unnecessary; even many of the FAQ answers on nagios.org do this. Dear Internet: You don't have to mess with argument passing; your script already has all the macros as global vars. I digress. Asterisk can't read wav files, so we need to convert it to GSM, like so:

```
sox somefile.wav -r 8000 somefile.gsm resample -q1
```

When we create the call file, we need a way to map some Nagios macro to a phone number. Since we're using Asterisk to backup SMS in this example, the $CONTACTPAGER$ macro will work, but if you were backing up email you'd either need a lookup table of some type or a custom notification macro that specifies our contact's phone number. Nagios has for some time supported addressX macros that are perfect for this; just make sure address4, for example, always has a phone number, and you're all set.

That gives you pretty much all the pieces you need. A simple shell script can then be written that is called with a logical or in the notification command, like so:

```
command_line   /usr/bin/printf "%b" "`echo $NOTIFICATIONTYPE$ | /usr/bin/
cut -c '1-3'`: $HOSTNAME$/$SERVICEDESC$ $SERVICESTATE$" | /usr/bin/
gnokii --sendsms $CONTACTPAGER$ || /usr/local/nagios/bin/contact_by
_phone.sh
```

Now if the SMS message fails, Nagios will call the contact_by_phone shell script, which will use various Nagios macros to create a GSM audio message and an Asterisk call file and place the call file into /var/spool/asterisk/outgoing.

If you want to get fancy, you could specify a dialplan context instead of an application name in the call file, conceivably allowing the person being called to do things such as "press 1 to acknowledge this alert," "press 2 to run event-handler X," etc. Asterisk has some pretty cool remote management potential in this regard, which is perhaps fodder for a future article. If you're currently doing any Asterisk/Nagios integration stuff, I'd love to hear about it. Feel free to drop me an email (you know, if anybody still uses email; my MX promises to be nice) or to post a comment on my blog, www.skeptech.org.

Take it easy.

REFERENCES

[1] Robustness principle: The ancient fallacy that one should be liberal in what one accepts and conservative in what one sends.

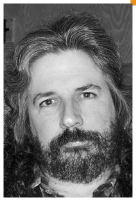
[2] http://www.gnokii.org.

[3] http://wiki.gnokii.org/index.php/Config.

[4] http://www.digium.com/en/products/analog/tdm410.php.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is an information security geek biding his time until that genius grant finally comes through.

rgferrell@gmail.com

ONE MAJOR STEPPING STONE ALONG the Möbius path to Compleat UNIX Systems Administrator is mastering command-line editors. I've steadfastly avoided getting involved in the vi versus Emacs holy wars, since I never saw any real reason to learn Emacs and thus am quite comfortable in my raging ignorance of this tool. Even though I've been using vi for well over twenty-five years now, I still on occasion stumble over options and commands I've never before encountered (or, more accurately, never found any utility in memorizing). In honor of my 51st birthday, which will be nothing but an Islay-clouded memory by the time this column goes to press, I have dug out some ancient notes on an early version of vi known as vi (not), translated them from Pig-Sanskrit, and here elucidate them in the familiar form of a man page for your edification and entertainment.

VI (NOT) (-1) VI (NOT) (-1)

NAME

vi (not), vi bother, *%#$!

DESCRIPTION

Vi (not) is a screen-disoriented text "editor" intended for use by people who think GUIs can be found buried on beaches in New England or are candy products popular with the toddler demographic. If you were not alive during the Vietnam conflict and you still insist on using *vi (not)*, you're either a poser or a TPL (Typophilic Pseudo-Luddite). Your brain does not work the way everyone else's does, and so you must be isolated for the good of society. Fortunately, that problem usually resolves itself.

The following command-line options are available (may not be functional in months with an "r" or if your compiler can't handle ALGOL 68):

-w Execute *init 0* a random number of seconds after startup. Part of a wisely abandoned early Windows simulator.

-g Start editing in Groucho mode. Quacks every time the secret Key of the Day (KOTD) is pressed.

-R	Copy a random system file into the screen buffer on startup. Garnish with celery and serve on a chilled plate.
-|n	Pipe every *nth* word to *smsclient*.
-s	Start editing in *scavenger hunt* mode, where the edited file is saved in a random directory and filename.
-?!!	Save the specified file to a location, or, if not available, save some file to a place specified that would have been available if the specified location had been available and not specified.
-S	Run with the ultra-secure edit option set, disallowing all access to everything. Forever.
-xxx	We sort of forgot what this one does. It's bound to be interesting, though.
-p	Start editing in the previously opened document.
-n	Start editing in a document you might want to edit in the future.
-F	Start editing in an actual functional editor (not implemented).

The commands to move around in the file, if you do manage to get one open, are:

1 + !	Move the cursor left one character.
2 + @	Move the cursor down one line.
3 + #	Move the cursor up one line.
+ + =	Move the cursor right one character.
%*$	Move the cursor over behind the fridge.

The commands to enter new text are:

}{	Append new text somewhere in the document.
<>	Insert new text, oh, about midway through the next paragraph.
(o)	Open a new line below the line the cursor is on. Watch YouTube. Rinse and repeat.
[@]	Open a new line above the line the cursor is on, then go read your email.

Note: If standard input is not a terminal, vi (not) will read commands from it regardless; however, it will mock the user mercilessly and use uucp to tell all its friends what a loser you are.

ENVIRONMENT VARIABLES

FONT
If set, vi (not) uses a font. Maximum flow rate is 42 gpm, unless the *-niagara* flag is set when compiling.

EXCUSE
A list of implausible reasons vi (not) didn't start up this time, either.

HOME
The user's home directory, where all the error messages, temp files, core dumps, and spurious gibberish accumulates.

LINES
An alleged professional American football team.

SHELL
Your ordnance of choice.

TERN
An aquatic bird known for diving after fish and pooping on piers.

See also
frags(100), smores(0), curses(666), kill(-9)

book reviews

ELIZABETH ZWICKY, WITH JASON DUSEK, SAM F. STOVER, AND RIK FARROW

THE CARTOON GUIDE TO STATISTICS

Larry Gonick and Woollcott Smith

Collins Reference, 2005. 223 pages.
ISBN 978-0-06-273102-9

HEAD FIRST STATISTICS

Dawn Griffiths

O'Reilly, 2008. 655 pages.
ISBN 978-0-596-52758-7

THE MANGA GUIDE TO STATISTICS

Shin Takahashi

No Starch, 2008. 211 pages.
ISBN 978-1-59327-189-3

Who would have thought that the graphic introduction to statistics was a genre? But in my search for appropriate ways to introduce system administrators to statistics, I came across no fewer than three books that are trying to help out with pictures.

Here's a quick reference to the three. *The Cartoon Guide* is a classic statistics book, embellished with lots of pictures. It covers a full statistics class, it's moderately easy to read, and there's fun history included. *Head First Statistics* is the best of the lot, fully using the power of pictures to cover most of the same territory (minus the history) much more palatably. *The Manga Guide* covers the least, but boy does it have that authentic manga flavor, which is pretty amusing.

The Cartoon Guide is the hardest going of the three. (Probably not coincidentally, it's also the oldest.) It's the only one that touches at all on history of statistics. If you're good with math and would like it minimally spiced up, this is the way to go.

The Manga Guide is ideal if you're disappointed by the lack of romance in your technical books. It does have a romantic subplot. A slightly creepy romantic subplot, if you ask me, since it revolves around a high school girl with a crush on one of her father's co-workers, but as manga goes, it's not bad. It would help if you had some familiarity with the visual conventions of manga. Again, it's not outrageous, but if you're still prone to be bewildered by characters who change size and shape to reflect the way they feel, you're going to spend as much time trying to puzzle out the illustrations as trying to puzzle out the statistics. It does flow the reader right through the content, and it has a handy guide to using Excel to do statistics, which none of the other books does. Its real weak point is a lack of coverage for quartiles and box-and-whisker plots; quartiles are extremely handy for working with a lot of non-normally distributed data that you see in computer systems.

Head First Statistics covers all that stuff, and it does it slowly enough that anybody with basic mathematical competence should be able to cope. It's a thorough, palatable introduction to statistics and would be my choice for most people.

THE DAEMON, THE GNU, AND THE PENGUIN

Peter H. Salus

Reed Media Services, 2008. 177 pages.
ISBN 978-0-9790342-3-7

This is a small history of free and open software. It is admiring without being sycophantic about any of the players, and it balances as much as possible among the often warring factions involved. I learned parts of the history that I didn't know, and I enjoyed the ride. This would be a great gift for somebody young who's just coming into the world of open source and would like a brief historical tour that doesn't have too many axes to grind.

THE BACK OF THE NAPKIN

Dan Roam

Portfolio, 2008. 272 pages.
ISBN 978-1-59184-199-9

Here's a book that doesn't even aim for fair and balanced. This is a passionate defense of the diagram as a way of thinking and talking about problems, with a set of clear procedures that should make it accessible to anybody (since drawing ability is not required). I enjoyed it greatly.

There are two situations at which this book takes particular aim. The first is selling an idea: trying to convince people that your solution is a good one. This is something we all do, whether or not we are salespeople, and if it's important to you, there's a great deal of insight here.

The second situation is creating solutions: getting your head around a pile of information and turning it into an understanding of the situation. The book offers some tools for doing that, alone or in a group.

I have two related misgivings. First, did I mention this is a passionate defense of a particular procedure? It's clearly laid out, and comprehensive, and probably not an exact fit for most people. I nodded my head all the way through, but I haven't actually tried it, because I need to figure out what parts I'm going to work into my process, and how I'm going to adapt it to my style. I'm not going to use it completely.

Second, the piece of it that I understood best is also the piece I liked least, which is never a good sign. As an illustration of why his diagrams are good for just about everything, the author talks about diagramming out how a software package works and says, "To improve security, we'd need to enhance protection around those areas where the most information enters and leaves the system." Given that the hypothetical product is said to be relatively secure to start with, this is staggeringly unlikely. Those are the most obviously security-related parts of the system, and somebody has probably already waved some sort of security wand at them. The weakest link is almost certainly somewhere else. Now, I don't expect the author to understand security engineering, but it's a good illustration of how drawing out the system only helps if you're bringing deep understanding to the surface or sorting out something you already know something about. It won't substitute for expertise in bringing you insights into unfamiliar territory.

ADOBE PHOTOSHOP CS4 ONE-ON-ONE

Deke McClelland

O'Reilly, 2008. 499 pages (includes DVD).
ISBN 978-0-596-52189-9

THE PHOTOSHOP CS4 COMPANION FOR PHOTOGRAPHERS

Derrick Story

O'Reilly, 2008. 177 pages.
ISBN 978-0-596-52193-6

Photoshop may seem like an odd topic for people who're used to mostly open solutions, but it's oddly familiar in many ways. Rob Pike is reputed to have said "UNIX is user friendly. It's just selective about who its friends are." Photoshop is much the same. Not only that, but it takes a Perl-like "let many flowers bloom" approach. I'm no Photoshop expert, but I know three different ways to fill an entire document with a pattern, and I'm not at all certain what's different about two of them. In the first video for *CS4 One-on-One*, Deke McClelland demonstrates at least five different ways to zoom in and out. Couple this with the documentation, which is, as the Australians say, "very ordinary" (i.e., terrible in all the usual ways), and you have a powerful but intimidating program. Early adventures in Photoshop tend to involve a lot of murmuring "Why did that happen?" and "Surely there's a way to do it. I wonder if it's on a panel somewhere?"

This of course means that there are 4 million Photoshop books available. I'm looking at CS4 books because CS4 just came out, and this therefore cuts down the crowd considerably. It's also interesting because CS4 introduces some serious changes to how Photoshop is best used.

The CS4 Companion is a nice, succinct introduction to Photoshop for photographers, and it takes full advantage of the new features in CS4. It's a great starting point if you're interested in using Photoshop for seriously managing photos. I have a few quibbles—photographic workflows are pretty idiosyncratic, and I disagree about some of them (e.g., why on earth would you reformat your memory card all the time?)—but mostly I find it a direct and useful reference that pushes a CS4-based process. This is important, and it's going to be a problem for most CS4 books based on CS3 books, because the best way of doing things has changed pretty dramatically. All the things you might be used to are still possible, but they're generally no longer ideal.

The main issue here, if you're familiar with Photoshop, has to do with adjustments, for things such as Levels or Curves. It used to be that the real hotshots used these with masks, which involved interesting tricks, and everybody else used the History brush. Now there are adjustment layers, which make it easy to do nondestructive changes you can always back out of without damaging your image, and just using Levels or Curves is wantonly destroying data for no good reason.

These changes are an issue for *One-on-One*, which does cover all the cool new features but introduces more traditional ways of working first. This makes some sense, because the book is clearly intended to teach people who want to be graphic artists and need to be able to integrate with other Photoshop users. But if you're learning for personal use, you're going to learn techniques and then discard them later.

As a tutorial for people who're committed to learning Photoshop, I liked *One-on-One* a lot. I think it uses video well and I like the way it works through examples to show you when techniques are useful. (The videos teach things that are hard to convey in text, but they're not essential to the experience.) It is really a class in a book, though. It's not for you if you want to pick up a technique at a time. For instance, I knew that CS4 had added a new automatic layering trick I wanted to try, and I looked at both books to figure out how to do it. Both books do cover it, but *The CS4 Companion* gave me a brief recipe on the most straightforward way of using it, while *One-on-One* gave me a sidebar about it, which did explain how to do it and showed a really interesting use of it, but was much harder for me to interpret in the moment.

MAVEN: THE DEFINITIVE GUIDE

Sonatype Company

September 2008. 468 pages.
ISBN 978-0-596-51733-5

REVIEWED BY JASON DUSEK

The book introduces Maven, its operation, its core configuration, a wealth of plug-ins, and the elements of customization. Maven, a build and release tool, models projects with the Project Object Model (POM). The copious examples are enough to remind one of how XML's verbosity can be made up for by its readability—for the POM is XML all the way down.

More than offering a view into Maven, this book offers a glimpse into the degree to which build and release management processes have been automated, codified, and standardized (which certainly came as a pleasant surprise to this reviewer, who still does all this stuff with Make). Maven offers a rich collection of tools for not only build management but dependency visualization, plug-in extension, and, to no small degree, Web publishing. The number of markup languages and template engines supported by Maven is no small credit to the project, and it is all to the good that I need only browse through a few pages of this book to find out about them.

One avowed goal of the book is to make whole the uneven documentation of Maven and its plug-ins. The authors are nothing if not thorough in this regard. Nearly every other page offers a figure, example code, or sample configuration. Where a model relationship is mentioned, a diagram is offered. The authors do not even assume knowledge of octal permissions—a link is offered to clarify.

The richness of Maven and more generally of Java technology perhaps makes a fault of the authors' diligence—one cannot escape a discussion of Inversion of Control containers, the multitude of JVM scripting languages with which one may extend Maven, or integration with Eclipse and the attendant "materialization." A very small section of the book is devoted to walk-throughs of build specification; the vast majority is given to concepts, features, and customization, in short vignettes unrelated to one another. This is more of a "how to do it to Maven" book than it is a "how to do it with Maven" book.

In consequence, the reviewer is left with a disappointment: the one thing he really wanted to know, "how to build something that is not Java with Maven," is handled by a plug-in not covered in this book.

DESIGN CONCEPTS IN PROGRAMMING LANGUAGES

Franklyn A. Turbak and David K. Gifford

MIT Press. 2008. 1200 pages.
ISBN 978-0262201759

REVIEWED BY JASON DUSEK

In comparison to texts such as *Structure and Interpretation of Computer Programs* or *Concepts, Techniques, and Models of Computer Programming*, this text is much more mathematically oriented, guided by an interest in language design and specification as opposed to program design and specification. As such, it is an encyclopedic reference of languages and language features as opposed to a deep exploration of one particular language and the means of programming with it.

For those of us who have nurtured an abiding interest in programming language theory with research papers and blog posts, this book offers a thorough and disciplined approach to terminology as well as a way to round out the inevitably lopsided knowledge of the auto-didact. The expository text and numerous citations place each development in context, historically as well as functionally.

Each design concept is accompanied by a mini-language, all of which are presented in a uniform, Scheme-like syntax. This uniformity dovetails with the small scope of the languages and makes translation from the grammar (provided in full for each language) to the concrete language a simple exercise. The text covers a variety of paradigms, including stack-based programming, purely functional programming with static and dynamic types, type inference, effect systems, and object-oriented programming languages. A listing of languages and their feature sets is provided at the end of the text.

Programming content is actually pretty light—most (all?) of the numerous exercises can be worked with paper and pencil. Some time is given to garbage collection and compilation, but efficient implementation is not a focus of this text. The book's Web site provides links to programming problems from an associated MIT course.

This book is more edifying than enabling for most of us: Rarely is the working programmer called upon to implement a programming language. Just the same, mini-languages find application surprisingly often; the material presented is food for thought the next time you are deciding just how to arrange a batch job format or a simple template system. In recent years, some "esoteric" languages—Factor, Haskell, and Erlang come to mind—have become surprisingly usable; this text provides an excellent basis for appreciating their guiding design ideas.

THE CRAFT OF SYSTEM SECURITY

Sean Smith and John Marchesini

Addison-Wesley Professional. 2007. 592 pages.
ISBN 978-0321434838

REVIEWED BY SAM STOVER

I'm not all that sure where this book came from, but I wish it had been around about 10 years ago. What started out as a working text for a college course has ended up as a solid intro to a large number of topics in the information security field. I really can't recommend it enough for the technical guy or gal who wants to learn more about security.

The book is broken down into five parts: History, Security and the Modern Computing Landscape, Building Blocks for Secure Systems, Applications, and Emerging Tools.

The three chapters in the History section discuss, well, history. I breezed over this, to be honest, but I think the authors do a reasonably good job of introducing important, albeit historical, information in an academic yet readable manner. That was one thing that really struck me throughout the entire book: the adherence to scientific principles when discussing technology. I've found that sometimes that style of writing doesn't play well with the "craft" of security, but these authors do it well.

Security and the Modern Computing Landscape consists of three chapters, which tend to be the meat of most other security books. OS Security touches on process isolation, filesystem permissions, and common attack vectors such as keyloggers and rootkits. Network Security follows the same style but deals with things such as the TCP/IP stack, different protocols on the stack, and common attacks for those protocols. Implementation follows suit in the application arena, dealing with buffer overflows, fuzzing, malware, and other tools and attacks relevant to applications.

Building Blocks for Secure Systems starts out with Using Cryptography, followed by Subverting Cryptography. I think these two chapters alone are worth the price of admission—the pros and cons, as well as common pitfalls of the different cryptographic implementations, should be required reading for anyone deploying crypto as a security measure. It's all too easy to assume that encryption equals security, and these two chapters remove a lot of that mystique. The remaining chapters in this section deal with Authentication, Public Key Infrastructure and Standards, Compliance, and Testing. Again, this is more great, highly applicable reading for anyone doing any kind of secure system building.

The Applications section focuses on Web security, common office tools, digital cash, and securing different types of information. These topics aren't typically my cup of tea, but, as with all topics in the book, they are covered in depth, yet remain very readable. Emerging Tools goes into the weeds a bit, but it's still great reading, covering things such as hardware-based security, virtualization, armored CPUs, RFID, and even a little AI for good measure. Some of it is directly relevant; some of it is a bit more theoretical—but don't think of theoretical as a negative thing. Theories are good. The book wraps up with some "take-home" thoughts that, again, head out into the weeds, but if you're a math geek, it's all you. (They actually discuss some quantum mechanics as a means to break RSA.)

Overall, the book was well written, informative, pertinent to contemporary security issues, and just a solid all-around reference. I have no problem recommending it to anyone who wants to bone up on any of the topics discussed. If there's one thing I would wish, it would be that more security books were written with the attention to detail and topic coverage that this one has. Happy reading.

STATISTICS IN A NUTSHELL

Sarah Boslaugh and Paul Andrew Watters

O'Reilly, 2008. 452 pages.
ISBN 978-0-596-51049-7

REVIEWED BY RIK FARROW

I got this book because I had recently been faced with just how much I had forgotten about statistics. I had taken more than one statistics class in college, but I found myself at a loss when attempting to interpret the statistics I encountered in CS papers.

Although the subtitle of this book is *A Desktop Quick Reference*, this nutshell book is really a textbook. It is written not just for learning about statistics but also for researchers who need to design their model data so that it can be properly analyzed. The target audience is really not those of us in CS, but practitioners in other sciences. Having said that, I have found that I will be using this book as my desktop reference, having read enough of it that I am immediately interested in.

