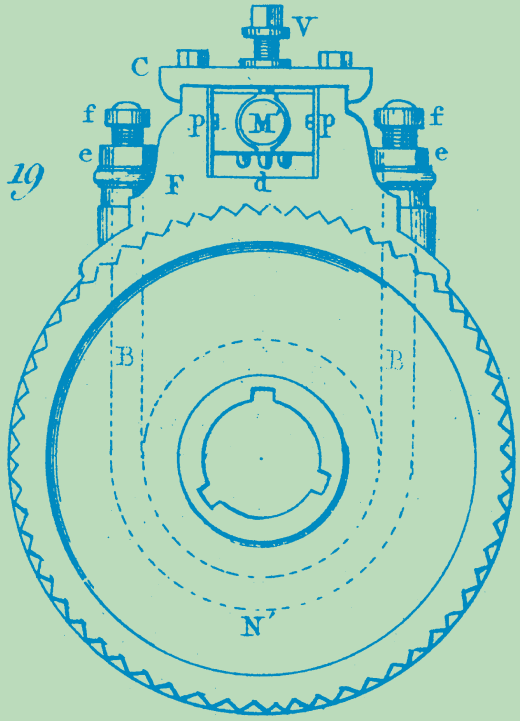
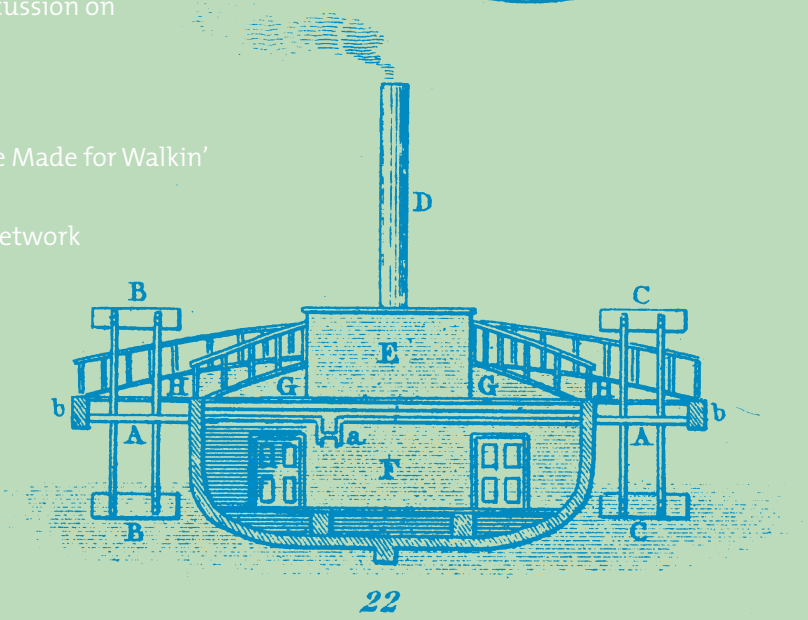
APRIL 2007 VOLUME 32 NUMBER 2

;login:

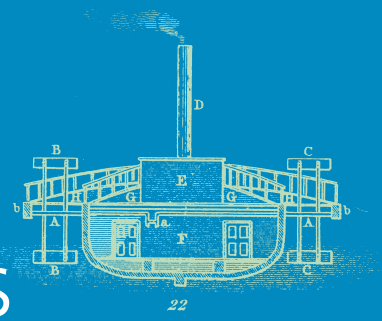
THE USENIX MAGAZINE





USENIX
The Advanced Computing Systems Association

USENIX Upcoming Events



11th Workshop on Hot Topics in Operating Systems (HotOS XI)

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

MAY 7–9, 2007, San Diego, CA, USA
http://www.usenix.org/hotos07

5th ACM/USENIX International Conference on Mobile Computing Systems, Applications, and Services (MobiSys 2007)

Jointly sponsored by USENIX and ACM SIGMOBILE, in cooperation with ACM SIGOPS

JUNE 11–15, 2007, Puerto Rico
http://www.sigmobile.org/mobisys/2007/

Workshop on Experimental Computer Science (ECS '07)

Sponsored by ACM SIGARCH and ACM SIGOPS in cooperation with USENIX, ACM SIGCOMM, and ACM SIGMETRICS

JUNE 13–14, 2007, San Diego, CA, USA
http://www.expcs.org/

Third International ACM SIGPLAN/SIGOPS Conference on Virtual Execution Environments (VEE '07)

Sponsored by ACM SIGPLAN and ACM SIGOPS in cooperation with USENIX

JUNE 13–15, 2007, San Diego, CA, USA
http://vee07.cs.ucsb.edu

Second Workshop on Hot Topics in Autonomic Computing (HotAC II)

Sponsored by IEEE in cooperation with USENIX and ACM

JUNE 15, 2007, Jacksonville, FL, USA
http://www.aqualab.cs.northwestern.edu/HotACII/

2007 USENIX Annual Technical Conference

JUNE 17–22, 2007, Santa Clara, CA, USA
http://www.usenix.org/usenix07

3rd Workshop on Steps to Reducing Unwanted Traffic on the Internet

Co-located with the 2007 USENIX Annual Technical Conference

JUNE 18, 2007, Santa Clara, CA, USA
http://www.usenix.org/sruti07
Paper submissions due: April 17, 2007

Inaugural International Conference on Distributed Event-Based Systems (DEBS 2007)

Organized in cooperation with USENIX, the IEEE and IEEE Computer Society, and ACM (SIGSOFT)

JUNE 20–22, 2007, Toronto, Canada
http://www.debs.msrg.utoronto.ca

Third Workshop on Hot Topics in System Dependability (HotDep '07)

Co-sponsored by USENIX

JUNE 26, 2007, Edinburgh, UK
http://hotdep.org/2007

2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)

Co-located with Security '07

AUGUST 6, 2007, Boston, MA, USA
http://www.usenix.org/evt07
Paper submissions due: April 22, 2007

16th USENIX Security Symposium

AUGUST 6–10, 2007, Boston, MA, USA
http://www.usenix.org/sec07

2007 Linux Kernel Developers Summit

SEPTEMBER 4–6, 2007, Cambridge, U.K.

21st Large Installation System Administration Conference (LISA '07)

Sponsored by USENIX and SAGE

NOVEMBER 11–16, 2007, Dallas, TX
http://www.usenix.org/lisa07
Extended abstract and paper submissions due: May 14, 2007

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

contents

;login:

VOL. 32, #2, APRIL 2007

EDITOR
Rik Farrow
rik@usenix.org

MANAGING EDITOR
Jane-Ellen Long
jel@usenix.org

COPY EDITOR
David Couzens
proofshop@usenix.org

PRODUCTION
Lisa Camp de Avalos
Casey Henderson

TYPESETTER
Star Type
startype@comcast.net

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

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

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

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

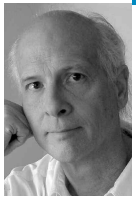
$85 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $115 per year.

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

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

RIK FARROW

musings



rik@usenix.org

IN MY LAST COLUMN, I ARGUED THAT we need to be willing to change. Change is inevitable. Just as the seasons change and the climate warms, our computing environment also changes, slowly enough that we sometimes miss noticing the changes until we run smack up against them.

Let's consider one change in the world of computing, and see how it relates to a topic that is only tangentially related—system administration.

One of the hot computing news stories eight years ago (1999) was the race to produce a desktop PC for less than $1000. These computers came with 366MHz Celeron CPUs and no monitor [1]. Fast-forward less than ten years, and you can buy a system with a dual-core, 64 bit, 2-GHz CPU with a 19-inch LCD monitor and one hundred times the storage capacity for under $700. I'd like to ignore discussions of Moore's Law and cut to the chase. Today's base system has capabilities generally undreamed of just ten years ago. Running multiple VMs on desktop systems is not only feasible but not uncommon.

Fertility

System administrators have more than high-performance desktops in their basket full of changes. I mentioned VMs, which seem to multiply like rabbits in the springtime. VMs encapsulate whole operating systems as well as at least one application, yet they still must be managed and configured, and the data—often just the data—must be properly backed up. So one system today may represent not just a faster system but many multiples of past systems.

What is true of desktops is true of servers as well. Servers have become just as fecund, if not more so, than desktops, expanding to fill ever-shrinking machine rooms, even as they release ever greater amounts of "waste" heat. It is not the energy by-product of servers that concerns me here, but their ever-increasing capacity for fruitful labor. Servers too become targets for VMs, so that every last erg of energy will be useful, and not simply add to the entropy of the universe and deplete our fast-declining store of fossil fuels for no good purpose.

Once upon a time, system administration meant editing configuration files, using your text editor of choice. Today, the hand-editing of files has gone

the way of the oxcart, with the rare exception of systems that are just too different to be managed en masse. As system administrators, we are faced with a choice: to become equally obsolete, or to change with the times and become meta-administrators instead.

Going Meta

One of the joys of being an editor is the ability to spout silly ideas, even attempt to coin words that may help us understand the world we live in. Meta-administration means system administration that has moved beyond the editing of configuration files. The knobs in those files remain there, just as adjustable as ever. What has to change is how we manage to tweak those knobs. It has always been difficult to know how to manage multiple versions of UNIX, where the configuration files may not be the same (think AIX or Mac OS X) and small changes may have large unintended consequences. Now we have multiplied the number of systems we must manage, even as the speed of those servers and desktops has grown exponentially. To expect that we can hand-edit configurations, or even launch a script that will zoom across the network making the same changes everywhere, is no longer plausible.