I also got *The Manga Guide to Statistics*, which Elizabeth reviewed, but can't fairly compare them. The difference between the two is so large as to be incomparable. Whereas *The Manga Guide* does present equations, for example, *Nutshell* explains where the equations come from. I found that I was uncomfortable with complex equations that just appear in comic panels. When the *Nutshell* authors introduce equations, they do so in a context that makes the equations appear understandable. I found the same equations much less impenetrable—in fact, less complex—with this presentation.

The *Nutshell* book authors carefully explain the usage of different statistical measures, as well as potential pitfalls. Most of this in *The Manga Guide* gets covered in one chapter, but the *Nutshell* book's authors cover a complete course in statistics geared to research in crystal-clear text and in depth. There are also chapters on research design and critiquing statistics presented by others. Chapters 17–19 cover business, medical, epidemiological, educational, and psychological statistics—a bit outside the scope of my own interests.

If you want a quick romp in statistics, perhaps the graphics versions we reviewed will provide you with the background you need to read papers. Perhaps. If you want to write your own papers and include statistical analysis of your data to prove your assertions, *Statistics in a Nutshell* is the book for you.

GETTING STARTED WITH ARDUINO

Massimo Banzi

O'Reilly, 2008. 117 pages.
ISBN 978-0-596-15551-3

REVIEWED BY RIK FARROW

You may have heard of Arduino, an open-source hardware project that uses a simple microprocessor, such as the Atmega168, and a handful of other parts that can be easily programmed. I had watched someone playing with an Arduino designed to be included in clothing (think Burning Man), so I wanted to check this out. O'Reilly sent me this book, and I ordered what turned out to be a slightly older model Arduino (the Diecimila) but it worked fine with this book.

The book is really aimed at nonprogrammers who want to play with physical computing. The Arduino allows you to do rapid prototyping, where the microprocessor coupled with the development environment does most of the work. Although you can accomplish some simple projects with nothing but your computer, a USB cable, the Arduino, and some simple electronics, getting a solderless breadboard for assembling resistors, photodetectors, other sensors, and LEDs is much handier.

If you are already a programmer, the online documentation may be all you need, but if you have been hesitating to leap into hardware, this book and Arduino hardware provide a painless way of doing so. Having the book is what pushed me over the edge, enabling me to discover just how easy it was to get started.

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, networks, 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/.

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: www.usenix.org/membership/specialdisc.html.

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

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

USENIX BOARD OF DIRECTORS

THE INTERNATIONAL OLYMPIAD ON INFORMATICS

Rob Kolstad, USACO Head Coach

Each year, the USENIX-sponsored USA Computing Olympiad identifies and trains our country's top four pre-college competition programmers in anticipation of the world championships: The International Olympiad on Informatics.

The 2008 Olympiad was conducted August 16–23 just outside of Cairo, Egypt, in six-year-old Mubarak Educational City. MEC is a walled compound of perhaps 50 acres capable of housing 1,000 guests (usually teachers for in-service training). It is surrounded by desert and located about an hour's drive from the outskirts of Cairo. While the blisteringly hot weather would wilt normal flora, a team of caretakers tends and waters beautiful foliage throughout the year, providing an attractive, if extremely warm (100+°F), outdoor environment.

Our team of four (see the photograph below) arrived on August 16 for the week-long event. Rising MIT freshman David Benjamin hails from Indiana; fellow MIT-student-to-be Jacob Steinhardt has graduated from Thomas Jefferson High School of Science and Technology in Virginia, hailed by some as America's #1 high school. They were joined by rising juniors Brian Hamrick, also from TJHSST, and Neil Wu from Louisiana.

From left: Jacob Steinhardt (Silver medalist), Brian Hamrick (Silver), Neal Wu (Gold), David Benjamin (Gold)

The student excursion to the Pyramids was surely the recreational highlight of the trip. Buses moved all 300 participants (and an additional 300 coaches/chaperones) to the Great Pyramids (and Sphinx) on the edge of Cairo. Many students were able to journey to the center burial chamber of one of the pyramids, a medium-sized room with a temperature of perhaps 110°F. Travel tip for visiting Egypt: choose one of the eleven months cooler than August.

Coaches and visitors were afforded an extra excursion to one of Cairo's bustling bazaars. The bazaar was enormous, with buildings and covered pedestrian-only walkways over an area of approximately half a square mile. In fact, it was large enough that I became lost after an unfortunate separation from my more navigation-savvy group. (A real-world application of a maze-solving algorithm returned me 20 minutes later to the bus.)

Two five-hour contests highlighted the Olympiad competition. Each contest included three extremely challenging tasks. Contestants coded solutions in C, C++, or Pascal, which were then submitted to a grading system that ran each task with many sets of test data to see if the program could calculate the proper answer within the time and memory constraints. See below for the hardest task; only four students received 60% or more credit on this one.

The results were announced at a gala marathon awards ceremony, followed by a banquet of sorts. The top 1/12 of the students (24 this year) earn Gold medals; the next 2/12 (1/6) earn Silver models; the subsequent 3/12 (1/4) earn Bronze models.

While official country results are not kept, unofficial tallies abound. China continued its winning streak but with slightly less domination than the past: three Gold medals and one Silver for the best overall performance. Poland was second (or tied for first, depending on your point of view), with three Gold medals and one Silver, but with slightly lower overall places and scores. The United States came in third, with two Gold medals and two Silver. Russia also earned two Gold and two Silver medals, but received lower places/scores than the U.S.

Individually, our Gold medal winners performed extremely well, with David Benjamin placing 8th overall (in the world!) and Neal Wu placing 10th. Jacob Steinhardt's 29th place put him just six spots out of the running for Gold. At 37th place, Brian Hamrick was not far behind Jacob.

The USA Computing Olympiad continues to hold monthly competitions for three divisions of competitive programmers. This year's contests have each drawn 1,000 or more competitors from more than 60 countries. Thanks to USENIX and its membership for continuing sponsorship of this great program.

FISH: THE GREATEST CHALLENGE

The task below is shown in full (just as presented to the students) at http://ace.delos.com/ioi2008-1.pdf.

It was told by Scheherazade that far away, in the middle of the desert, there is a lake. Originally this lake had F fish in it. K different kinds of gemstones were chosen among the most valuable on Earth, and to each of the F fish exactly one gem was given for it to swallow. Since F <= K, two or more fish might swallow gems of the same kind.

As time went by, some fish ate some of the other fish. One fish can eat another if and only if it is at least twice as long (fish A can eat fish B if and only if LA >= 2 * LB). There is no rule as to when a fish decides to eat. One fish might decide to eat several smaller fish one after another, while some fish may decide not to eat any fish, even if they can. When a fish eats a smaller one, its length doesn't change, but the gems in the stomach of the smaller fish end up undamaged in the stomach of the larger fish.

Scheherazade has said that if you are able to find the lake, you will be allowed to take out one fish and keep all the gems in its stomach for yourself. You are willing to try your luck, but before you head out on the long journey, you want to know how many different combinations of gems you could obtain by catching a single fish.

Write a program that, given the length of each fish and the kind of gemstone originally swallowed by each fish, finds the number of different combinations of gems that can end up in the stomach of any fish, modulo some given integer M. A combination is defined only by the number of gems from each of the K kinds. There is no notion of order between gems, and any two gems of the same kind are indistinguishable.

Constraints:

1 <= F <= 500,000
1 <= K <= F
2 <= M <= 30,000
1 <= LX <= 1,000,000,000
3 seconds on modern PC
64MB memory limit

GOT A MINUTE?

Jane-Ellen Long,
Director of IS & Production

USENIX and SAGE offer oh so many ways for you to contribute to the community and get your name in lights. Well, OK, in print or on the Web, at least. Here are a few options to get you started.

GET THE WORD OUT: WRITE FOR ;LOGIN:

Air a controversial opinion, share a great tool or technique, bare the dark underside of a recent hard-won tech battle. How? Write a *;login:* article or proposal.

Not enough time for that? How about writing the occasional book review, or reporting on conference or workshop sessions?

Send your article, proposal, or other offer to login@usenix.org.

HELP YOUR COMRADES

Can you (perhaps with a little help from your friends) supply a quickstart guide to a sysadmin task? SAGE is always on the lookout for new authors and subjects. Whether you have an idea for a whole Short Topics booklet (see http://www.sage.org/pubs/short_topics.html for the current list) or you just want to distill a recent experience into a short White Paper, suggestions@sage.org would like to hear from you.

CALLING ALL ACADEMICS

USENIX Campus Representatives act as our eyes and ears on campus, spreading the word about USENIX, helping their school's students get USENIX travel grants to attend events, and making our CFPs and publications available on campus. What's in it for you? Plenty. See http://www.usenix.org/students/outreach.html for the details.

GOT TOOLS? (OR BOOKS)

The Sysadmin Toolbox—http://www.sage.org/field/toolbox.html—and the extensive SAGE Recommended Reading lists— http://www.sage.org/books/books.html—always need updating. We welcome ideas for new sections, as well as comments about individual books. Contact suggestions@sage.org.

LOOKING FOR LOCALS

Both the USENIX and the SAGE Web sites maintain lists of user groups. Names, meeting information, and contact information for local, national, and international groups will be added promptly upon receipt by production@usenix.org. News of defunct organizations is also solicited.

USENIX DISCUSSION LIST

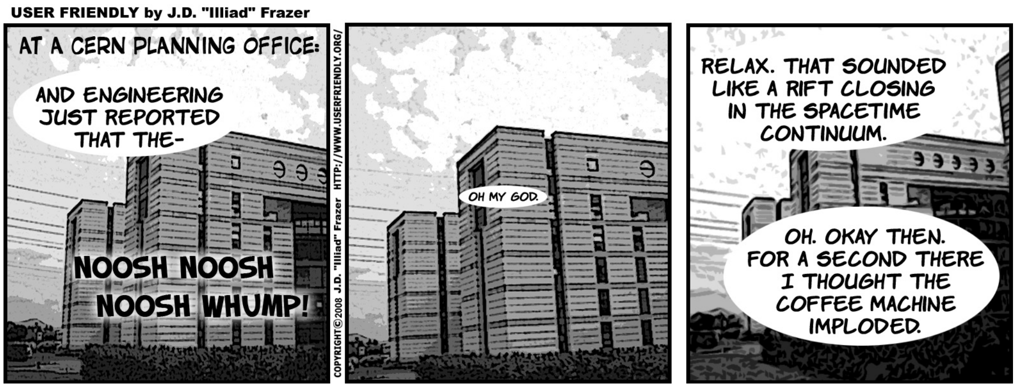
Anne Dickison, Director of Marketing

Want to talk about a *;login:* article? Have suggestions for future topics? Share your thoughts on the new USENIX discussion list. Sign up at http://lists.usenix.org/mailman/listinfo/usenix-discuss.

NEW VIDEOS AVAILABLE!

Anne Dickison, Director of Marketing

Did you miss LISA? Want to see the USENIX Security '08 keynote? Videos from 2008 conferences are now available at http://www.usenix.org/publications/multimedia.


USER FRIENDLY by J.D. "Illiad" Frazer
AT A CERN PLANNING OFFICE:
AND ENGINEERING JUST REPORTED THAT THE-
NOOSH NOOSH NOOSH WHUMP!
COPYRIGHT©2008 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
OH MY GOD.
RELAX. THAT SOUNDED LIKE A RIFT CLOSING IN THE SPACETIME CONTINUUM.
OH. OKAY THEN. FOR A SECOND THERE I THOUGHT THE COFFEE MACHINE IMPLODED.

conference reports

THANKS TO OUR SUMMARIZERS

LISA ’08: 22nd Large Installation System Administration Conference 59

Ben Allen
Kyrre Begnum
Alex Boster
Leah Cardaci
Marc Chiarini
Andrew Hoblitzell
Qi Liao
Rowan Littell
Will Nowak
Patrick Ntawuyamara
David Plonka
Matthew Sacks
Andrew Seely
Josh Simon

CHIMIT ’08: Symposium on Computer Human Interaction for the Management of Information Technology. 82

Æleen Frisch
Eser Kandogan

The 8th Privacy Enhancing Technologies Symposium 86

Rae Harbird

LISA ’08: 22nd Large Installation System Administration Conference

San Diego, CA
November 9–14, 2008

Summarized by Rik Farrow

Mario Obejas led off with thanks to the program committee members and USENIX staff for putting together another successful LISA. Then the SAGE award was given to the SAMBA group for its work on interoperability. SAMBA Team member (and USENIX Board Member) Jerry Carter accepted the award. The Chuck Yerkes award was given to Dustin Puryear for his helpful posts to sage-members.

The Best Student Paper award went to Xiaoning Ding of Ohio State University, Hai Huang, Yaoping Ruan, and Anees Shaikh of IBM T.J. Watson Research Center, and Xiaodong Zhang of The Ohio State University for “Automatic Software Fault Diagnosis by Exploiting Application Signatures.” The Best Paper award went to Qi Liao, Andrew Blaich, Aaron Striegel, and Douglas Thain of the University of Notre Dame for “ENAVis: Enterprise Network Activities Visualization.”

KEYNOTE ADDRESS

- ***Implementing Intellipedia Within a “Need to Know” Culture***
  Sean Dennehy, Chief of Intellipedia Development, Directorate of Intelligence, U.S. Central Intelligence Agency

Summarized by Andrew Hoblitzell (ahoblitz@cs.iupui.edu)

Sean Dennehy discussed technical and cultural challenges being introduced by the introduction of Web 2.0 tools in the United States intelligence community. Dennehy traced the start of the movement to Dr. Calvin Andrus’s publication of his 2004 paper “The Wiki and the Blog: Toward a Complex Adaptive Intelligence Community” and showed the development of various Web 2.0 tools in the intelligence community up to the current day. The tools are used by intelligence analysts from 16 intelligence agencies, as well as other parts of government.

Dennehy began his presentation by showing a document from the Office of Strategic Services, the predecessor to the Central Intelligence Agency. The document described many of the characteristics of a typical bureaucracy, such as deferring decisions because of worries about management issues, conflicts about the specific wording of things, etc. Dennehy revealed that this document was produced by the Office of Strategic Services as a guide to sabotaging an organization. Dennehy proposed that Web 2.0 tools were a solution to these common problems.

In the intelligence community, information sharing is especially difficult, because information is typically available on a “need to know” basis. The information is classified to different levels, usually Controlled Unclassified Information, Secret, and Top Secret. Information may also be "compartmentalized" to further restrict its access. Two U.S. networks for transferring this information are SIPRNet for secret information and JWICS for top secret information. In contradiction to the common tenets of the intelligence community, Dennehy said, Intellipedia would aim information at the broadest audience possible. In this spirit, Dennehy said that the approach has been shared with allies and the media.

Dennehy pointed out a number of differences between Intellipedia and public Web 2.0 tools such as Wikipedia. Dennehy pointed out that although edits in Wikipedia may be anonymous, every edit to Intellipedia must be attributable. Other differences pointed out by Dennehy included that Intellipedia need not be encyclopedic and that Intellipedia only uses attributable point of view instead of Wikipedia's neutral point of view (NPOV). Dennehy said that these differences allowed Intellipedia to encourage professionalism, prevent anonymous edits, and encourage collaborative disagreement. Dennehy said the government's Web 2.0 tool set system had been further enhanced with a YouTube-like video channel, a Flickr-like photo-sharing feature, content tagging, blogs, RSS feeds, and many other Web 2.0 tools.

Dennehy noted that there would never be a "one stop shop" for the intelligence community and that Intellipedia was still in the early stages of development. Dennehy said that, despite this, Intellipedia would act as a powerful way to connect people. He said that people acting selfishly could in the end help each other and that Intellipedia would help connect analysts to information that they might not have been able to find otherwise.

Dennehy said that intelligence workers had started using Intellipedia as a replacement for their email and telephone conversations and are sharing it with each other in their personal blog postings. Dennehy said that Intellipedia would catch on because it has strong support from the top, including from some in the Joint Chiefs of Staff, but that it would ultimately have to become a self-sufficient grassroots movement kept active from all levels of the intelligence community.

In response to questions from the audience, Dennehy stated that old or dead data could automatically be tagged in the system after a given timeframe by using bots or similar technology. Dennehy also stated in response to a question from Dan Klein that he has seen more interagency cooperation as a result of the wikis. He noted that the intelligence community would have a strong interest in utilizing and adapting software that is currently used for Wikipedia and other Web 2.0 tools.

THINK ABOUT IT (META-ADMIN AND THEORY)

Summarized by Alex Boster (aboster@machineepsilon.com)

- ***Designing Tools for System Administrators: An Empirical Test of the Integrated User Satisfaction Model***
  Nicole F. Velasquez, Suzanne Weisband, and Alexandra Durcikova, University of Arizona

Nicole Velasquez explained that the motivation for this work came from a usability study at IBM done on system administration tools that yielded negative feedback from their target users. The conclusion of that work was that the tools were "built for grandma," not system administrators. A new, hybrid theoretical model, the Integrated User Satisfaction (IUS) model, was developed, combining the strengths of TAM and IS Success, older usability models. In this study, a Web-based survey was taken of system administrators, without respect to the specific type of administration they performed, using verified methodologies. Then this new usability model was tested using the survey results with various statistical methods to validate the model. The IUS model was validated by the survey.

Nonintuitively, the IUS model suggests that for system administration tools that use a GUI, currency (up-to-the-second information) and speed were not significant factors in final user attitude, despite the fact that the survey suggested that users thought they were important traits for a tool to have. Accessibility was not important—the tool users were all power users. Scalability did not factor in, since the tools in actual use were already selected by their users as being appropriate for the scale of their systems. Scriptability stood out as an important factor that is obvious to sysadmins, but less so to management and tool authors. The IUS model may be a useful way to validate and explain to management what makes one tool better than another. Known limitations of the study include all the issues involved in a voluntary response survey conducted at one point in time.

A questioner asked about the effect of already efficient tools being surveyed. Only tools that had been chosen for use were in the survey, so self-selection was a factor. Another questioner asked whether documentation was studied—it was not.

- ***Dynamic Dependencies and Performance Improvement***
  Marc Chiarini and Alva Couch, Tufts University

Marc Chiarini stated that traditional methods of performance improvement analysis are costly, so mostly ad hoc methods based on experience and intuition are used. These methods are not always reliable or generalizable. The authors' approach to performance improvement analysis used an exterior model, a technique that treats the system as a black box and is guided by analysis of changes in externally observable statistics as a probe load is varied. The authors factor the system into components (e.g., Web server, disk, CPU, memory, file server, etc.) under test and synthesize so-called steady-state behavior. This could be thought of as a typical "nonloaded" state. It is noteworthy that production systems are seldom in steady state, as factors such as time of day and day of the week become important.

In the dependency model, different servers are competing for the same finite resources. A resource is deemed critical if reductions in its availability affect the performance of some system component. Micro Resource Saturation Tests (mRST) are run in a steady-state environment. A resource is chosen to be made less available in various increments, then response times are monitored to see whether any go "off the charts." Statistical methods are used more rigorously to test the significance of the test results. It was noted that, to make these tests significant, large enough sample sizes must be used. Also, each probe of the system must be independent of the others; this is accomplished, for example, by spacing the probes a bit.

In conclusion, the speaker stated that this model allows us to reliably improve system performance without knowing too much detail—it's a guided approach that can help systems administrators know what to expand first. A questioner asked how to do this routinely. The speaker said that more work is needed, including tool development. Other questions focused on the particular statistical methods discussed in the paper.

- ***Automatic Software Fault Diagnosis by Exploiting Application Signatures***
  Xiaoning Ding, The Ohio State University; Hai Huang, Yaoping Ruan, and Anees Shaikh, IBM T.J. Watson Research Center; Xiaodong Zhang, The Ohio State University

  Awarded Best Student Paper!

Xiaoning Ding stated that the authors' goal was to create a black-box method to automatically diagnose application faults such as bugs and misconfiguration. It was noted that misconfiguration is far more common than bugs in production systems. The basic approach is to record the normal execution of an application into a trace. Over time, a series of normal traces becomes an application signature, which might be roughly thought of as a set union of normal traces.

Run-time attributes such as system call signatures, signals, and environment variables are used. This means there is no need to instrument the application. The collection of observed values is recorded. System calls provided a challenge. A graph is made of the system calls plus their call stacks. That way, repeated calls from the same point in the code are merged to one node in the graph. When a fault occurs, the trace of the fault is compared to the signature and mismatched attributes are flagged as potential root causes. Causes closer to the head of the call trace graph are given priority, as they are more likely to be root causes. For example, intermittently failing httpd daemons were diagnosed and the correct root cause (log files close to 2 GB in an older filesystem) was automatically diagnosed.

Experiments on real-world problems found from online sources such as Bugzilla and various forums, including CVS, Apache, and PostgreSQL, were used to test this technique. The experiments worked well except in the case of hardware faults. Performance overhead was initially pretty high, as much as 30%, but after optimization the overhead could be brought as low as 2%. A questioner asked about high-noise situations, such as several high-priority, possible root causes. Lots of traces are needed to help weed out the noise.

INVITED TALK

- ***Integrating Linux (and UNIX and Mac) Identity Management in Microsoft Active Directory***
  Mike Patnode, Centrify

  Summarized by Will Nowak (wan@ccs.neu.edu)

Mike Patnode, the Vice President of Technology of Centrify Corporation, gave a state of the world overview of integrating UNIX-based machines into an Active Directory environment.

Patnode started by discussing the open source and free solutions available. Generally these open/free tools were regarded as difficult to integrate, hard to find support for, and not always the right solution. Microsoft Services for UNIX (SFU) were discussed, although it was noted that these tools are limited in scope. Microsoft does provide support, but customers using SFU may find themselves wanting more. In addition to discussing various tools for solving integration issues, Patnode painted a picture of a typical organization's mess of authentication issues—local accounts, mixed directory systems, and no synchronization anywhere.

Patnode drew several conclusions. Integration is a good thing, and organizations should strive to reach this goal. User administration becomes easier, users are happier, and the all-important auditors are happier. Picking a tool or product for the job is difficult; it is up to the organization to decide whether it wants a commercially supported full-featured system such as Centrify's products or whether it is more comfortable tooling together something with freely available options. Patnode's presentation helped put the audience in the right mindset for making these important choices.

INVITED TALK

- ***Programming the Virtual Infrastructure***
  Paul Anderson, University of Edinburgh

  Summarized by Patrick Ntawuyamara (ntawuyamarp@hotmail.com)

Paul Anderson's presentation was about the complexity of configuring the virtual infrastructure. He identified some similarities and analogies with his experience programming early computers.

Anderson described how programming has evolved and how some of the problems and solutions mirrored those of system configuration. He emphasized that the virtual infrastructure does not fundamentally change the nature of the configuration problem but increases its complexity.

Referring to John Backus (developer of Fortran) and the early programmers, he pointed out that the properties of efficiency, correctness, portability, and usability that were important in the early days of programming are still relevant for the virtual infrastructure.

He stressed that programming languages have typically taken many years to become accepted practice and only survive when they can prove themselves practically. The same is likely to be true for system configuration tools such as LCFG, Cfengine, and Bcfg2.

He also discussed some different programming paradigms and pointed out features such as agile programming that have interesting analogies in the configuration context.

Declarative specifications have become the norm for system configuration, but the virtual infrastructure requires a different kind of approach to deal with its complexity. Fully automatic configuration within this framework will be difficult to achieve. Paul referred to the I-X framework, which supports a combination of human and automatic processes to achieve its goal, and suggested that this type of hybrid configuration may be a way forward for configuring complex virtual infrastructures.

The question still remains how best to coordinate efforts to find the language design that will deal with these complexities.

For an article based on the talk, see page 20 of this issue of *;login:*.

Paul Anderson's complete presentation can be read at http://homepages.inf.ed.ac.uk/dcspaul/publications/lisa-2008-talk.pdf.

LARGE-ISH INFRASTRUCTURE

Summarized by Matthew Sacks (matthew@matthewsacks.com)

- ***Petascale System Management Experiences***
  Narayan Desai, Rick Bradshaw, Cory Lueninghoener, Andrew Cherry, Susan Coghlan, and William Scullin, Argonne National Laboratory

In this talk, representatives from the Argonne National Laboratory spoke of some of their troubles and successes administering the Intrepid supercomputing cluster. Intrepid is an IBM Blue Gene/P system with proprietary job control and monitoring systems, which the Argonne engineering team describes to be at odds with the commodity cluster designs.

In the Argonne implementation the service infrastructure required much more manual servicing than the Blue Gene/P control system and the IBM Reliability, Availability, and Serviceability (RAS) infrastructure. The Argonne team concludes that more HPC systems will need to move to a more self-contained, self-healing systems such as the Intrepid system at Argonne.

- ***Rapid Parallel Systems Deployment: Techniques for Overnight Clustering***
  Donna Cumberland, Randy Herban, Rick Irvine, Michael Shuey, and Mathieu Luisier, Purdue University

The Purdue computing team presented their paper on how they were able to deploy the 7000-core "Steele" cluster in under 24 hours. Achieving this undertaking was a monstrous task, but the team was able to overcome most of the difficulty in meeting the time limit with manpower for unboxing and racking machines. Why would anyone want to or need to deploy a supercomputing cluster in under a day? Because research staff members want to be able to use the cluster as soon as possible to meet their deadlines, and the hardware is losing operational lifespan and warranty lifetime sitting in the box. It took a total of 190 volunteers just for the unboxing, racking, and waste-disposal effort. The physical deployment team started at 8 a.m. and was done by 1:30 p.m.

A small system administration team of four people, using extremely creative modifications to standard deployment methods, performed the nonphysical initialization of the computer cluster in under 15 hours. All hosts are installed using PXE and Red Hat kickstart; however, when kickstarting the immense volume of machines all at once, the team anticipated difficulties in setting up kickstart configuration files as well as network contention. This was addressed by using IPVS (IP Virtual Server) for load-balancing kickstart file-serving and a simplified kickstart configuration by using a single IP for the cluster.

Through the Purdue team's creative efforts, they were able to prove that even a small IT group can rapidly deploy large systems with the help of a large volunteer team to power through the manual labor portion of a large deployment.

- ***ENAVis: Enterprise Network Activities Visualization***
  Qi Liao, Andrew Blaich, Aaron Striegel, and Douglas Thain, University of Notre Dame

  Awarded Best Paper!

Qi Liao presented the ENAVis network visualization project. The ENAVis model addresses the issues that most conventional network monitoring and visualization systems lack, which is correlating data in three context rings: host, user, and application. Liao said a convention for such a model is the HUA model (host, user, application). ENAVis uses agents for data collection to overcome the limitation of using only network logs or SNMP for gathering and trending network activity. The amazing thing about the ENAVis application is that the agent is a simple bash script using commonly available commands, so it is flexible in terms of the systems it can collect data from. Liao argues that point-to-point (IP address and port) netflow-type data is not useful for monitoring and visualizing network activity, which holds much weight considering applications oftentimes do not run on standard ports. Applications, hosts, and users are all taken into account, giving a more accurate view of network activity. Data is then correlated and visualized on a central server.

Liao demoed some of the abilities of ENAVis by showing a file-sharing user and which applications he was using on the university network as he utilized a large amount of bandwidth. ENAVis introduces a more modern, thorough approach to trending network data and encourages better correlation of data to provide more accurate network monitoring. The code for ENAVis is freely available.

INVITED TALK

- ***How to Proceed When 1000 Call Agents Tell You, "My Computer Is Slow": Creating a User Experience Monitoring System***
  Tobias Oetiker, OETIKER+PARTNER AG

  Summarized by Marc Chiarini (marc.chairini@tufts.edu)

A few years ago Tobias Oetiker was called by some IT staff over at Swisscom, who asked if he would help them solve perceived performance problems with their CRM software. As Oetiker jokes, they were apparently laboring under the false assumption that he was an expert in this area. Nonetheless, he decided to take up the challenge and learned quite a few lessons in the process that he graciously shared with us in this invited talk.

Swisscom has a very complex Windows IT environment with a large call center where customer service agents deal with all manner of problems and requests. Agents were intermittently experiencing slowdowns, but no one on the IT staff was capable of pinpointing the cause. Using off-the-shelf products (sysinternals tools, winspy, various Perl modules, etc.), Oetiker initially created a three-part system called CPV to get to the root of the matter. CPV comprises a passive monitoring component (monitor), an interactive reporting component (reporter), and a data analysis component (explorer).

CPV monitor feeds filtered events (mouse clicks, log entries, network events) from each client in the call center (initially kept small for buy-in) into FSMs (finite state machines) that mimic the behavior of the call center software and look for invalid sequences of events. CPV reporter is a small tray tool that allows the agent to send a wealth of system information (crash logs, configuration, etc.) along with a message to the IT staff *when* the problem is occurring. CPV explorer allows the staff to perform in-depth visual and statistical analyses of the data sent from any combination of monitors over any time period.

Oetiker's first lesson was that the complexity of FSMs has the potential to grow quickly and without bound, making them slow and impractical. Over the course of building CPV and looking for an alternative, Oetiker investigated ways in which observability at the client could be increased. Further lessons ensued, including:

1. They determined why agents could sometimes not press the modal "send" popup button in Outlook. The CRM software used an outdated aspect of a Windows system call that preemptively stole the key state. The problem was solved with Perl.
2. Switching from the GetWindowText call to WMGetText in CPV monitor caused severe slowdowns because of undocumented behavior of the latter system call.
3. Once the monitor was running on ~1500 clients and other devices, too much data was produced (100 million samples in 40 days) and the central database used for analysis was unable to keep the entire DB index in RAM (4 GB). The solution was to perform index compaction, enabling analysis of up to 7 years of data at one time.
4. Threaded Perl on win32 does a full environment copy whenever a new thread is created, utilizing a large amount of memory. The solution was to only use threads where necessary.
5. Agents and IT staff were interested in how long it took to boot a system and then log in. When agents were on the line with a customer, they wanted to know how long they would need to stall before their system came back after a crash. Oetiker used the WMI (Windows Management Instrumentation) interface to identify the contributors to boot and login time.
6. Detecting crashes and (truly) hung applications does not work in the same way as with UNIX. In the first case, the monitor needed to be built to find the active window, attach via a process handle, and continuously poll for the appropriate exit code. In the case of hangs, the active window was identified and messages were sent to it until it returned. All this was necessary to distinguish between real crashes and users closing the CRM software or CPV monitor out of frustration.

Oetiker states that the learning process with CPV is ongoing. Some of the positives are that the reporter makes agents feel useful, the explorer gives effective access to previously hidden data, there is a closed feedback loop, and IT staff now have efficient tools to perform structured problem solving. Some of the stickier issues remain: What or who is being monitored? How do agents change their behavior? How does problem diagnosis move forward in the absence of previously available information?

INVITED TALK

■ ***How to Stop Hating MySQL: Fixing Common Mistakes and Myths***
Sheeri K. Cabral, The Pythian Group

Summarized by Will Nowak (wan@ccs.neu.edu)

Sheeri Cabral presented a question: Why do we hate MySQL? Over the course of her in-depth examination of MySQL performance and configuration issues, she highlighted tips that should make any database administrator's life easier.

Cabral started with an overview of myths and rumors: Don't use ENUM, MySQL is slow, it uses too much memory, etc. For nearly all of these myths, Sheeri gave a simple and easy fix for the problem. One cool tip was to store IP addresses using INET_ATON and INET_NTOA. These two MySQL built-in functions convert a string representation of an IP address into an unsigned integer. This solves a performance issue with storing variable-length strings in a database. Throughout the course of the presentation numerous little hints like this were provided. More than just SQL, machine tuning was also covered, as were common issues with deciding whether or not to use RAID in a database server—why bother if you already have 10 replicas set up? Other software-independent issues were touched upon, such as network bottlenecks with multiple queries that could be reduced. A concrete example was given: Most people select from a database to see whether an ID exists before creating it, but you can combine those two operations into a single INSERT...ON DUPLICATE KEY UPDATE query, cutting the number of database connections in half.

TRUST AND OTHER SECURITY MATTERS

Summarized by Alex Boster (aboster@machineepsilon.com)

■ ***Fast, Cheap, and in Control: Towards Pain-Free Security!***
Sandeep Bhatt, Cat Okita, and Prasad Rao, Hewlett-Packard

Firewall ruleset analysis may yield significant gains in operational efficiency, Cat Okita said. Currently, most firewall rules are implemented once and never removed. There is a cargo-cult mentality because changes can have unpredictable effects. The authors' approach is to analyze the structural properties of security rulesets, define a standard grammar for ruleset actions, develop a set of ideal conditions for rulesets, and define metrics to measure the differences between rulesets.

Ideally, rules exhibit noninterference, simplicity, and consistency. The authors define noninterfering rules as those that do not interact with other rules. Simplicity posits that only triggered rules are defined, all defined objects are used, and rules match policy. Consistency requires objects to have consistent names. A new metric, effectiveness, is defined as a measure of the fraction of a given rule or object that is not interfered with. For example, if "permit ssh, telnet" is followed by "deny ssh," the second rule is less effective as it is partially obscured by the first rule. A tool has been written in Java 1.5 with CLI and Web interfaces, with Checkpoint devices fully supported and Cisco PIX/ASA, pf, and ipfilter in the proof-of-concept phase.

Several use cases have been explored. The tool can be used to compare differences in configurations, for example, for migration planning and verification, pre-change impact verification, post-change confirmation, and incident analysis. Ruleset remediation such as identification and removal of ineffective rules, resolution of rule conflicts, and improved accuracy of rule sets is possible. And the tool can be used for reporting.

The authors have made a number of discoveries. More rules lead to more issues. Humans are effective at finding completely ineffective rules, but they are bad at finding partially ineffective ones. Ruleset effectiveness declines over time. People clean up rules, but not objects.

A questioner asked about typical effectiveness ratings. These are 50% or more. Another questioner asked whether unused rules, like dead code, aren't really harmless. Compliers strip dead code but firewalls are still slowed down by ineffective rules.

■ ***Concord: A Secure Mobile Data Authorization Framework for Regulatory Compliance***
Gautam Singaraju and Brent Hoon Kang, University of North Carolina at Charlotte

Gautam Singaraju began by saying that regulatory compliance standards (RCS), such as the Feinstein bill and HIPPA, stipulate the safeguarding of data confidentiality and the reporting of incidents. In complex corporate environments threats come from a number of sources: servers with bad configurations, unpatched servers, spyware, and malware. Mobile devices have most of these issues, plus they can be lost or stolen. Furthermore, internal users are the largest threat, with 60% of attacks coming from disgruntled employees and other insiders.

The Concord framework addresses these issues by assigning trust to a collective interaction of system components, not just one component. Clients, including mobile clients, must get authorization before accessing even local data. Such access requires both user and system authorization. Such authorization can be revoked upon loss of a mobile device. In addition, accesses are logged, so organizations can discover what assets were compromised after an incident.

The prototype implementation is in Java, using Java ME for mobile platforms, and uses mRSA for encryption. Preliminary results yield a 7.5-second access and decryption time for a client with a mobile, disconnected enforcer device, but only 0.5 seconds for a connected agent. A questioner asked about real-life use cases, given the high overhead. This was acknowledged, but the system should be used when dealing with highly sensitive data such as social security numbers. A questioner asked about backups. Old keys must be backed up along with data—there is more work to be done in that realm.

- ***Authentication on Untrusted Remote Hosts with Public-Key Sudo***
  Matthew Burnside, Mack Lu, and Angelos Keromytis, Columbia University

Matthew Burnside explained that this paper was motivated by the following scenario: Consider logging in with ssh to a remote computer, then using sudo there. The traffic is encrypted, but if the target machine is compromised, the password has been revealed to the attacker. To address this, the authors have created SudoPK.

SSH provides agent-forwarding to allow chained ssh logins. The authors have built a generic interface to SSH-USER-AUTH that leverages agent-forwarding. Their authentication module makes running sudo like hopping to another ssh host. The password stays in the encrypted tunnel. This provides for ease of use and is no worse than plain agent-forwarding.

A questioner asked whether this has been submitted back to OpenBSD. It has been, but the status is unknown. Why is this preferable to putting someone's public key in root's authorized key file? Because sudo's grant/deny scheme is richer and more fine-grained. A questioner commented that this was like Kerberos. The speaker said that it was, but it is much simpler and therefore perhaps better suited to small networks.

INVITED TALK

- ***Does Your House Have Lions? Controlling for the Risk from Trusted Insiders***
  Marcel Simon, Medco Health Solutions

No summary available: See www.usenix.org/lisa08/tech/ for presentation slides and live streaming of the talk.

INVITED TALK

- ***Spine: Automating Systems Configuration and Management***
  Rafi Khardalian, Ticketmaster

Summarized by Marc Chiarini (marc.chairini@tufts.edu)

Rafi Khardalian presented Spine, Ticketmaster's homegrown system configuration management framework. Ticketmaster's worldwide IT infrastructure comprises roughly 4000 RPM-based Linux servers running different combinations of distributed and high-availability applications. Multiple small (3- to 7-person) system engineering subteams are responsible for managing over 100 different configurations, each with subtle variations. When setting out to create a useful tool, the Ticketmaster team had several goals in mind: the ability to define a hierarchical configuration that allowed for overrides from least to most specific; the ability to leverage existing tools with a small, straightforward, modular, and pluggable codebase; a simple management interface using text files; the need for admins to only understand simple programming constructs (conditions, flow control, etc.); the minimization of configuration data duplication; and, finally, versioning and rollback functionality. From the details of the invited talk, we can surmise that these goals have largely been met.

Spine is designed to be invoked on a regular basis after machines have already been provisioned (using another homegrown tool called Provision). It has two primary run-time phases, each consisting of several subphases. The first phase is configuration parsing, which includes discovery, hierarchy traversal, and key/value processing. Discovery consists of collecting network information, identifying the team (or business unit) that will manage the server, group type (e.g., prod, dev, qa), product, system class (e.g., proxy, app, cache), and the particular class instance. A single host can also be uniquely configured in circumstances where one-offs are unavoidable. Hierarchy traversal descends a class hierarchy that is represented naturally via a filesystem. Each directory in the hierarchy is considered a "node" and must contain at least two subdirectories, config/ and overlay/, which are allowed to be empty. In the key processing subphase, files found in the config/ directory are examined. The filename serves as the key, and the lines in the file represent values (or operators, such as <regexp> to remove all entries matching the regexp from a key). Values associated with nodes higher in the tree can always be overridden by those in lower nodes. To further improve the reusability of configuration data, other config nodes can be "grafted" into the tree via a config_groups/ directory at any point in the hierarchy.

The configuration application phase is the real workhorse of Spine. All configuration is applied via plug-ins, which allow for extensions to Spine without modifying the core code. The first step is to populate a temporary staging directory with all necessary configuration files and permissions relative to root. Then, templates written in Template Toolkit can apply the necessary tweaks according to the configuration hierarchy. Once the staging directory is ready, the overlay structure is applied directly to the node's root filesystem. The next steps install and update necessary RPM packages, remove packages not explicitly listed in the configuration (and dependencies), restart or execute commands based on configuration changes, enable or disable services, configure the default boot kernel, and apply basic security best practices.

To provide versioning and rollback, Spine also provides a *publisher* component that can pull an entire configuration "release" out of a SVN repository into an ISO image. Publisher can also request "configballs" via HTTP for either a previous release or the latest update. This approach is extremely flexible and allows for regular changes to be rolled out much like software upgrades.

On balance, Spine has been serving Ticketmaster very well. Some planned improvements include making the core more generic, adding a wider variety of package management substrates (currently only RPM with APT is supported), simplifying the installation process, improving user documentation, and providing detailed API docs for creating plug-ins and extending Spine. The team is actively soliciting large installations that would allow Ticketmaster to get them up and running with Spine. More information is available at http://code.ticketmaster.com.

PLENARY SESSION

- ***Reconceptualizing Security***
  Bruce Schneier, Chief Security Technology Officer, BT

  Summarized by Leah Cardaci (lcardaci@cs.iupui.edu)

Bruce Schneier examined the human element in security and how people make decisions about risk and threats. Schneier began with the idea that two types of security have to be examined: the feeling of security and the reality of security. It is important to understand when they are the same, when they are different, and how they can differ. This can be hard to discuss, because language does not easily handle this split.

Security decisions tend to involve some type of trade-off. For example, people living in gated communities trade convenience and privacy for some added security. These types of decisions are subjective, and they may differ from person to person. The evaluation is not based on how good the result is, but on whether the gain is worth the loss. When trade-offs don't make sense, it often means that the true trade-off being made is not visible from the outside. Some imperceptible factors are involved which may not have anything to do with the security gained from the decision.

Like all animals, humans make security trade-offs all the time; it is part of survival. Human beings should be good at making these types of decisions, but they are not. Humans react to the feeling of security, not to the reality of security. This approach will succeed only while the feeling and the reality of security are the same. It breaks down as they begin to diverge. Our intuitive responses work well for the security of pre-history society, but they do not work as well for modern security issues.

To help understand how these decisions are made, two portions of the brain were examined. The amygdala is a primitive part of the brain that is the center of the "fight or flight" reflex. It reacts faster than conscious thought, but it can be overridden in humans by training. The neocortex is a newer section of the brain where consciousness is centered. It is slower and uses heuristics and is the source of cognitive biases, which can fail.

Schneier described an experiment on how people react to risk. The test subjects were divided into two groups, with each group given a different choice. The first group was asked to choose between a sure gain of $500 and a 50% chance to gain $1,000. The second group was asked to choose between a sure loss of $500 and a 50% chance of losing $1,000. It would make sense for the two groups to have similar percentages of those willing to take risks and those unwilling to take risks. However, the results showed that although 84% will take a sure gain over the chance for a greater gain, 70% will take the chance of a greater loss over a sure, but lesser, loss. This makes sense on a survival level, where any loss or even the lack of a gain can mean death. Similar studies over varied demographics suggest that this is a general human response to risk.

Other biases are a tendency toward optimism, a greater fear of risks that can't be controlled, placing greater importance to risks from people than from nature, and putting greater importance on risks to children. People tend to think that something is more likely the easier it is for them to think of it.

People also tend to value something more if they already have it than if they want to have it. This can be seen in a study where the subjects were divided into two groups, one given mugs and one without. Those with mugs were asked how much they would sell the mug for and those without were asked how much they would pay for one. Those with a mug tended to ask twice as much as those without the mug were willing to pay. This experiment has been repeated with more expensive items and with groups with different levels of wealth.

To further complicate the discussion, security can be thought of as having three parts. The feeling of security is how people intuitively feel about their security. The reality of security is the real world. The model of security is an intelligent representation of reality. Models can be limited by lack of good information and lack of technology. When models differ from feelings, they tend to be rejected. Over time, the models can be accepted and will eventually disappear into feeling. However, it can take a long time for the model to be accepted. For example, it took decades for the model of the dangers of smoking to be accepted. This can be a problem in technical security because the fast growth of the technology means the model may be defunct by the time it is accepted.

People are a crucial part of security systems, and their role has to be considered. It is important to consider more than just the security reality when designing a system. The feeling of security will also figure into how people react to the system. In addition to designing better systems, people need to be presented with better models. All three parts of security—reality, feeling, and model—need to be considered.

When asked whether people who study statistics make better decisions using mental shortcuts, Schneier replied that they do not, but they tend to be surer of their decisions. It was suggested that part of the reason that these shortcuts are used is the cost of cognition. Schneier replied that this is a factor but at this point it is hard to understand how important the cost of cognition is in when shortcuts are made. He agreed that the way people think about security can be considered rational, but he pointed out that it is not analytical. Dan Klein mentioned that it was likely that many in the audience were INTJs on a Myers-Briggs typology and thus more likely to see the flaws in these shortcuts. He asked how such more analytical people can relate to security issues. Schneier replied that the human response to story-telling should be leveraged.

VIRTUALIZATION

Summarized by Andrew Hoblitzell

- ***STORM: Simple Tool for Resource Management***
  Mark Dehus and Dirk Grunwald, University of Colorado, Boulder

Dehus said his group defined an appliance as "a combination of operating system and application components that provide a specific computing task," such as spam filtering, a wiki, etc. He introduced STORM, his group's virtualization system, which was designed to simplify development, deployment, and provisioning for common applications and appliances. He showed that the system reacts to changes in system load to deploy additional services and that it also can dynamically power client machines using IMPI controls to promote energy savings. The system was demonstrated using a scalable mail appliance, and he said that they had designed the system to be easy to configure and maintain. He said the group planned to eventually place their project on SourceForge.

In response to questions from the audience, Dehus said that the layering used by his group's system was rather immediate and that disk images were kept separately, so overhead should be kept to a minimum. Dehus said his group had not had time to test what would happen if a very large number of virtual machines were requested at once and the system was unable to handle it. Another audience member mentioned that HP had conducted similar work.

An article with more details about STORM begin2 on page 16 of this issue.

- ***IZO: Applications of Large-Window Compression to Virtual Machine Management***
  Mark A. Smith, Jan Pieper, Daniel Gruhl, and Lucas Villa Real, IBM Almaden Research Center

Mark Smith presented IZO, a large-window de-duplication compression tool that his group developed which provides faster and increased compression over existing large-window compression tools. He said that large-window compression methods were becoming more and more relevant because of falling storage costs and larger storage applications and that this would also make his group's tool increasingly relevant. He showed that they had applied their method to a number of VM management domains (e.g., deep freeze, backup) and that his group's system would help administrators more effectively store, administer, and move virtual machines. He cited initial experiments which showed up to 86% storage savings for some scenarios, and he said that in future work his group was concerned with the rapid growth of metadata in backups, the ability to determine optimal chunk size automatically, and adding additional convenience features to their project.

In response to audience questions, Smith said that chunk size would be adjustable to allow for lowering the probability of hash collisions for specific applications. When he was discussing the current rapid growth of data in the computer industry, Smith said he was happy to see that a new linear accelerator was currently being constructed in France.

- ***Portable Desktop Applications Based on P2P Transportation and Virtualization***
  Youhui Zhang, Xiaoling Wang, and Liang Hong, Tsinghua University

This paper focused on play-on-demand for common desktop applications. The group's approach was based on lightweight virtualization and network transportation to allow a user to run personalized software on any computer, even if that computer doesn't have that software installed on it locally. In the group's method, registry, files/directories, environment variables, and such are sent to a portable device as they are needed. Access control methods could be added to the method to prevent illegal access, and other methods besides P2P could be used to implement the methodology. The group's method was said to be especially relevant for the developing world. In the future, the group planned to make the system work entirely over the network and to run in user-level mode instead of in administrator mode.

When answering questions from the audience, the speaker stated that dependent libraries could be detected during the installation of a product in a software environment. Further discussion about the issue took place offline.

INVITED TALK

- ***Mac OS X: From the Server Room to Your Pocket***
  Jordan Hubbard, Director, UNIX Technology Group, Apple, Inc.

Summarized by Matthew Sacks (matthew@matthewsacks.com)

Jordan Hubbard spoke about some of the innovations and latest developments in the Mac operating system. Hubbard provided much information about the security subsystems and modifications to the OS X operating system that were exclusive and difficult to find in the documentation. Some of the mysteries debunked included the sandbox profile language and file quarantine APIs. The sandbox system offers fine-grained controls over what sandboxed processes can do via system calls, including limiting what files or directories can be read or written and whether the process may have network access. The security system is based on TrustedBSD's Mandatory Access Control framework.

Hubbard also provided suggestions for accessing some of Apple's open source projects such as MacRuby, WebKit, and ZFS. MacRuby is a version of Ruby that runs directly on OS X's Objective C libraries. Hubbard spoke of a little-known library for MacRuby that makes Ruby development for OS X integrate into HotCocoa. HotCocoa makes it even easier to write Ruby code to quickly develop OS X applications.

Although there wasn't much talk about WebKit, which now powers Google Chrome, there was much interest in the topic of OS X supporting ZFS (as of Leopard 10.5). ZFS is now available on macosforge.org and is a read-only file system. Sun just recently announced bootable ZFS, so there may soon be ZFS support in an upcoming release of the Mac OS, although this was not discussed at the talk.

The iPhone was also a topic of discussion about the Mac OS X operating system. Mac OS X is officially a fully certified UNIX system and so is the iPhone. There were many developments with the iPhone's graphical API, Core Animation, that actually made it back into the regular operating system, deviating from the normal integration pattern of moving from desktop to mobile.

There were many exciting bits of information presented at this talk about some of the quieter innovations in OS X at Apple. Hubbard tantalized and informed the crowd with often-exclusive information about the Mac OS and the growing strength of mobile as a computing platform. His mention of a 2009Q1 release date for Snow Leopard (10.6) soon hit the Apple-loving press.

INVITED TALK

- ***An Open Audit of an Open Certification Authority***
  Ian Grigg, CAcert

  Summarized by Leah Cardaci (lcardaci@cs.iupui.edu)

Grigg discussed CAcert's open audit. He examined both the decision behind the choice of an open audit and how successful the open audit has been. Grigg began by discussing his role and background. He is the independent auditor for CAcert, with a background in cryptography, security protocols, and architecting systems.

Audits are often used to create a sense of trust when the security of a company is not transparent. CAcert is supposed to have an open process that will itself reveal flaws.

CAcert is an open group organized as a community. CAcert can be divided into three groups: the assurance group, the business group, and the tech group. All of the people in these groups are members of CAcert.

As CAs were introduced and their numbers increased, there came a need to manage how CAs were added to browsers. The method that emerged was a systems audit. However, audits have some problems: Audits are expensive, the audit statement is too brief to be useful, it often isn't clear, the process is closed, and the process has not been updated to reflect current threats. As Mozilla became popular, it needed to write a policy to determine the procedure for accepting CAs. Mozilla performed this task very well, appointing Frank Hecker to lead a project to create the policy for accepting CAs to the root list. Grigg called this project "one of the best open governance projects I've ever seen." This policy allowed open criteria and open reviews to qualify.

To meet Mozilla's requirements, CAcert needed to undergo an audit. In mid 2005 David Ross created the criteria for CAcert. Ian Grigg took over the audit in early 2006. An important feature of the David Ross Criteria is that they focus on the risks, liabilities, and obligations. This means that they focus on the interests of the end users. This is different from WebTrust, which seems to be set up more to protect the CA.

The industry standard is to set liability to zero. The problem with liability to end users is that the number of potential end users affected is so great that it would be an enormous cost to provide even a small amount of liability. The question is, if there is no liability, what is the value of a CA? CAcert answered this question by providing access to everyone but different levels of liability to members and nonmembers. Nonmembers may use CAcert, but they may not rely on it—there is no liability. Members are allowed to rely on CAcert certificates and accept some liability. Liability is capped at 1000 euros, is allocated from member to member, and will be decided by CAcert's own arbitration system. All members of CAcert are subject to arbitration. CAcert has an open process, which allows end users to decide whether CAcert is worthwhile.

CAcert uses a web of trust to assure identity. In CAcert's web of trust, only Assurers can make statements. This makes it important to properly validate the Assurers. To meet this need, the Assurer Challenge was introduced and passing it was required to continue as an Assurer. In addition, an Assurance Policy and Assurance Handbook were created to create a standard of assurance.

In the classical PKI model, the importance of a certificate is to provide a name. This allows trading with good people and a way to track down those who are not good. However, online markets, such as eBay, demonstrate the fact that having the "real" name itself is not as important as having a way to track reputation. The cost of resolution of disputes increase with distance, which is counterintuitive, as certificates should be more valuable as distance increases. Here the CAcert arbitration system is invaluable, because it limits the increased cost of resolution over greater distances.

The CAcert system consists of the end users on the Internet, the CAcert core server, which is accessible by the Internet, and the signing server, which is only connected to the core server by a serial link. The threats to CAcert may be grouped into bad certificates, data breaches, and the compromise of the root keys. CAcert's lines of defense are the community itself, reducing the data stored, typical online security technology, the security of the architecture, and the governance of the community.

Owing to a series of issues with hosting and problems relocating the machines in an auditably secure location, core machines failed the audit, as did the root key stored on some of those systems. Because of these events, the entire system administration team needed to be replaced with a new location and new team in the Netherlands.

Currently, physical security is ready to be audited, the logical security is close to being ready to audit, the documentation has a bug in CPS that needs to be fixed and the security manual needs some work, and eventually there needs to be some type of cross-team security. The next steps for the audit are to check documentation and assurance and to create new roots.

CAcert has done well by describing risks, liabilities, and obligations. It is also doing well by providing arbitration and progressing well with education. In the beginning, there were problems in the secrecy involved, but a move to openness has been made and CAcert has formally adopted a "no secrecy" policy.

After the talk, a questioner asked how, since in security secrecy is often considered a good thing, do you change that mindset? Grigg recommended small, gradual changes.

When asked what the plans were for physical security, Grigg replied that although the hosting building has a high security rating, there are weaknesses in the physical security. However, additional physical security is not the highest priority; instead, other security issues such as governance are being focused on now. CAcert has a different approach from other CAs; others seek to protect the root keys at all costs, which might be considered to be an absolute approach, whereas CAcert seeks to plan for all failures including compromise of roots, which Grigg referred to as disaster-style management. The latter approach is favored generally in audit criteria.

ON THE WIRE

Summarized by David Plonka (plonka@cs.wisc.edu)

- ***Topnet: A Network-aware top(1)***
  Antonis Theocharides, Demetres Antoniades, Michalis Polychronakis, Elias Athanasopoulos, and Evangelos P. Markatos, Institute of Computer Science, Foundation for Research and Technology (ICI-FORTH), Hellas, Greece

Demetres Antoniades presented Topnet, a modified Linux top command that can provide a process-oriented approach to network monitoring. Given that sysadmins use the top command to monitor process activity, it is perhaps equally accurate to characterize Topnet as implementing a network-centric approach to process monitoring. Topnet does this by augmenting the traditional top command to show two additional columns with each process's traffic inbound and outbound and, by introducing new hot keys, allowing the user to select how the processes are sorted by activity, among other things. The authors challenged themselves to essentially use only two sources of information: netstat (existing kernel-provided network statistics) and libpcap (the portable packet capture library used by tools such as tcpdump and wireshark). Thus Topnet does not require kernel modifications; instead it maintains a hash table that is used to correlate network traffic flows with processes based on the network endpoints bound to each process.

Demetres then presented results of various experiments to assess Topnet's performance. These focused on two areas: (1) the accuracy of Topnet's measurement of traffic volume over time, and (2) the load Topnet imposes on the system itself. The authors performed a number of TCP bulk-data transfers with both synthetic and real Web (via wget) and BitTorrent traffic. Overall, Topnet was shown to exhibit only small errors. Two sources of the discrepancies are (a) small differences in the binning into timeslots and (b) discrepancies that arise by observing traffic at the interface (Topnet) versus within the process (wget, etc.), such as retransmissions by the TCP layer. Regarding the load placed on the system when using Topnet, the results generally showed that although the CPU load can be prohibitively high at very high traffic rates, it grows linearly, being low at traffic rates that one would expect for most Linux machines. Thus the performance evaluation suggests that, like top, Topnet would be an effective sysadmin tool to run on demand for troubleshooting or information gathering in most environments.

One attendee asked how we might identify processes that elude Topnet, whether maliciously or otherwise. Demetres agreed that it is sometimes impossible for Topnet to associate some traffic with a process, such as when raw sockets or sockets with ephemeral bindings are employed; however, he noted that Topnet still reports such traffic as "orphaned," so the sysadmin is not left completely unaware. He further noted that it would likely require the addition of kernel support for the measurement of network traffic on a process basis to solve this problem completely. Demetres was also asked whether Topnet could report on threads rather than processes, to which he said no. Another attendee inquired whether Topnet can monitor traffic on loopback and alias interfaces. Demetres said that it can monitor any interface that libpcap can, so Topnet should be able to monitor those interfaces. Finally a number of attendees wondered when and where they could obtain Topnet. Demetres admitted that, although Topnet is based on freely available software, it is not yet available for download. This is, in part, because not all of its authors are currently available to work on it.

However, he invited interested parties to contact him by email. He can be reached at danton@ics.forth.gr.

- ***Fast Packet Classification for Snort by Native Compilation of Rules***
  Alok Tongaonkar, Sreenaath Vasudevan, and R. Sekar, Stony Brook University

Alok Tongaonkar presented work that improves the popular Snort Intrusion Detection System's (IDS) performance. Snort takes a hybrid approach to matching its rules to packet content; it first uses fast string matching techniques then follows with more general, but slower, regular expression techniques. Whereas much prior work has focused on improving only the latter, Alok and his collaborators focus on improving packet classification performance by compiling Snort rules into native code that is dynamically loaded. Their claim is that this strategy has at least three advantages: (1) It can speed up each of Snort's matching components; (2) the technique is also applicable to Deep Packet Inspection (DPI); and (3) these improvements could be used as a basis for IDS design.

Alok presented an overview of Snort's rule-based language, configuration, and variable declarations, then described how Snort uses an interpreter to process these at run time. This interpreter uses various sequentially accessed data structures and function pointers that, not unlike an interpreted programming language, limit its performance, for instance in how they force operations on the CPU data cache. The new technique achieves performance gains by having Snort use compiled rules rather than interpreted rules, in much the way that a compiled programming language and its resulting executables with native machine instructions typically yield better performance than interpreted languages. Since it would be difficult for network administrators to write, maintain, and evaluate IDS rules written directly in C code, the authors developed a compilation process that uses an intermediate language that they call the Packet Classification Language (PCL). Using PCL, the stepwise process is: (1) translate Snort rules to PCL using a PCL translator, (2) translate the PCL-specified rules to C code, and then (3) use the C compiler to create a shared object that is loaded by Snort, thus enabling Snort to perform tests to match packets to rules as native instructions. Finally, Alok presented an evaluation of the performance of this native compilation of rules versus traditional Snort, by testing 300 rules both with publicly available packet traces (DARPA '99) and using sample traffic from the campus. Overall, they achieved up to a fivefold improvement in Snort's packet classification (non–regular expression) performance for both the traces and campus traffic.

An audience member noted that PCL rules look very similar to user-supplied expressions for tcpdump and libpcap and wondered whether these improvements could be done in libpcap so that applications other than Snort might benefit. Alok said that, indeed, they have been considering how the native compilation technique could be used to improve Berkeley Packet Filter expressions.

INVITED TALK

- ***OpenSolaris and the Direction of Future Operating Systems***
  James Hughes, Sun Microsystems

Summarized by Alex Boster (aboster@machineepsilon.com)

The future of OpenSolaris, and indeed operating systems in general, is in providing the support necessary to effectively utilize parallel processing on systems with an ever-increasing number of processor cores. Sun's Batoka server with up to 256 threads and 0.5 TB of memory is an example. The search for so-called Goldilocks applications, such as MapReduce, that can make full use of such machines continues.

High-thread-count machines are the future and the winners will solve their problems with highly parallel solutions. Run-time parallelization remains the holy grail in the realm, with languages hiding the parallelism from the programmer. Operating systems must provide the tools, libraries, and developer support for this shift, by reducing complexity while enabling efficiency.

There are new features of Solaris that the audience might not be aware of. Solaris is now NUMA aware, including a NUMA memory management system that is fully transparent to the developer. ZFS is now a first-class, bootable filesystem usable as the root filesystem. Among other features, ZFS provides data integrity features, encryption, and snapshots. DTrace, familiar to this audience, allows complex debugging logic to be executed on production code. The future must enable bigger, faster applications, many of which, from simulation to games to finance, are numerically intensive.

Currently, there are several noteworthy parallel programming languages. Fortress is noteworthy for enabling implicit parallelism—hiding the complexity from the programmer. MapReduce is a real-world example in common use (e.g., by Google). Both Hadoop, a scalable Java MapReduce solution, and Phoenix, written in C, are available for OpenSolaris.

The coming revolution toward high-thread-count parallelism looks to be the biggest architectural change in computing history. Applications must become scalable or die. However, computers will be able to do a lot of this parallelization for the developer—such IT infrastructure will be a competitive advantage.

OpenSolaris is the leading edge of Solaris, with ZFS root and boot, new packaging and patching facilities (thanks to ZFS snapshots), and a userland more familiar to Linux users. It is possible that OpenSolaris will become Solaris 11.

New security features include encrypted storage, key management, and high assurance containment (e.g., running Windows in a Solaris TX labeled zone). With the spiraling issues surrounding securing data in a world with laptops, thumb drives, smartphones, and so on, encrypted ZFS (eZFS) is being developed. Most filesystem encryption systems still have a number of weaknesses, such as: root can read user data when the user is not logged in; deletion does not erase data; and raw data on RAID may be in the clear. eZFS addresses each of these issues by requiring the user's key to access data and zeroing data on deletion. Key management is a critical problem, but the solutions are not high-tech: Keys must be captured, categorized, and managed over their whole lifetime.

A questioner asked about performance testing of eZFS on x86 hardware. The presenter noted that ZFS is cache-hungry and asynchronous, so if the load fits in cache, latency is low—also, encryption is four times faster than compression. To a question about large vendor support for OpenSolaris, Hughes said that when there is enough community demand, support should be forthcoming. A questioner asked how one does backups on eZFS user data if it is encrypted from the admin user. The backup system must have user keys, Hughes explained, or the users must handle their own backups. In response to whether Linux is seen as a competitor to OpenSolaris, Hughes agreed that it is.

INVITED TALK

- ***Auditing UNIX File Systems***
  Raphael Reich, Varonis

  Summarized by Matthew Sacks (matthew@matthewsacks.com)

Reich presented some of the reasons for auditing and common pitfalls in auditing UNIX file systems and how auditing fits into the broader topic of IT governance. The context of auditing UNIX file systems in this talk was focused on unstructured data, such as documents, images, blueprints, or any kind of data stored on a file share. IDC estimates that unstructured data represents about 80 percent of all business data. Reich claims that unstructured data is overly accessible in general.

The problem with auditing file systems is that it is difficult to understand who owns data, and using metadata is a poor method for understanding the owners and consumers of data. This problem becomes increasingly complex when data is being migrated across different locations. IT professionals often do not have the business context for the data, although they are often the individuals managing where the data is stored.

Auditing usually involves capturing live, real-time information, which causes poor performance on the system being audited. As a result, the methodologies for auditing data are typically episodic, so data is only captured when an incident happens or is reported and defeats the purpose of auditing or results in too much information to sift through.

The system for solving this problem is to use probes and a central database to constrain the auditing information. Resolution of incidents where a user has access to data that he or she should not have is done by only running a simple command or, if using a graphical interface, by clicking.

The end goal is that people who need to have access to data only have access to the data they need, and those who do not need to access certain data will not be able to. Without a system or way to audit this process, it is an inefficient, manual process. Reich admits that he works for a vendor whose focus is an auditing product, but that this type of auditing awareness and implementation is important regardless of where the auditing solution comes from.

GETTING STUFF DONE

No summary available: See www.usenix.org/lisa08/tech/ for the papers in HTML and PDF.

INVITED TALK

- ***WTFM: Documentation and the System Administrator***
  Janice Gelb, Sun Microsystems

  Summarized by Alex Boster (aboster@machineepsilon.com)

Most system administrators fear and hate documentation, both writing and reading it. This talk attempted to alleviate that frustration by explaining why system administration documentation is important, showing how to resolve common documentation problem areas using real-world examples, and describing how to improve product documentation from your company and from companies that make products you use.

Documenting systems and procedures can reveal missing information, gaps, and inefficiencies. What should be documented? The list includes project plans, diagrams, infrastructure details, feature and equipment requests, server log formats, and backup and restore procedures, as well as user documentation.

If starting from scratch, consider hiring or assigning a technical writer for the initial wave of documentation. Make an outline of the document and perhaps use existing documentation structures, such as those suggested by Network DNA (http://network-documentation.com/). Structuring before you write helps make sure your documentation covers all necessary material. Making a documentation template helps a lot. Keeping documentation up to date is also very important.

A long list of detailed techniques and dos and don'ts was presented, along with real-world examples from commercial product documentation. Problems with procedure lists included failing to number sequential steps or numbering steps that aren't sequential, burying actions in paragraphs, and failing to note important information before the step in which it applies. When and how to use lists, tables, FAQs, and illustrations was discussed.

A questioner asked about wikis. These are good for community involvement, but not for product documentation—as such, though, they are often useful for system administration documentation. A questioner suggested that style guides, such as Strunk and White, were useful resources. The speaker recommended their use and also mentioned *Read Me First! A Style Guide for the Computer Industry*, for which the presenter was the project lead, but which is not Sun specific. In response to a question about how to hire help on a shoestring budget, starving students and students in organizations devoted to tech writing were mentioned as good resources.

INVITED TALK

- ***Fighting Spam with pf***
  Dan Langille, Afilias USA, Inc.

  Summarized by Qi Liao (qliao@nd.edu)

Dan Langille talked about how to use a firewall to fight off spam. The specific firewall in question presented in this talk is the Packet Filter (pf) in OpenBSD and FreeBSD. There has been increasing adoption among system administrators toward using "greylisting" to fight spam. Dan showed how to achieve this greylisting functionality in pf.

If the sender is known to be good, its packets will be directly passed to the local MTA (whitelisting). If the sender is known to be bad, the connection is terminated (blacklisting). Greylisting is a practice between whitelisting and blacklisting, in the sense that the receiver is not sure about the sender. So if the sender is new to the receiver, the pf will redirect the connection request to TCP port spamd, which temporarily refuses and asks the sender to resend at a later time. The basic assumption here is that the spammer is lazy! A spammer would move on rather than requeueing the messages. If the sender does come back after several retries, spamlogd will then move items from the greylist to the whitelist stored in spamdb.

Several concerns were raised by the audience:

- High latency for emails. This is the top concern among most system administrators. In many cases, researchers collaborate over a document and circulate it around. It would be unacceptable for users to be waiting for such important emails while they are going through the greylisting process.
- Lack of standards compliance.
- Lack of empirical evidence on the false positive rates.
- Lack of a reputational score in a dynamic environment where the reputation of the sender can change over time, as it can be infected by viruses and worms.

Although this solution is not perfect, it does have the advantage of simplicity, requires no changes to existing mail servers, and provides a cheap way of defending against spam. For administrators who are interested in setting up pf/spamd, the detailed instructions can be found at http://www.freebsddiary.org/pf.php.

PLENARY SESSION

- ***The State of Electronic Voting, 2008***
  David Wagner, University of California, Berkeley

  Summarized by Alex Boster (aboster@machineepsilon.com)

How did we get here? Florida, 2000. A voting system vendor sent a plea for help: A machine had recorded negative 16,022 votes for Gore. Reflected in media vote totals, this caused Al Gore to call George Bush to concede the election. An aide managed to catch Gore while en route to give his concession speech—and the rest is history. No explanation was ever given for the –16,022 votes.

This incident, along with the Florida 2000 recount generally, was a "nuclear bomb" in election systems. In response, Congress passed the Help America Vote Act (HAVA), requiring most counties to replace old voting systems, such as punch cards. The act had a firm, use-it-or-lose-it funding deadline. This led to a rapid acceleration of the adoption of electronic voting systems, including so-called direct recording electronic (DRE) systems, typically electronic touchscreen machines with no paper record.

One example of voting system problems occurred in a 2006 U.S. House election in Sarasota, Florida (FL-13). The margin in the district was 369 votes (0.15%), whereas no vote in the House race was recorded for 18,412 ballots. In the county in question, this undervote amounted to 14% of all ballots, but other counties in the district had a more typical 2%–3% undervote. Paper ballots also showed a 2%–3% undervote. Trouble logs show lots of voter complaints in precincts that used the suspect DRE system. It is suspected that the issue was a bad ballot layout that made it easy to overlook the U.S. House race. This, along with the famous "butterfly ballot" in Florida 2000, demonstrates the major importance of usability considerations.

Reliability issues have plagued electronic voting systems. The 2004 elections yielded several examples. In North Carolina, at least 4500 votes were lost after the capacity of the machine was exceeded, yet the machine failed silently while appearing to record votes. In Columbus, Ohio, a memory card failed on a machine that did not checksum its data—it recorded more votes for Bush than there were voters in the precinct. In Broward County, Florida, negative votes were recorded after about 32,000 had been cast—a clear sign of a 16-bit signed integer overflowing.

Major security issues have also been discovered in electronic voting systems. After Diebold accidentally made an archive of its election systems software public via anonymous FTP, researchers at Johns Hopkins University quickly released a report detailing many serious security issues. Diebold refused to allow academics or government representatives to examine its hardware or software, claiming they were proprietary. In 2006, Princeton researchers gained physical access to a machine and discovered a buffer overrun exploit that allowed a software upgrade to be loaded from the memory card without authorization. One hotel minibar–style key could open the physical locks on all machines; copies have been made from photos of these keys.

In 2007, California Secretary of State Debra Bowen commissioned a top-to-bottom review of voting systems and required vendor cooperation in order to recertify their machines—and, indeed, she decertified vendors that did not cooperate. The speaker was the principal investigator for this study. The commission discovered serious security issues with all the systems studied. Cryptographic issues included trivial password encodings, hard-coded passwords, and unencrypted passwords. Many simple bugs were uncovered, such as the misuse of || versus && and using a simple char as a buffer instead of char[]. All systems allowed malicious code to propagate virally, allowing one malicious memory card to infect not just one machine but the county central machines, and then on to all voting machines. No vendor followed standard defensive and secure programming techniques. As a result of these findings, Secretary Bowen decertified all of the machines until voter verifiable paper trails with audits were in place.

How do things stand going forward? The most important attributes of voting systems are auditability, recountability, and transparency. However, one-fourth of states still use paperless DREs—the very same systems the commission studied. There is no silver bullet: Elections are complex and are run by an army of trained volunteers.

A questioner asked about any partisan correlation to the pattern of states that still use paperless DRE machines, but the speaker doubted it—other factors are more probable. Another questioner noted that other secretaries of state think Minnesota (undergoing a statewide recount at the time) was lucky to have a paper trail, but politicization had delegitimized paper. In response to a question about the mechanical lever machines used for many years, the speaker noted that they had many of the same issues with known methods of cheating, but that only one machine at a time could be rigged—there is no viral capability. A non-American audience member asked about federal standardization. Wagner noted that although there are many things the federal government could do but has not yet done, the states and counties run elections in the United States. It was noted that the Constitution gives Congress the authority to decide how its members are elected, but Wagner pointed out that there are many political constraints on this. As to whether using PDFs or TIFFs instead of paper was a option, the speaker replied that it was very hard to find a suitable replacement for paper.

WORK-IN-PROGRESS REPORTS (WIPS)

Summarized by Alex Boster (aboster@machineepsilon.com)

David Pullman of the National Institute of Standards & Technology presented "Moving to a Geographically Distributed HA Cluster Using iSCSI." An old HA cluster was presented for comparison, along with a diagram and details of the new HA cluster housed in two different buildings for better reliability. Objectives for the new system included using commodity hardware, leveraging existing Veritas/Symantec expertise, and moving to an iSCSI SAN architecture. Details of the particular hardware chosen were presented. The system is in place and running with a few post-implementation issues—including vendor issues—some of which are due to pushing the envelope on iSCSI.

Dan Kegel of Google described a pre-commit autotest system. Automatic testing on nightly builds or after commit is common, but it comes at a cost—any issues found are already in the source tree. The speaker's experience with the system he built is that pre-commit testing leads to a cleaner source tree and happier developers. The worst problems he encounters are broken tests, particularly those that only break sporadically. A questioner asked how changes are propagated to the build machine. Each developer could get a separate branch, or patches can be sent via mailing list. Two questioners asked about merging issues, but merging is a problem that always requires human intervention.

Joe Muggli of the University of Illinois at Urbana-Champaign discussed SELS, a Secure (Encrypted) Email List System (http://sels.ncsa.uiuc.edu). SELS is based on proxy encryption techniques, which enable the transformation of cipher-text from one key to another without revealing the plaintext. Exchanging emails using SELS ensures confidentiality, integrity, and authentication. This includes ensuring confidentiality while in transit through the list server, a functionality that is uniquely supported by proxy encryption. SELS makes use of OpenPGP and Mailman and is compatible with common email clients including Outlook, Thunderbird, Mac Mail, Emacs, and Mutt. Several national and international incident response teams are using SELS. A questioner asked about the use of Mailman, a system with a history of security issues. The Mailman service sees only encrypted messages, so this is not considered a major problem.

Brent Chapman of Netomata presented "Automating Network Configuration: Netomata Config Generator (NCG)." NCG generates config files for all network services using templates and a generator engine—think of it as Puppet or Cfengine for networks. Automatic generation leads to greater reliability through more consistent configuration and better scalability owing to the ease of adding devices and services. It is written in Ruby and is in alpha now. One questioner asked where to draw the line between devices and hosts. NCG is agnostic on this issue. Another questioner asked about conflicting rules. Since NCG is template-based, the user must review the output to make sure it's sensible.

Dave Plonka of the University of Wisconsin discussed "Network Admins Are Programmers: An Analysis of Network Management Artifacts," which asks what programming tools and analysis techniques are useful in network management, with tens of thousands of devices to be managed. Device configuration files are placed in a CVS repository and various standard analysis tools are then used to analyze the repository. Each device corresponds to one CVS file, and files are grouped into "modules," which correspond to geographic areas. So, for example, one can track the activities of one, or all, users in a day and discover that Wednesday is the main day network changes are made. Lines of "code" turn out not to be a good measure for complexity, but churn (changes per day) is.

Jason Faulkner of mailtrust.com discussed some new strategies for networking and load balancing. The requirements of this system included the use of commodity hardware, use of Linux, and that the backend software be aware of the external user's IP address. There were issues that excluded the use of SNAT, DNAT, shared IP, multiple load balancers, and layer 7 proxies. This technique uses one-to-one IP to firewall mapping, and it takes advantage of the fact that Linux supports 255 routing tables. Each IP is mapped to a different routing table. Downsides include that round-robin DNS must be used and there is no way for the firewall to know about backend server load and adjust accordingly.

Will Nowak of Northeastern University discussed migrating his college to OpenLDAP from legacy systems. To that end, a modular Python-based conversion and syncing tool was written to push from NIS to LDAP. Details of the college's old SunOne LDAP cluster and the new OpenLDAP cluster were presented. Code is available at http://code.google.com/p/nisldapsync/.

Chris McEniry of Sony Computer Entertainment America presented a password management tool that uses a generic command-line interface to Web services called shiv. Knowledge of particular commands accepted by a Web service is not needed, so the CLI client is decoupled from the service. A group password storage utility has been written on top of this system, allowing all the benefits of a CLI. Other Web services, such as inventory management and auto discovered switch information, are also accessed with shiv. Chris was hopeful that shiv would be generally available soon.

Nicole Velasquez of the University of Arizona presented a configuration management tool. A test lab within IBM has developed a tool to monitor and manage its configurations from the port level. The tool is built with MySQL, Apache, and PHP. This tool has allowed users in three countries and two continents to share hundreds of servers, more than 100 switches, and over 60 enterprise storage boxes seamlessly, resulting in greater system utilization and higher availability. Clean, color-coded interfaces allow users to visualize the system at a glance and flexible reporting keeps management at bay. Workflow management functionality has been included in the tool to allow for the request and tracking of configuration changes.

INVITED TALK

- ***Deterministic System Administration***
  Andrew Hume, AT&T Labs—Research

Summarized by Marc Chiarini (marc.chairini@tufts.edu)

Andrew Hume gave an invited talk on his vigorous attempts to combat "pixies" and entropy at the data centers used by his AT&T Research Lab. The symptoms of entropy include, but are not limited to, senior architects producing nothing but complicated drawings of rack layouts, excessive use of large cable lengths, dormant racks, attached disk arrays without power, and an inability to understand what connects to what. AT&T commissioned a study that found that 100% accurate physical inventories eventually declined to 60%–70% over one year. Hume insists that we can do better.

After a brief detour into logical positivism (which describes a worldview built only on empirical statements), Hume offers a naive solution: Produce a description of what you want, place equipment in a pile in the middle of the machine room, select a piece of stuff, and verify it is connected correctly; if not, make it so. Repeat until everything is verified. The biggest problem is that, in the presence of pixies, one must verify endlessly. Because of various constraints, this is not usually done.

Fettle is Hume's primary contribution to the battle against entropy. Given a moderately simple textual description of component definitions, cable types and lengths, and specifications for layout and connectivity, Fettle is able to produce a visual representation of the desired rack design, a list of cable orders, power analyses, and more. Written in Ruby, the program is designed to provide bookkeeping functionality and quick feedback, and Hume has been pleased with the results so far. Although he admits to not caring much about networking, Hume turns to PXE booting and two more Ruby tools, Limn and Treetop (a little language), to help with the tasks of logical network layout and giving devices most of the necessary configuration information to start up.

In Fettle one can specify several (somewhat hierarchical) elements when defining one or more racks. Racks, bars (tie bars), boxes, and ports are instantiated and "anchored" (given a location according to name and dimensions). Wire elements connect any two port elements. A rack element contains the names, dimensions, and 3D locations of one or more primary doors, bars, boxes, and access ports. Bar elements specify the names, dimensions, locations, and type of cable held by each cabling channel. Boxes declare the names and dimensions of each physical device placed in a rack. Port elements describe the names and dimensions of each port assembly on a box and also the name, type, and dimensions of each individual port. Finally, wire elements describe the length and type of cable between two ports and what bar to run along. Optionally, macros can be defined to instantiate multiple servers plus the wires they need.

Once a specification is processed, Fettle outputs VRML visualizations that can be manipulated in 3D to see exactly how different racks and cabling should be laid out. This allows comparisons to be made between what should be (the model) and what is (the reality of the machine room). Cable labels are also generated from the diagram, and certain types of connectivity can be easily spot-checked (e.g., power, KVM). Fettle itself produces other kinds of output, such as automated routing of cables (which Hume admits is slow in Ruby), and machine-readable tabular summaries of the specs.

All in all, Hume considers Fettle a lifesaver, although more work needs to be done to speed up automated cable routing and improve networking aspects. He hopes that others will pick up where he leaves off when the tools are ready for general consumption. Despite its successes, tools such as Fettle can only aid the system administrator in matching reality to the ideal. Hume remains amazed at the consistency and speed with which configurations decay.

INVITED TALK

- ***Designing, Building, and Populating a 10-Megawatt Datacenter***
  Doug Hughes, D.E. Shaw Research, LLC

  Summarized by Ben Allen (bsa8923@rit.edu)

Doug Hughes from D.E. Shaw Research, a firm that develops novel algorithms and machine architectures for high-speed molecular dynamics simulations of proteins and other biological macromolecules, presented a talk on building his company's new data center and many of the choices, gotchas, and tips he found along the way.

Hughes went on to show the benefits and differences between wet and dry cooling systems. The main benefit of a wet cooling system is efficiency. However, water pipes take quite a bit of space and pipes can leak. Electronics and water do not mix well. In addition, humidity control with a wet system can be problematic. Finally, if your environment freezes in the winter, you must ensure that cooling water pipes do not freeze during maintenance periods. Dry systems, in contrast, can control humidity well using waste heat and can be placed almost anywhere with no major structural pipes required. However, dry systems are less efficient.

In a data center there are two competing factors: human comfort and maximizing the temperature difference (ΔT) between the inlet and the outlet. No one wants to work in a frigid environment nor in a blistering hot one. Creating a high ΔT results in the coldest possible "cold aisle" and hottest possible "hot aisle." The best cooling design is a compromise between human comfort and ΔT.

Humidification is used in a data center to moderate static electricity. Various sources of humidification are available, including plant steam, steam canisters, and infrared or ultrasonic systems. A general recommendation of 40%–60% humidification is the industry standard, but Hughes believes this tolerance is a bit tight.

Various economizing techniques are available to decrease the cost of cooling a data center. On the air side, venting waste heat directly outside can lead to considerable cost savings. In addition, humidification of cooler air is more efficient, as cooler air's dew point is lower. On the water side, a heat exchanger placed outdoors can increase savings, especially in colder regions.

Hughes recommends a few things when using a wet cooling system. First, disable humidification on all but one of your cooling units, or use a specific-purpose humidifier. The other option is to ensure precise calibration of all cooling units. Hughes recommended this because the cooling units will fight to humidify and dehumidify. Next, keep an eye out for changing conditions in your environment such as increased load on servers, increased or decreased number of servers, and changes in outside temperatures. Next, disable reheat. Reheating is the process of dehumidification where the system chills the air down to the dew point and then reheats it. It is much more efficient to have one unit bypass warm air from outside.

A number of points must be considered for the flooring of a data center. Generally two choices are available: cement/epoxy floors and raised floors. A cement/epoxy floor has a high weight load, but it is bad for chilled water cooling, as pipes are usually run under the floor. A raised floor is more expensive, but it allows room for chilled water piping and cables. A raised floor becomes more expensive when high load capacities are needed and as the height of the floor increases. In addition, when using a raised floor with chilled water cooling, consider that leaking water pipes and power cables do not mix well.

Hughes presented various fire-suppression techniques and technologies available for data centers. A pre-action system is a water-based system with sprinklers. However, the pipes of the system are filled with air during normal operation. When smoke is detected, a pump is turned on and floods the pipes. Water is only released when the individual sprinkler heads reach a certain temperature. The biggest problem with using a water-based system is cleanup. In addition to cleanup, water is not efficient at putting out interior fires or fires inside contained areas, as it takes a little while for the water to reach inside the contained area. The next type of fire suppression is a dry-agent-based system. The benefits of a dry system are minimal downtime, as there's no dry-out period, and that interior fires are extinguished quickly. However, some systems require room sealing and have a corrosion potential. A new, potassium-based system named Aero-K is safe for humans and hardware.

A number of suggestions were presented on power efficiency. High voltages and fewer voltage conversions equal better efficiency. Typically, each power conversion wasted about 1% to 2% of the energy. The use of a three-phase system allows 173% more power to be carried in a power line than with a single-phase system. When buying equipment, insist on high efficiency and correctly sized power supplies from vendors with power factor correction. Finally, use only as much redundancy as required.

Another consideration when building a data center is the density required (e.g., how many kilowatts per rack). Hughes noted that although blade servers offer cabling advantages and space savings, their typical power requirements per compute unit are about the same as new servers. In general, as density increases, cooling and power requirements increase too.

During the Q&A session, the suggestion was made by an audience member that sump pumps and drip pan pumps should be on protected power. A question of flywheel use in Hughes's data center was brought up. Flywheels are used as an alternative to uninterruptible power supplies. Although they only offer a short duration of power, this is typically enough time for generators to turn on. Flywheels are less toxic, require less maintenance, and are more efficient than uninterruptible power supplies.

LUNCHTIME TALK

- ***"Standard Deviations" of the "Average" System Administrator***
  Alva L. Couch, Tufts University

No summary available: See www.usenix.org/lisa08/tech/ for the presentation slides.

INVITED TALK

- ***System Administration and the Economics of Plenty***
  Tom Limoncelli, Google NYC

  Summarized by Qi Liao (qliao@nd.edu)

Tom Limoncelli talked about how changing resources changes the practice of system administration. Computing resources are getting cheaper and are ample nowadays. This movement from scarcity in the past to the current land of plenty has a significant impact on system administration policies. For example, CPU resources were once scarce, so administration mainly focused on fair timesharing. Now, since everyone has his or her own CPUs on PCs, modern policy switches to focus on desktops. Having cheaper servers shifts the dominant cost from hardware to power. Therefore, modern policy becomes focused on green power. Cheaper and larger storage makes it a community resource. Increasing network bandwidth results in the modern policy of dedicated port per user while keeping bandwidth shaping. Finally, helpdesks evolved from a rigid, do-not-want-to-be-abused attitude to a more user-friendly environment.

The role of the system and network administrator as a gatekeeper has been made obsolete by Internet resource abundance (e.g., there are 11 hours of video uploaded each minute onto YouTube, blogs, etc.). The gatekeeping role is shifting to a curator one, in a process of disintermediation (removing the middleman). The traditional model, in which the IT department picks the apps and controls everything rather than the users, is no longer valid.

Hosted applications (or the fancy name Software as a Service, SaaS) and cloud computing are gaining popularity these days. The question is, "Is cloud-based computing the end of system administration?" Tom suggested several directions in system administration: cloud systems, legacy apps, desktop life-cycle management, help desks, monitoring and SLAs, IT coordinators, change management, release engineering, and security and compliance.

One audience member was opposed to getting rid of the gatekeeper. Tom agreed and further emphasized that having the right choice of gatekeeper (what to block or allow) is often a difficult task. Another audience member offered that what he got from the talk was that the scarcity of resources is changing from hardware/bandwidth to electric power and system admins' time. Tom commented that he would like to talk about green computing and time management of system admins another time.

Tom Limoncelli is the author of *The Practice of System and Network Administration* and a few other books. For more information, visit www.EverythingSysadmin.com.

- ***Inside DreamWorks Animation Studios: A Look at Past, Present, and Future Challenges***
  Sean Kamath and Mike Cutler, PDI/DreamWorks

No summary available.

INVITED TALK

- ***Beyond VDI: Why Thin Client Computing and Virtual Desktop Infrastructures Aren't Cutting It***
  Monica Lam, MokaFive and Stanford University

  Summarized by Ben Allen (bsa8923@rit.edu)

Monica Lam presented current issues and myths of Virtual Desktop Infrastructure (VDI) and MokaFive's LivePC product. MokaFive can be found at mokafive.com, where the player portion of LivePC is available for free.

There are several reasons why thin-client computing does not reduce the cost of hardware. First is the reduced cost of a PC today and the similar cost of thin-client hardware. Monika gave the example of a suitable PC costing $499 and thin-client hardware costing $300 plus $60 a year. In addition, employers can depend on employees using their own computers. Another reason is that moving desktop virtualization into a data center incurs additional datacenter operational costs (e.g., having to provide cooling and power). If the virtualization is running at the end point, often passive cooling can be used. Finally, when designing the systems for servers in a data center to support virtualization you must provision for the "Super Bowl effect" or the theoretical event when all your users log in and use their virtualized desktops at the same time.

Centralized management does not have to lead to a bad user experience. VDI, where the virtual machines are run on a central server, introduces a few factors that lead to a bad user experience. First, VDI has overhead requirements: Running multiple virtual machines on a single server causes resources to be shared among many users. Next, because the user is running on a remote display, all display information has to be sent and received across whatever network connection the user is on. Often this leads to very slow interaction performance. In addition, 3D graphics or other graphic-intensive applications are very difficult to interact with over a remote desktop.

MokaFive's LivePC product attempts to solve the problems of VDI by offering a centralized management interface that allows administrators the ability to create, update, and publish virtual machines. The virtual machines are published to an HTTP server and made available for downloading by the client. The client portion of the product lives on the user's machine and downloads and runs the virtual machine. Virtual machines created by LivePC maintain two virtual hard drives: one managed by the administrator remotely and another used to store any local changes to the virtual machine. LivePC will automatically pull differential updates to the first virtual hard drive as it is updated by the administrator. In addition, LivePC allows the user to revert back to the original virtual machine, undoing any changes to the operating system.

In response to a question about the case of users needing shared access to large pools of data, Monica noted that this is the one application where VDI is quite useful. Another question was asked how to back up local changes in the LivePC product. Monica responded that MokaFive decided to let users decide how to back up local changes, as most enterprises have their own backup solutions in place. A security-based question arose regarding whether LivePC prevents the host OS from screen capturing the guest OS. Monika said MokaFive treats any data that is displayed on the screen as gone and not securable. She then noted that the only real solution to this problem is a trusted computing environment. The last question of the session was whether MokaFive offered a bare-metal install of its LivePC product. Monika answered that MokaFive initially developed a bare-metal installation before going to using the virtual machine player model, and it is still available.

UNIVERSITY ISSUES WORKSHOP

Summarized by Rowan Littell (rowan@hovenweep.org), with help from Josh Simon

In its fourth year, the University Issues workshop included 15 participants from a variety of higher education institutions, primarily in the United States, with representation this year also from Finland, Australia, and Slovenia. Schools represented ranged in size from a few hundred students to tens of thousands, some with single campuses and others with over 20 campuses. The format of the workshop was a semi-structured round table discussion; one participant described it as Advanced Topics for people in education.

One of the topics that generated the most discussion was organizational structure. Many larger schools have divisions between central computing and various academic and administrative departments. These divisions lead to tensions and challenges, such as who owns the network equipment within a research cluster or how cost recovery is performed for shared services. Being able to compare methods of dealing with these was a highlight of the workshop. Participants stressed the importance of good communications channels within IT departments and with the rest of the institution and having SLAs to structure support agreements with other departments.

An ongoing area of discussion from previous years was the outsourcing of core software services to such places as Google Apps or other hosted providers. One participant described her institution's project to provide the option of Google Apps for all students, noting that the challenges were only partially technical and that of greater importance is having a policy and governance structure for the outsourcing of a core IT service. Regarding such efforts, others brought up concerns about document-retention policies, particularly for public institutions, and information classification and protection. A slightly different area of concern, particularly regarding email, was the fact that students are starting to prefer newer forms of instant communication over email and not seeing the value in an email account provided by the school; some places have considered providing different communications solutions for faculty and staff versus students or even not providing students with local email accounts unless requested.

Identity management systems were mentioned at several times during the day, as they have implications for many of the other topics of discussion. Although some institutions are able to consolidate all authentication and identity management into one system such as Active Directory, others use a variety of solutions, including different LDAP implementations, Kerberos, and Shibboleth, most of which are tied together with local tools. Service authorization is still a problem area; traditional UNIX groups, even in LDAP, have limits. One institution is using Shibboleth-style entitlements and LDAP ACLs to limit access to services, and others are using Cosign in conjunction with single sign-on.

A number of other topics were discussed briefly, including virtualization systems, print management, feedback structures for users, and open source software. Several places are starting to use virtualization for production systems, usually lightweight servers (Web, DNS, DHCP); however, its use is sometimes limited by the political need to tie a particular project to identifiable hardware. Many institutions are inclined to use open source software; however, management still wants the kind of supportability and accountability that seems to come from commercial vendors.

The workshop closed with participants briefly describing some of the things that they've found that really work well, whether they're large systems or simple tools; the most popular suggestion in this discussion was the use of IPSca for free educational SSL certificates. In response to a final question, most participants said they expected to be working in academia a year from now.

GOVERNMENT AND MILITARY SYSTEM ADMINISTRATION WORKSHOP

Summarized by Andrew Seely (seelya@saic.com)

The Government and Military System Administration Workshop was attended by representatives from the U.S. Department of Defense, U.S. Department of Energy, NASA, Raytheon, CSC, Science Applications International Corporation, Advanced Concepts, Los Alamos National Lab, Internet Software Consortium, Equilibrium Networks, Makena Technologies, Cfengine AS, and USENIX. Although there have been .gov BoFs in the past, this was the first time a workshop with this focus has been held at LISA.

The workshop concept was to create a forum to discuss common challenges, problems, solutions, and information unique to the government sector, where participants would be able to gain and share insight into the broad range of government system administration requirements. LISA allowed diverse government and military organizations to come together in a unique forum; it's not common to have highly technical staff from DoD, DoE, NASA, and industry at the same table to candidly discuss everything from power supplies to policy. All expected to find similarities and hoped to discover exposure to new ideas, and no one went away disappointed. The day's specific agenda was developed in the weeks before the workshop through email, with each attendee providing a short introduction and identifying a specific goal that he or she hoped the workshop would address. The agenda was adjusted as the workshop progressed, in order to capture emergent topics.

While the workshop met goals for general discussion, it also produced several "lightbulb" moments that were taken away for action: Three potential corporate partnerships were established. Datacenter environmental modeling tools were introduced. Information on setting up a corporation for holding security clearances was shared. A thin-client bug fix applicable to a military command in Europe was uncovered. Useful ideas for productivity while awaiting a clearance and practical ideas for coping with frustrations at the constraints of the government environment were introduced by people who have discovered creative solutions to these hard problems.

The day started with introductions and a reminder that the environment was uncleared and that non-U.S. people were in the room. For system administrators outside the government sector this would seem like an unusual caveat, but for people who work in classified environments it is always a safe reminder to state what the appropriate level of discussion is for any new situation, especially when the discussion is about government systems and capabilities. The group agreed that the day would be strictly *unclassified* and that no For Official Use Only or higher material would be discussed.

The day was loosely divided between technical and organizational topics. Technical topics discussed included products, challenges, solutions, and future issues with multilevel security systems (MLSs), PKI and identity management systems, and infrastructure issues, including cooling, plumbing, and how HVAC in some cases has overcome CPU as a primary metric for procurement. Open source and software development issues in the government domain were also addressed, as were setting up and maintaining development labs and using virtual networking.

A short presentation on DNSSEC was provided by the ISC. OMB Memorandum M-08-23, which mandates the use of DNSSEC on all .gov domain systems by the end of 2009, was introduced and its impact discussed. New software opportunities and small business initiatives were discussed, with Cfengine AS and Equilibrium Networks representing two small companies who are posturing to do business with the government sector. This led to a detailed discussion on accreditation of software and systems and a survey of issues surrounding how a corporation can successfully interface with a government entity.

Organizational topics discussed included time management issues, general rules and regulations for systems and personnel in government and military facilities, challenges of a nonclassified government environ0ment, working with vendors who are unfamiliar with classified systems and environments, working with non-U.S. or noncleared service providers or customers, and the contractor experience (working for two or more masters while keeping the mission in focus).

Debate was opened about the presence of non-U.S. attendees in the workshop. Certain levels of security clearance require the reporting of foreign contacts, which could discourage some people from attending this workshop in the future. It was agreed by all that the presence of non-U.S. attendees did not in any way detract from any discussion, that no additional information would have been discussed given a U.S.-only room, and that the inclusion of our non-U.S. attendees contributed significantly to our overall discussion. It was agreed that any future workshops would explicitly state that non-U.S. attendees were welcome and that all workshop discussion would be unclassified and unrestricted.

All attendees presented what types of personnel their respective sites or companies are seeking to hire. Over half had positions to fill but almost all required clearances for defense work. DoE and NASA were not generally hiring, but defense organizations were. Hiring information and career Web sites were shared.

The final topic was to determine whether there would be sufficient interest in this workshop to repeat it at LISA '09. It was agreed that it was a valuable experience for all attendees and that all would support a follow-on workshop. This workshop was a small step forward in shaping our profession of system administration in the government and military sector.

ADVANCED TOPICS WORKSHOP

Summarized by Josh Simon (jss@clock.org)

Tuesday's sessions began with the Advanced Topics Workshop; once again, Adam Moskowitz was our host, moderator, and referee. We started with our usual administrative announcements and the overview of the moderation software for the five new folks. Then we went around the room and did introductions.

For a variety of reasons, several of the Usual Suspects weren't at this year's workshop. Despite this, in representation, businesses (including consultants) outnumbered universities by about 4 to 1 again; over the course of the day, the room included 5 LISA program chairs (past, present, and future, up from 4 last year) and 9 past or present members of the USENIX, SAGE, or LOPSA Boards (down from 11 last year).

Our first topic was a round-the-room survey of the biggest problem we'd had over the past year. Common threads include career paths, such as whether to leave system administration for management or development, stay in system administration and motivate both yourself and your employees in a stagnating position, or find a job that's a better fit; reorganizations and lack of structure; doing more with less; and writing tools to automate tasks.

Our next topic was a brief comment about the effective expiration of RAID 5 as disk sizes increase. When you have a 5+1 RAID 5 array of terabyte drives, recomputing the checksum during recovery requires reading 5 TB of data; any unrecoverable error means data loss. Using 2 TB or larger disks means that the odds of unrecoverable errors rise to 80% or higher. Andrew Hume said, "By the end of 2009, anyone still using RAID-5 storage on large drives will be professionally negligent."

We next discussed storage. There was some question as to the best way to make very large data sets available to multiple machines at the same time. Some sites are stuck with NFS version 3 because of interoperability issues. The consensus is that NFS is like VHS: It's not the best technology, but it's what we've got: Use it if you want it, or don't use it and write your own. If you're doing high-performance computing (HPC), GFS, Lustre, and ZFS may be worth investigating, depending on your requirements. The consensus on using iSCSI heavily in production server environments is "Don't."

Our next discussion was automation. We started with automation of network configurations, since all the good solutions now in that space cost money. There should be a free tool, such as Cfengine or Puppet, explicitly for networking: It should inspect your environment and all its configurations. The goal here is more about managing racks, power, load balancers, VLAN configurations, ACLs on the switches, NAT on the firewall, updating the monitoring (Nagios), and trending (MRTG) tools. Other automation tools mentioned include Augeas and Presto.

The mention of integrating with monitoring led to a discussion as to when it's appropriate to add a new server or service to your monitoring system. One argument is to deploy both service and monitoring updates at the same time, with alerts silenced until it goes into production because it tightly couples deployment and monitoring. Another argument is only to add the new service to monitoring when it's ready to go into production, although this is harder to automate in a one-stop-shop mode, since there can be a long time between initial deployment and going into production. There is no single right answer, since the pros and cons tend to be environment-specific.

After our morning break, we resumed with a round-robin list of the latest favorite tools. This year, the list included BLCR (Berkeley Lab Checkpoint/Restart), C++, Cfengine, git, IPMI (Intelligent Platform Management Interface), MacBook, pconsole, pester, pfsense, Roomba, Ruby, SVN, Slurm, TiddlyWiki, Tom Binh backpacks, virtualization (including OpenVZ and Parallels), and ZFS.

Our next discussion was on cloud computing and virtualization. We're seeing it more and more and are wondering where the edge cases are. It tends to work well at the commodity level but not for certain services (such as file services). Managing virtual machines can be problematic as well. Some people are too optimistic with what they think they can gain; some folks are over-allocating machines, which can lead to outages. Managing the loads is a hard problem, since the tools may not show accurate information. HPC shops don't see much gain from virtualization, since they tend to be very computation-intensive and the ability to relocate virtual machines to different physical machines may not be enough of a gain to be worthwhile. The consensus was that virtualization is more useful in smaller development and Web service shops, especially in providing QA or test environments that look like their production counterparts. It's also useful to try something new: Take a snapshot, then work on it, and if you destroy it (or the software install wipes it out, or the patch blows up horribly) you can roll back to the snapshot you took. Finally, server consolidation (especially in data centers) and reducing power consumption is a big driver.

We next talked about career satisfaction and the lack thereof. Some senior folks (in both engineering and operations sides of the shop) are writing policies instead of doing "real work." This is similar to the shift from technical to management career paths; it works for some but not for others, in part because the work itself is different and in part because the reward is different. There's some concern that, as we age, we may lose touch with technology, in which many of us have bound our self-identity or self-worth. This is more problematic for those without similarly inclined peers with whom to discuss issues. Part of the problem is also that we as a profession are moving from server- or service-focused roles to more of a business focus; we exist to keep the business running, not to play with the cool toys. Some people have come back to system administration from management and feel that having the experience on the business side has been a huge benefit and makes them better system administrators. Additional satisfaction can come from mentoring.

This segued into a discussion about when it's appropriate to break policies when they're preventing the work from getting done. The summary is that rewriting them to avoid the problem or amending them to allow for IT-based exceptions was the best course of action.

After our lunch break, we talked more about monitoring. The best practices seem to include using both black-box monitoring tools (in which closed systems pretend to be the user) and white-box ones (in which one collects statistics and analyzes them later). Also, keeping historical data around is required if you want to do any trending analysis or need to audit anything. One argument is to capture everything, since you don't necessarily know what you'll want next month or next year; however, the counter-argument to just capture what you need for business purposes has advantages of using less disk space and making data unavailable for legal discovery later. It all depends on what you care about, and what level of failure in data collection you can live with. Clusters have interesting challenges; how much of your limited CPU (and cache and network bandwidth and so on) are you willing to allocate to monitoring, since that impacts your ability to process real data? Monitoring should not be an afterthought but an integrated part of any solution.

It should be noted that monitoring, checking the availability or function of a process, service, or server, is a different problem from alerting, telling someone or something the results of a monitoring check. This led to a discussion about not getting alerted unnecessarily. In one environment, the person who adds the monitoring rule is responsible for documenting how the Help Desk escalates issues, with the last-resort rule of "Contact the developer." This becomes more complicated in multi-tier environments (e.g., if you are monitoring in development and QA as well as production) and in environments with no 24/7 support access.

Maybe 5 of the 30 attendees were satisfied with the state of the monitoring in their environments.

Our next discussion topic arose out of the previous satisfaction issues involving bad policies. The right answer in most cases is that the policies need to be fixed, and that requires escalating through your management chain. Most policies in this context boil down to risk management for the business or enterprise. Security as it affects risk management needs to be functional instead of frustrating; there needs to be an understanding of the business needs, the risks, and how to work both within and around the policies as needed. We need to move away from the us-versus-them mentality with security for things like this. This might even include getting written exceptions to the policy (e.g., "No downloading of any software is allowed, except for the IT group whose job it is to do so"). Note also that some suppliers are more trustworthy than others, so some stuff can be fast-tracked. Document that into the policy as well. Policies should have owners to contact for review or explanation.

Next we did another round-robin on the next big things on our plate for the next year. For us, it includes automating manageability, backing up terabytes per day, building out new and consolidating existing data centers, centralizing authentication, dealing with globalization and reorganizations, designing a system correctly now to deploy in three years, doing more with less, excising encroaching bad managers from our projects, finding a new satisfying job, mentoring junior administrators, moving back to technology from management, remotely managing a supercomputing center, rolling out new services (hardware, OS, and software) the right way, scaling software to a hundred thousand nodes, transitioning from a server/host-based to a service-based model, virtualizing infrastructure services, and writing policies.

After the afternoon break we had a brief discussion about IPv6. A little fewer than half of us are doing anything with it, and those mostly on the client side. The consensus is that there's no good transition documentation explaining what providers need to do to transition from v4 to v6. It was noted that you need to say, "Here's the specific thing we need IPv6 to accomplish," then you'll be able to move forward instead of being thought of as the crazy one.

Next we discussed chargebacks; people seem to find it mostly useful. Some places have problems with their internal auditors. It was noted that chargeback encourages perverse behavior, such as doing what's easiest to measure and not what's necessarily useful or desired. Also, some departments tried to charge the time it took to convert to a new system to the department that rolled that system out. Some want to use chargebacks to provide accounting and force departments to forecast quantities, such as CPU time or disk space utilization. Charging for the technology resource at some rate (such as per CPU hour or per gigabyte per month) tends to work well, but that cost needs to include the human cost and yet not be so large as to discourage users from using your service.

Our next discussion was on professionalism and mentoring. How do we attract new blood into system administration? There's no good answer; in some environments, clearances are needed; in universities, many of the technically interested people go into development rather than system administration. Hiring student interns who want to be system administrators can help (if you're in a position to hire students), or going to local user groups, but good people are hard to find.

It may be that market forces will help; the demand for system administrators will drive up salaries in the long run. In tandem, recent articles mention that system administrators, network administrators, and database administrators are recession-proof jobs. But money talks. However, it's hard to get quality if you're interested more in money than the work. There's also conflation of the term "system administrator": Is it working with big, cool machines, or supporting users, or fixing printers, or being the computer janitor? People are starting to recognize that professionalism is important. Expectations as to IT staff behavior are higher than in the past: Use full sentences, be polite, answer questions, and help solve problems.

This boils down to how we get people into the profession. They're already maintaining their own desktop and so they're not seeing any of the cool side. People come in through the help desk and programming, but what other vectors are there (and how common are they)? It used to be hard to provide certain services that are now trivially easy. For example, mirroring is easy now (using rsync and cheap disk).

Our last discussion was on power consumption and "green" computing. Many places are running out of space, power, or both, and they need to increase efficiency. Most non-HPC places are just starting to look at the whole issue, although there's general consensus that it makes sense, both in terms of environmental issues (reduce, reuse, recycle) and economic issues (lower power bills with more instructions per kilowatt). Suggestions included defaulting to duplex printing, powering off desktops and monitors overnight, raising the cold aisle temperatures one degree in your data centers, running three-phase 208V power, virtualization of services that can be, and not allowing "throw hardware at it" as a solution. Low-power CPUs and variable-speed disk drives may help out as well.

This year's Talkies Award goes to DJ Gregor. Last year's winner, David Williamson, was not present; Andrew Hume, a former Talkies Award winner, was in the bottom five this year. (On a personal note, I actually managed to get my name in the speakers' queue on a relevant issue, surprising our moderator.)

VIRTUAL INFRASTRUCTURES WORKSHOP

Summarized by Kyrre Begnum (Kyrre.Begnum@iu.hio.no)

Virtualization was a key topic at this year's LISA conference, with virtualization-specific tutorials nearly every day. Paul Anderson decided to run a workshop with virtual infrastructures in mind. The workshop aimed at identifying the present-day challenges in integrating and running virtualization in large infrastructures. He did a lot of work during the planning phase to get people of different fields to give short presentations. In the end Kyrre Begnum chaired the workshop.

After the presentations we did some quick polls to identify the types of attendees. The group could be divided into three general classes: practitioners, who currently were using virtual machines (often on a large scale), researchers, whose main concern was management of virtual machines and automated service deployment, and sysadmins, who were going to deploy virtualization at some point and wanted to learn more.

Most of the practitioners were using more than one virtualization technology. Everyone believed that the number of virtual machines was going to expand in the future.

The workshop was organized around short presentations with following discussions. The presentations were divided into three subjects: deployment and performance, service management, and virtual machine management. The first subject was initiated by Gihan Munashinge, who presented his real-life experience with hosting virtual machines for customers. This talk helped set the tone for the rest of the workshop. Some very important discussion topics surfaced quickly, such as storage and lack of technology-independent management tools. All practitioners considered storage a major factor in the success of the virtual infrastructure. Three dimensions of storage were discussed: reliability, performance, and management. Most large infrastructures depended on redundant storage. ISCSI and NFS were common, but with low performance in the latter. Some had created their own storage solution, such as the layered approach used in the STORM project (see the article in this issue of *;login:*).

Next, Lamia Youseff presented performance results from using Xen virtual machines for HPC clusters. The lack of significant performance penalties intrigued the audience, and the discussion turned toward comparing experiences and impressions on real-life performance of VMs. One interesting topic here is the way in which VMs underperform compared to a traditional hardware-based server. Performance degradation appeared to be more dramatic after a certain threshold was crossed. The lack of publications comparing performance of different technologies was discussed briefly.

Deployment issues were laid to rest and focus shifted toward deploying services and approaches to create autonomic tools. The first presenter was Andy Gordon from Microsoft Research. The focus was on describing both the setup and the operational logic of a running service. A prototype system was presented, Baltic, where the overall functioning of a service was described in F#. Features such as automated scaling were supported and could be described in operational terms. Along similar lines, Nigel Edwards from HP Labs presented his experience deploying SAP on virtual machines. He shared with us some interesting real-life issues with dynamic services and cloud-like scenarios, such as added complexity in management, software licensing, and loss of control. Both presentations illustrated the potential in automated scenarios, but most practitioners used manual operations today to roll out new VMs. For some, scripts or configuration management tools inside the virtual machine would do the individualization of the VM.

Licensing was also discussed in this context. Many licenses were VM-unaware; this created problems for sysadmins. One example is a license that is hardware-profile aware. In such a case, moving the software over to a VM from a physical server would be problematic. Also, cloning VMs would potentially violate single-copy licenses.

The last topic was management and security. Anna Fischer from HP Labs talked about how to achieve secure communication among virtual machines, even when they are on different servers. Her architecture used MAC-address rewriting to create transparent communication among individual virtual machines. Several networking-related problems were discussed in relation to Fischer's work (e.g., the problem of inserting security tools into the servers in order to protect virtual machines). Further, enabling QoS on the network in order to quench VM traffic was discussed. Most practitioners did not enforce QoS on the virtual machines; instead they had several separate networks: SAN, management, and LAN.

Richard Elling from Sun talked briefly about reliability and fault tolerance in virtual infrastructures. He then proceeded to discuss bundling demos into virtual machines with regard to a new storage product released by Sun.

Kyrre Begnum talked about approaches for virtual machine management. His argument was that creating architectures for load-balancing services and virtual machines was very difficult, and he saw little adoption by the community. A different approach would be to put much of the monitoring and decision-making into the virtual machine itself, letting the underlying servers play a more passive role. There was a lively discussion around this approach, where trade-offs between the two approaches were analyzed. Many found a so-called hybrid approach interesting, where the virtual machines assisted the decision-making of the servers based on their individual policies.

This workshop provided an excellent opportunity for practitioners, researchers, and curious minds to exchange ideas and experience. The discussions were fruitful, with most of the 27 participants chiming in on various subjects.

Management of virtual machines seemed to be one of the key issues for most practitioners. Decoupling the management interface from the virtualization technology would be one way in which different management approaches could be tried on the same infrastructure without switching the underlying virtualization layer. Describing the behavior of services on a high level and transforming this description into real deployments are research challenges. Still, people from each camp came together in breaks and continued discussions also after the workshop. Many were interested in keeping in touch later and updating each other on new developments.

Our thanks are owed to Patrick Ntawuyamara and Nii Apleh Lartey for taking notes during the workshop.

CHIMIT '08: Symposium on Computer Human Interaction for the Management of Information Technology

Sponsored by ACM, in cooperation with USENIX

San Diego, CA
November 14–15, 2008

Summarized by Eser Kandogan (eser@us.ibm.com) and Æleen Frisch (aefrisch@lorentzian.com)

The second ACM CHIMIT Symposium took place in San Diego, right after the USENIX LISA conference. Like LISA, CHIMIT is about IT management, but from the perspectives of technology, people, and business. IT is vital to millions of people at home, at work, or on the go, in small or large enterprises. CHIMIT brings together researchers and practitioners who design systems management tools, to discuss human factors issues, system administrator work practices, and systems design issues. Presenters discussed their latest research on usability models of system administration, studies on system administrator practices regarding knowledge and activity management, adoption of user-centered design practices for IT systems, and human factors in system configuration and security. The symposium was sponsored by ACM in cooperation with USENIX, with sponsorship from IBM, Microsoft, and HP.

PLENARY

- ***I Got My Jet Pack and I'm Still Not Happy***
  David Blank-Edelman, Northeastern University

Even after decades of development of increasingly complex and sophisticated system administration tools, significant problems remain, both with them and for system administration in general. Historically, the tools used by system administrators have barely kept up with the challenges of the field, and the continuing promises of solutions from new tools and new interfaces have provided little but temporary and limited relief. Offerings in this area have been like the jet packs referenced in the title. Jet packs are now available—for a substantial price—but they fly neither high nor far and are a far cry from the devices envisioned in so many science fiction stories. Similarly, grand promises of innovative and cool solutions to system administration problems being just around the corner have been realized only as minute and mostly insignificant changes in interface look and feel.

Blank-Edelman highlighted the challenges still facing system administrators by reviewing a series of relationships to other fields that he has presented in the past. These comparisons both highlight the problems faced by system administrators and suggest innovative lines of approach for further investigation and thinking. For example, system administrators are like veterinarians in that they must diagnose and correct problems with systems which cannot answer direct questions (unlike doctors), for which there is little in the way of instrumentation (unlike auto mechanics), and which must generally remain functioning while being treated/modified. Looking at the diagnostic processes used by vets as well as the ways they are trained (classroom instruction combined with clinical practice) suggests directions for both system administration education and training and the sorts of tools that would be most helpful but that currently do not exist. Blank-Edelman also considered similarities to sex therapists and to storytellers and sign language interpreters, in order to illuminate challenges and potentially fruitful approaches and directions for diagnostic, debugging, and communication issues in system administration.

PAPERS: WORK PRACTICE

- ***Work Practices of System Administrators: Implications for Tool Design***
  Nicole F. Velasquez, IBM; Suzanne P. Weisband, University of Arizona

Recognizing that system administrators are special computer users and potentially have different user requirements, the authors conducted field studies in which they shadowed system administrators as they did their work, and they interviewed several administrators to develop a model of user satisfaction geared specifically to system administrators. Their findings indicated that many of the tools available to system administrators were not always practical for their work environments, given the complex, risky, and large-scale operations common in most locations. Although the administrators expressed a preference for CLI tools, the main issue was not the interface technique but the ability to get what administrators want done quickly and accurately. Based on their studies, the authors developed a model that took into account system qualities such as flexibility, scalability, accessibility, speed, and reliability and information qualities such as accuracy, completeness, format, and currency. The authors argued that their model provides an opportunity to improve system management tools by linking system design attributes to end-user satisfaction.

- ***Understanding and Supporting Personal Activity Management by IT Service Workers***
  Victor M. Gonzalez, University of Manchester; Leonardo Galicia and Jesús Favela, CICESE

Multi-tasking is common to all knowledge workers. Those involved in IT services face particularly challenging situations: their work environment is typically crisis-driven, with tight deadlines and long hours. To help IT service workers, the authors developed a personal activity management tool, taking into account the characteristics of IT service work. They developed a workflow model to guide the design of their tool, based on three fundamental aspects of personal activity management: (1) capturing and listing commitments; (2) flexible definitions and execution of work environments; and (3) constant review of commitments. Thus, their model has five meta-activities: capture, classify, focus, manage, and revise, each of which is implemented in separate modules in their tool. In a study with four IT services workers over a period of eight weeks, they found that their personal activity management tool is primarily used as a central repository, complementing email and calendar.

- ***Towards Virtualizing the Helpdesk: Assessing the Relevance of Knowledge Across Distance***
  Kevin F. White, Wayne G. Lutters, and Anita H. Komlodi, University of Maryland, Baltimore County

System administrators are working in increasingly heterogeneous environments, which require in-depth knowledge to manage such complex systems. To increase employee efficiency and reduce costs, managers of IT in large organizations deployed organization memory systems to preserve and reuse expertise within an organization. Small businesses, however, depend on external sources of support. The authors presented research to understand the efficacy of forming partnerships for information-sharing among non-competing businesses, specifically on employee satisfaction with documentation as the distance between information seeker and source increases. They presented findings from their study of five diverse research sites. A semi-structured interview with IT staff included questions regarding origin, clarity, usefulness, quality, accuracy, authority, and competence of the information source. Their findings suggest that in building partnerships it is critical to understand the abilities of employees beforehand and make an appropriate match. Differences in overall quality were argued to result from incongruent levels of education, training in technical writing, and general abilities. Authors also found that in-house documentation often contained subtle contextual cues which may lead to increasing satisfaction when authorship abilities are matched. Thus authors argued that if appropriate training is provided there is more value to fostering internal sharing of documentation.

PAPERS: TOOLS

- ***Sysadmins and the Need for Verification Information***
  Nicole F. Velasquez, University of Arizona, Alexandra Durcikova, University of Arizona

In this paper the authors argue that traditional usability metrics may not be appropriate in system administration. Given that system administrators work in complex and risky environments, they often need powerful but also informative and credible tools. The authors studied the relationship among task complexity, task risk, and information verification activities in GUI tools. They examined whether tasks with greater complexity would lead users to verify information from another source, and, if so, whether there would be an increase in the amount of verification information paralleling task complexity. They also examined if similar hypotheses would hold for task risk. Their findings suggest that higher-complexity tasks led users to seek verification information. Regarding task risks, they did not find a significant factor contributing to information verification. Authors do caution about the small sample size of their experiment but argue that tool designers should consider credibility beyond usability.

- ***Information Displays for Managing Shared Files***
  Tara Whalen, Elaine G. Toms, and James Blustein, Dalhousie University

File-sharing problems in the workplace may hinder collaboration among co-workers and security of information sources. The authors argue that a solution could be to improve presentation of file-sharing settings and activities. They conducted two studies. The first used a group card-sorting activity, an icon-labeling task, and a questionnaire to examine how users currently label file-sharing concepts and interpret icons. Results of the first study suggest that an arrow label did not appropriately indicate file activity and that that there is little overlap between people- and file-related concepts. File-sharing concepts such as “sent” were not clear from an iconic representation alone. The first study was conducted with a small focus group, but the second study surveyed a larger pool. According to the survey for pull-oriented sharing, the words “opened” and “accessed” were very popular, while for push-oriented sharing, there was only one clear favorite: “sent.”

EXPERIENCE REPORTS

- ***Strategies to Influence and Accelerate Adoption of User Centered Design Best Practices in a Company***
  Vijay Agrawal and Ken Chizinsky, Cisco Systems, Inc.

Cisco Systems has over 150 products with user interface aspects, most of which did not incorporate User Centered Design (UCD) methodology in their software development processes, resulting in many inconsistencies in look and in functionality. The goal of the central User Experience Group was to change this by making it easier for development groups to do so. They began by developing a set of application UI guidelines based on a large number of interviews, surveys, and product studies. They also provided consulting services to product teams in order to ensure that the guidelines resulted in maximally usable products. Initially, the group encountered resistance due to increased costs and emphasis on features over UCD compliance. The group devised strategies for overcoming such resistance and lowering the adoption barrier by engaging product teams early in the design cycle, building a design pattern library and support tools that made building compliant interfaces easy and efficient, and creating instrumentation for measuring the positive impact of compliance. These strategies have enabled the group to achieve significant success, with over 40 teams adopting the design patterns and tools.

- ***Configuration Tool Development Experiences***
  Narayan Desai, Argonne National Laboratory

Desai and coworkers are the authors of the Bcfg2 configuration management tool. They have worked on Bcfg2 and its predecessor, Bcfg, for almost six years. The design of the earliest versions focused on implementation of features based on the team’s considerable expertise with system administration. It paid little direct attention to usability and interface design (due in part to their lack of HCI experience). User feedback resulted in substantially more time and effort going toward usability-related aspects of the tool as it developed over time. With Bcfg2, usability became an equal partner to functionality, with the team successfully prioritizing the usability side of design tradeoffs.

PLENARY

- ***Human-Centered Design: Finding the Sweet Spot Among the Many Stakeholders in the Design of a Complex System***
  William B. Rouse, Tennenbaum Institute, Georgia Institute of Technology

Human-Centered Design is a systematic process that ensures balanced consideration of concerns, values, and perceptions of all stakeholders in the design of a complex system. This is important because the success of a product usually depends on a wide range of players, including designers, developers, and, of course, users. The goal of this process is to find a sweet spot that will delight the primary stakeholders and at the same time foster acceptance of the design by secondary stakeholders. Rouse argued for a top-down approach where the issues related to the viability, acceptability, and validation are given appropriate attention at the beginning of the design process. He suggested two principles to guide the design: plan top-down and execute bottom-up. Rouse argued that planning too late and executing too early are typical problems in failed designs.

He suggested four phases: (1) a naturalist phase, in which stakeholders are identified and their roles and concerns are understood; (2) a marketing phase, in which solutions are introduced and viability, acceptability, and validity are planned; (3) an engineering phase, in which issues related to conceptual design and technological realities are sorted and evaluation, demonstration, verification, and testing are planned; (4) a sales and service phase, focused on remediating problems, recognizing further opportunities, and maintaining relationships. Rouse provided three examples to demonstrate human-centered design based on his framework: an intelligent cockpit design, a tradeoff analysis tool, and a strategy and planning tool.

INVITED TALK

- ***Ergonomic Issues in System Configuration***
  Paul Anderson, University of Edinburgh

System configuration is a challenging area in many ways. One of the most basic is the problem of specification. A typical problem is usually expressed in a high-level, general way—e.g., we need another Web server—which must ultimately be implemented by acquiring and configuring a variety of hardware and software components, and then also be maintained over time. The difficulty of this general problem becomes clear when such a scenario is multiplied by thousands of computer systems, each with its own set of tens to hundreds of software applications and the resulting hundreds to thousands of parameters all interacting with one another. Add users and administrators into that mix, and things become very complicated very quickly, generally far beyond the scope and capabilities of the available tools.

Anderson described how configuration languages can make such situations better or worse. He described seven levels of configuration abstraction, ranging from precise and complete specification of each operation at one extreme (e.g., copy this disk image onto those machines, then set these parameters to these values, and so on) to a general statement of requirements at the other end (e.g., configure enough mail servers to provide an SMTP response time of *x* seconds). He went on to discuss the current and potential degrees of automation possible for each level, as well as how configuration management tools can aid in identifying and mediating configuration and functional conflicts. An ideal system would allow for decision-making by both humans and intelligent modules, including cross-vetting of decisions and suggestions.

POSTERS

A diverse set of work presented in the posters session included field research on IT management software deployment, studies on policy-based IT automation, complexity management in middleware systems, challenges in data mining models, design of dashboards for lifecycle management, analysis of workflow management systems, and an analysis of the SAGE salary survey regarding teamwork among IT staff.

PAPERS: SECURITY

- ***Network Authentication Using Single Sign-On: The Challenge of Aligning Mental Models***
  Rosa Heckle, Wayne G. Lutters, and David Gurzick, University of Maryland, Baltimore County

Information security is of particular concern to healthcare organizations. The authors made an ethnographic study of single sign-on technology use in healthcare. They found that often users' mental model of how the single sign-on technology functions was incorrect. They argued that a significant factor contributing to this result was the inappropriate presentation of the technology to healthcare professionals by the IT staff. Specifically, they identified a mismatch between the system model and user mental models, which were formed by users' past experiences, word-of-mouth, and various brochures and presentations during the introduction of the technology. From the beginning, the IT staff took the simplistic view that users only cared whether they would still authenticate as they had in the past; staff therefore focused on backend considerations of the technology. Users, however, believed that they would have one password across all their applications. This led to significant user dissatisfaction with the technology and in turn brought the problem to the attention of the help desk. Recognizing the misalignment of mental models eventually led to improved satisfaction.

- ***Guidelines for Designing IT Security Management Tools***
  Pooya Jaferian, David Botta, Fahimeh Raja, Kirstie Hawkey, and Konstantin Beznosov, University of British Columbia

The usability of security management tools is critical for the effectiveness of security staff. The authors presented a survey of design guidelines for security management tools based on prior work and their own studies. They identified several categories, including task-specific organizational complexity, technology complexity, and general usability guidelines. Those guidelines included: making tools combinable; supporting knowledge sharing; using different presentation and interaction methods; using multiple levels of information abstraction; providing customizability; helping task prioritization; providing communication integration; facilitating archiving; providing appropriate UI for diverse stakeholders; flexible reporting; supporting large workflows and collaboration; making manageable, easy-to-change configurations; and supporting rehearsal and planning, automatic detection, and error messages. They also identified the relationships between these suggestions and provided some information to help users identify the importance of these guidelines to their own tools.

PANEL

- ***Designing for Complexity: New Approaches to System Administration UIs***
  Jeff Calcaterra, IBM Systems & Technology Group; Eddie Chen, BMC; Luke Kowalski, Oracle Corp.; Ian Lucas, Microsoft Management & Services Division; Craig Villamor, Salesforce.com

The closing panel of the conference included senior interface architects and designers from several major hardware and software vendors. The panelists briefly outlined recent interface challenges facing their companies and products, including those related to large-scale computer deployments. Lively audience discussion followed the panelists' presentation, which focused on the trade-offs between command-line and graphical tool designs, the need to design tools that can adapt to changing conditions, and designing future tools in light of the fact that contemporary system administration more often focuses on managing system behavior than on maintaining some specific system state.

The 8th Privacy Enhancing Technologies Symposium

Leuven, Belgium
July 23–25, 2008

Summarized by Rae Harbird (rbird@gmail.com)

This conference was the eighth in a series designed to promote research in privacy enhancing technologies. It marks the transition from workshop to symposium with published proceedings. The conference organizers were keen to maintain the spark and spontaneity of previous years and so kept the "talk for 3 to 5 minutes on anything you like" rump session. The program also included a new feature, HotPETs, in which invited researchers talked about their latest ideas. The social events also provided plenty of opportunity for lively discussion; the banquet featured the PET awards conferred for work that, in the view of the judges, strengthened privacy protection through technology.

PETS/WOTE JOINT SESSION

- ***Using a Touchscreen Interface in Prêt à Voter***
  David Lundin and Peter Y.A. Ryan, University of Surrey, UK

The first morning of the PETs conference was held in conjunction with the Workshop on Trustworthy Elections. In this presentation David Lundin described an extension to Prêt à Voter (PAV), an open-source, electronic voting system, designed to enable the use of a touchscreen interface. Up until now, PAV has used only printed ballot papers, but a touchscreen interface alongside traditional printed voting slips offers improved usability and increased accessibility. Recent PAV developments include the addition of a paper audit trail to support human-readable electronic verification; this has had the side effect of facilitating the use of interactive voting machines.

PAV encodes votes using a randomized candidate list, thus ensuring the secrecy of each vote and removing any bias that might occur with fixed ordering. Conceptually, the PAV ballot has two parts: One is placed in the ballot box and can be used as a Human Readable Paper Audit Trail (HRPAT); the other is used as a protected receipt, which can be printed and kept by the voter, electronically published, and used for vote counting. Both parts of the ballot paper contain an encrypted, randomized list of the election candidates known as an *onion*. The onions are related cryptographically such that it can be clearly shown that one is derived from the other, even when modified with information about the vote cast.

Briefly, the voting procedure is as follows: The voter casts a vote at a terminal, which displays a ballot form generated from a peeled (decoded) onion. The terminal prints a two-part ballot paper, each containing marks indicating the cast vote and an onion. The onion on the receipt part is modified to include information about the vote. The voter retains the receipt and, after sealing the HRPAT, takes it to a teller, at which point the teller posts the ballot (still in the envelope) into a sealed, transparent ballot box. Using the cryptographic relationship between the onions and the candidate list it can be determined that the terminal has not cheated in either printing the candidate list or recording the vote. A threshold set of decryption tellers performs the final stage, decrypting the onion representing each vote.

KEYNOTE SPEECH

- ***Analyzing PETs for Enterprise Operations***
  Stuart Shapiro and Aaron Powell, The MITRE Corporation, USA

Enterprises, as guardians of personally identifiable information (PII) in both the private and public sectors, are beginning to acknowledge the need to address the threats to privacy. There are very few PETs that can support the PII life-cycle management from collection through destruction of that information. PETs do not necessarily help unless they reflect and support the business process of the organization, and this may not involve privacy-specific technologies.

Understanding which technologies can be used to support particular business processes is not always straightforward. The authors have found it useful to categorize specific business processes and PETs in the context of an organization's requirements as a means of assisting the selection and deployment of particular technologies. In general, many PII-related business processes are common across organizations, a lesser number are specific to the type of organization, and a few pertain specifically to the organization. PETs can be categorized according to their primary function: data desensitization or anonymization, identification of PII, those that assist with policy enforcement such as network monitoring and end-point event detection, etc. Once categorization has been achieved, it is much easier to map PETs to business processes. This helps to identify higher-risk business processes, and these can be examined in further detail with use cases delineating precisely how the technology will be employed. It is also much easier to identify potential gaps in the enterprise's processes, ensuring that the enterprise is receiving the maximum business benefit from the technologies used.

In the long term, the authors advocate a more holistic approach, introducing the concept of a Privacy-Enabled Architecture (PEA). We are familiar with using architectures as templates, encapsulating desirable properties, and it should be possible to embed privacy-enabling technologies within a system or enterprise design to achieve comprehensive privacy risk management. There is a clear analogy with service-oriented architectures, which focus on business processes and in which there is a loose coupling of services and specific technologies.

SESSION 1

- ***Perfect Matching Disclosure Attacks***
  Carmela Troncoso, Benedikt Gierlichs, Bart Preneel, and Ingrid Verbauwhede, K. U. Leuven, ESAT/SCD-COSIC, IBBT, Belgium

Anonymous network communication software is used to hide the identities of communicating parties but allow information to be gleaned from analyzing traffic data between them. Disclosure attacks can be used to uncover the relationships between those communicating and the pattern of their communications. In this presentation Carmela described the two contributions of the paper. First, the authors have developed an improved and more realistic user scenario, in which a set of users communicate with each other over an anonymous channel modeled as a threshold mix. Previous workers investigating these attacks only considered a very simple model, in which users choose their communication partners with uniform probability and the effectiveness of attacks investigated strongly relies on this model. The model in this paper relaxes many of the constraints previously imposed, introducing behaviors that are more random in nature.

Second, Carmela described a new attack, known as the Perfect Matching Disclosure Attack (PMDA), which can be used to discover the relationship between messages sent between the set of senders and receivers, enabling attackers to build a profile of users' behavior. PMDA is based on graph theory: communications between users are represented as a maximum weighted bipartite graph. An attacker, observing communications over several rounds, is able to represent the edges in the communications graph as a matrix of log probability values. Subsequently, linear assignment methods can be utilized to find the maximum weight bipartite graph.

The authors assessed the performance of PMDA by comparing it in simulations with the previously published Statistical Disclosure Attack. Results show that PMDA is more accurate when linking senders and receivers. In the future the authors intend to generalize the user communication model yet further by introducing behavior variance over time. They also intend to improve the efficiency of PMDA by parallelizing attacks and by parallelizing the linear assignment problem solver.

SESSION 2: ANALYSIS

- ***Metrics for Security and Performance in Low-Latency Anonymity Systems***
  Steven J. Murdoch and Robert N.M. Watson, Computer Laboratory, University of Cambridge, UK

Tor is the latest generation of Onion Routing software used for anonymous communications over the Internet. This research aims at answering the following question: How do Tor routing decisions affect the risks of communication compromise? Tor directory authorities maintain a list of nodes, with associated attributes, participating in the Tor network. Clients initiate a connection by retrieving information about a set of candidate nodes, known as a consensus directory, from a Tor directory authority. Path selection is carried out subsequently by Tor clients based upon mapping their own requirements to the available node selection algorithms and applying the algorithm to the list of candidate nodes. A user may prefer, for example, a route offering more security from compromise or may choose a route that has more bandwidth capacity.

The authors have developed a Tor path simulator that uses a consensus directory as input with a given number of the nodes within it marked as malicious. Acting as a set of Tor clients, the simulator generates paths with given characteristics (fast or stable). The security of the routes generated is measured as the probability, with respect to the cost and selection algorithm, of connection compromise, characterized by an attacker being able to control the first and last nodes in a Tor connection. The authors evaluated four path-selection algorithms, drawing from current Tor behavior and research in this area. In further experiments the network performance (connection latency) was evaluated using the set of route-selection algorithms. The authors conclude that the results are surprising: within the constraints imposed by experimentation, Tor's default bandwidth-weighted path-selection algorithm offers improved performance in terms of compromise and network latency over the supposedly more secure Tor uniform path-selection algorithms.

PANEL

- ***Data Minimisation, Proportionality, and Necessity in Privacy Systems***
  Moderator: Caspar Bowden, Microsoft UK
  Reported Panelists: Paul de Hert, Vrije Universiteit Brussel, Belgium; Eleni Kosta, K.U. Leuven, Belgium; Gordon Nardell, 39 Essex Street, UK

Caspar launched the panel discussion by introducing the panelists and briefly explaining the theme. Paul de Hert followed, talking about problems with privacy-related legislation in Europe: There is a solid human rights framework, but it allows for many exceptions. Article 8 of the 1950 European Convention on Human Rights (ECHR) is used to protect both electronic communications and equipment; furthermore, most countries have extended these rights in their own legislation. The ECHR also contains a set of very broad exceptions; hence privacy-infringing laws are seldom challenged in court. Moreover, these exceptions are not always applied in a consistent manner. In practice, the proportionality of many new privacy-infringing technologies remains largely unchecked and they are accepted by a simple balancing of interests that always turns out in favor of the government that proposes them.

Paul believes that there are several possible remedies to redress this lack of balance. First, put technology back in context by understanding the values that are at stake. New technologies challenge current value settings; a true balancing exercise should therefore integrate our current understanding of these values and our ambitions to uphold or alter certain settings. Second, we can create new human rights. In the 2001 EU Charter on fundamental rights the right to privacy was complemented with a right to data protection. The presence of this addition forced judges to reconsider the traditional privacy balancing act. Principles such as "purpose limitation" and "consent" do now have a human rights status and their disrespect will necessarily influence a balancing act. Third, more attention needs to be paid to the concept of proportionality. German constitutional law offers a far superior concept. Next to the question "Is this law proportional?" (proportionality in a strict sense), it addresses questions such as "Is there not an alternative that better respects privacy?" (subsidiarity) and "What is left of a specific human right when the governmental initiative to deploy a certain technology gets a green light?"

Gordon Nardell followed. Like Paul de Hert, Gordon noted that Article 8 of the ECHR is a contradiction in some sense, "giving with one hand and taking away with both." Earlier this year the ECHR recognized this dilemma when it gave judgment on a case brought by Liberty and two other NGOs against the UK government. It discovered that the UK Ministry of Defence had intercepted communications that flowed through two of British Telecom's radio stations while in transit between the UK and Ireland. The data gathered was filtered by using search engines and keyword lists and used by intelligence analysts. The statutory procedure involved an application to a minister for a warrant. The problem lay with the analysis of the data after collection rather than with the interception, because a warrant authorized indiscriminate interception of huge numbers of communications, while the process of isolating individual communications, where the real interference with privacy took place, was governed by secret "arrangements," which were not detailed in the legislation. The European court upheld that legal discretion granted to the executive was unfettered and noted that most other European countries publish more information on their respective surveillance laws than the UK. The judgment raises questions over the UK's controversial Regulation of Investigatory Powers Act (2000) and there may be a revision of that law. The UK government should ensure that the use of intercepted data is just as transparent as the acquisition of that data. This ruling may also have an impact on the state's obligation to intervene in cases of private data mining.

Eleni Kosta discussed the European Union's data retention directive of 2006. The purpose of the directive is to harmonize the obligations of Internet Service Providers (ISPs) and telecom operators with respect to the retention of certain data. The directive states that information such as the source, destination, and type and date of communications should be harvested but not the content. Privacy analysts argue that this is not as clear-cut as it might seem. For example, an email address or source information can reveal more information about the participants than is strictly necessary. Data collected must only be provided to competent national authorities, but there is some question surrounding which entities this covers. In the case where a provider shares information inappropriately (i.e., to an authority that is not entitled to receive the data), it will be liable for any resultant damage. The directive has been challenged in front of the European Court of Justice (ECJ): Ireland has asked for an annulment of the legislation on the grounds that it has not been adopted on a proper legal basis. In April of this year, over forty NGOs signed an amicus curiae brief asking the ECJ to annul the EU directive on data retention. They pointed out that, apart from the formal grounds put forward by Ireland, the directive is illegal on material grounds, mainly for infringement of the right to privacy (Article 8, ECHR). In Germany the piece of legislation that transposes the data retention directive into national law has already been challenged in front of the Constitutional Court, which has still not published its decision. However, the Court adopted an interim ruling, stating that data retention as such is not unconstitutional. However, the law implementing the data retention directive does not provide sufficient guarantees concerning the access to the data and the crimes for which data retention may be used. Data retention may only be used for serious crimes and when a judicial warrant is present and therefore the relevant article must be revised.

SESSION 3: ATTACKS

- ***Chattering Laptops***
  Tuomas Aura and Michael Roe, Microsoft Research, Cambridge, UK; Janne Lindqvist, Helsinki University of Technology, Finland; Anish Mohamed, Royal Holloway, University of London, UK

Janne Lindqvist reported the results of an investigation into the information that could be gleaned from wireless-enabled laptop computers by an attacker snooping the packets broadcast by the operating system in order to bootstrap networked system services such as network connection (DHCP), network address resolution (DNS), and network file systems (e.g., NFS). Most operating systems will attempt to initiate these kinds of services as soon as they are switched on and will periodically retry even if failure occurs. Inspection of these preliminary protocol interactions shows that a lot of information is revealed which may enable identification of the service providers (domain name, service details, and server name or IP address) and users (user name, email address, and real name). This information may invite unwanted attention for the user or expose the user's computer to further hacking attacks.

The authors propose a partial solution to this dilemma, namely, policy-controlled use of Network Location Awareness (NLA). Some operating systems allow the user to configure and select one of a set of network profiles. For example, Windows Vista implements the NLA service, which will identify the network without user intervention. NLA creates a fingerprint of the access network based on a set of parameters associated with that network. In the case of authenticated networks the fingerprint might be the security parameters; in all other instances it could be the gateway MAC address. A cryptographic hash is generated from the fingerprint and this is used to identify the network when the user is interacting with it. The next step is to disable and enable service discovery protocols depending on the observed network fingerprint. Some useful default policies were described: NetBIOS should be disabled and enabled separately for individual networks, the default DNS suffix should be disabled outside the domain network, and network file shares and printers should be probed only in the network for which they were originally configured.

HOTPETS

- ***PRAIS—PRivacy impact Assessment for Information Sharing***

Rae Harbird, Mohamed Ahmed, and Anthony Finkelstein, University College London, UK; Andrew Burroughs, Coram, UK; Elaine McKinney, Logica, UK

The UK government is promoting multi-agency information sharing as a key component of new work practices for those providing services to children and families. Despite the plethora of guidance available, staff do not always feel confident to share what they know. This research project has involved a short-term collaboration between computer scientists and child-protection experts. Together they have developed a prototype decision support tool, known as PRAIS (PRivacy impact Assessment for Information Sharing) in the domain of children's social care. The PRAIS system is designed to assist in the decision-making process, not to replace it. Users are not bound to follow the information-sharing actions advocated by PRAIS and staff will be aware that all information-sharing decisions are taken at their own discretion, based on, among other things, their assessment of risk to the individuals involved. PRAIS can also be used as a training tool to help professionals learn experientially about the issues in managing personal information.

PRAIS has been engineered as an expert system, comprising a user interface, a knowledge base containing privacy-related facts and rules, and an inference engine, which can interpret the knowledge base and draw conclusions. The user interface is a Web application representing some of the common work procedures in a social worker's day-to-day tasks that may involve information sharing. The rules are compliant with the UK Data Protection Act and have been reviewed by the Information Commissioner's Office. Project participants are working toward securing a new partner with whom they can develop an operational version of PRAIS and evaluate its efficacy in a realistic environment.

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.

Thanks to USENIX and SAGE Corporate Supporters

USENIX Patrons
Google
Microsoft Research
NetApp

USENIX Benefactors
Hewlett-Packard
IBM
Linux Pro Magazine
VMware

USENIX & SAGE Partners
Ajava Systems, Inc.
DigiCert® SSL Certification
FOTO SEARCH Stock Footage and Stock Photography
Splunk
Zenoss

USENIX Partners
Cambridge Computer Services, Inc.
GroundWork Open Source Solutions
Hyperic
Infosys
Intel
Oracle
Sendmail, Inc.
Sun Microsystems, Inc.
Xirrus

SAGE Partner
MSB Associates

METRO TORONTO CONVENTION CENTRE, CANADA

CONFERENCE & EXPO : 8 APRIL 2009

MAXIMIZE YOUR BUSINESS SUCCESS

UNIFIED COMMUNICATIONS
VoIP
SECURITY
DATA CENTERS

it@360°

Calling all
IT professionals and business executives to experience the latest leading industry advancements

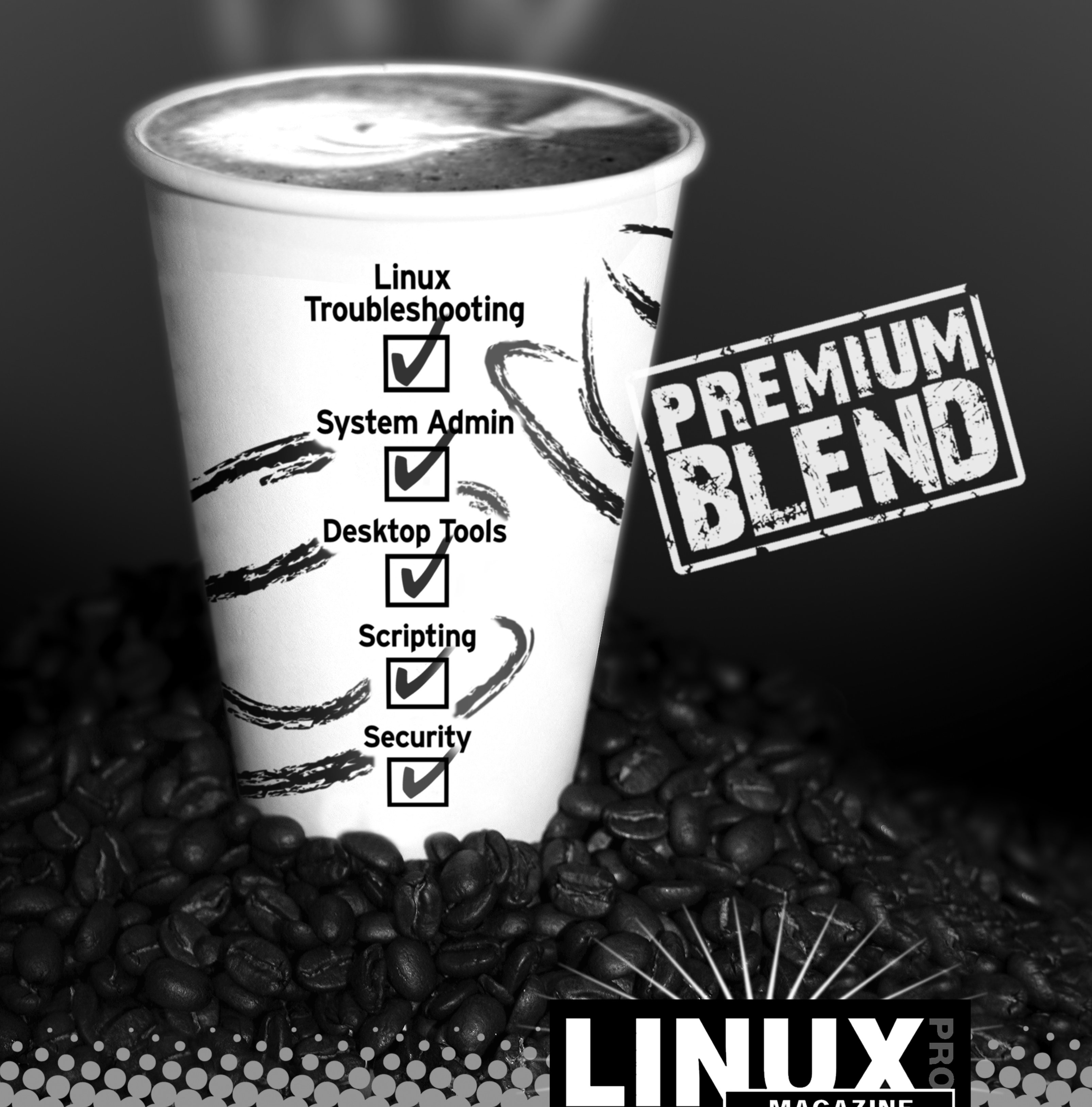
www.it360.ca

PRESENTING: ASTERISK AND OPEN TELEPHONY CONFERENCE

IN COOPERATION WITH: CIO Executive Council — Leaders Shaping the Future of Business

MEDIA PARTNER: itWorldCanada — IDG CANADA

ASTERISK & OPEN SOURCE TELEPHONY CONFERENCE SPONSOR: digium



LINUX PRO MAGAZINE

If you like the taste of Linux, why not treat yourself to the best?

Linux Pro Magazine delivers real-world solutions for the technical reader. In every issue, you'll find advanced techniques for configuring and securing Linux systems. Learn about the latest tools and discover the secrets of the experts in Linux Pro. Each issue includes a full Linux distribution on DVD.

www.linuxpromagazine.com

nsdi '09

6th USENIX Symposium on Networked Systems Design and Implementation

April 22–24, 2009, Boston, MA
sponsored by USENIX in cooperation with ACM SIGCOMM and ACM SIGOPS

NSDI '09 will focus on the design principles of large-scale networks and distributed systems.

Join researchers from across the networking and systems community in fostering cross-disciplinary approaches and addressing shared research challenges.

Don't miss these co-located workshops, both of which will take place on April 21, 2009:
- 8th International Workshop on Peer-to-Peer Systems (IPTPS '09)
- 2nd USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '09)

http://www.usenix.org/nsdi09/workshops/lof

USENIX
The Advanced Computing Systems Association

Register by Monday, March 30, 2009, and save! http://www.usenix.org/nsdi09/lof

;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