Meta-administration means that we will be providing guidelines for how the system of systems should behave. It does not mean that we will never again edit a configuration file. Hardly that. The deep knowledge will be more important than ever, because it will be skilled practitioners who will still be needed to solve the intractable problems that arise. But much of the day-to-day management of systems will be left to agents, guided through configuration management techniques that we are still evolving today.

The Lineup

This issue of *;login:* has been dedicated to the inevitability of change in system administration. When I first approached the concept of system administration, I had little notion of what was involved. Happily, no one else did either at the time (1983), so I was not alone in my ignorance. I thought that all I needed to do to write a book to guide future system administrators was to go to UC Berkeley and interview real, live working sysadmins to understand what it was that they did. I did make that trip in 1985 to Evans Hall, but I discovered that the sysadmins working there had no idea what they were doing. I don't mean to say they were clueless, because any one of them could hand-edit configuration files with the best of us. But those UCB sysadmins had no knowledge of the bigger picture, that is, what it meant to be a system administrator. They merely carried out assigned tasks, such as adding user accounts, managing printer queues, and handling backups.

If you have managed to read this far, you have, hopefully, come to the same conclusion I have: those days are over. Although you may know a lot about what goes on "under the hood," the primary task of system administration has become meta-administration.

In that vein, we begin this issue with an article by Glenn Fink and Deb Frincke about autonomous systems. Glenn and Deb start off with the notion that we will be using autonomous agents to manage not just our systems but aspects of our networks as well, and they examine the consequences of utilizing agents. For example, what does it mean when an agent "decides" to purchase increased bandwidth from your ISP to handle an

expected increase in network traffic to maintain a service-level agreement? The agent made that buying decision, but you will be paying for it. Ultimately, who is responsible for the actions of autonomous agents?

Chad Verbowski follows up with an article about Flight Data Recorder (FDR), a new feature in Vista, which someday will be found in the next server version of Windows as well. Chad had delivered a paper about FDR at OSDI '06, but I wanted more details, and Chad has them here. Essentially, FDR provides the administrators of Windows systems with fine-grained details about significant changes to those systems, using a clever method for compressing the huge amount of log data before it ever leaves the Windows box, yet leaving that compressed data in a form that can be rapidly searched for nuggets of key information.

In the next article, Brendan Quinn shares his experiences in working with LDAP. LDAP means different things to different vendors, and Brendan provides useful hints on surviving conflicting expectations of those vendors and still having a useful directory service.

John Lloyd next attempts a difficult feat by explaining system engineering in the short space permitted (and goes a bit over the limits I generally enforce in terms of page count). I found John's information interesting and useful, both in the real terms of designing systems that will work the way you expect them to and in coming to true agreements about what management/powers-that-be expect those systems to do. If you have ever gotten into trouble when building systems for some specific task, you owe it to yourself to read this article.

Although Mark Burgess had completed his series about configuration management, I felt he had left some aspects uncovered. After a short pursuit, Mark agreed to be interviewed about his concept of "promises" and how they fit into the world of configuration management. Promises fit in very well with my own concept of meta-administration, and they had a strong hand in my creation of this column. Mark also relates Alva Couch's closures as well as Paul Anderson's aspects to promises, so you can at least begin to understand where some significant players in the configuration management community are heading.

Dan Appelman follows up with significant advice for any system administrator involved in the management of mail servers. Gee, did I manage to leave any sysadmin out there? Based on his tutorial at LISA, Dan describes some of the impact of U.S. law, as well as actual case histories on the uses and abuses of email. Dan warns us that anyone who sends out email may run afoul of anti-spam legislation and can suffer the legal and financial consequences of doing so.

Thomas Sluyter shares helpful tips for sysadmin consultants. Thomas explains how to produce regular reports as you carry out contracts, so that management is constantly aware of both your progress and any significant hurdles in the way of progress. Even if you are not contracting sysadmins, I highly recommend his advice to you. Keeping a log of the work you do, along with the tasks you have been assigned to do, will prove helpful when your next job review comes around.

David Blank-Edelman cheerfully describes climbing down trees in Perl. The trees are file systems, and the task is a common one for any system administrator. Robert Haskins writes about ways of providing VoIP at the provider level, and Heison Chak tells you how to reflash a cheap file server to turn it into an Asterisk server—PBX on the cheap. Robert Ferrell then provides advice on the care and feeding of sysadmins.

Elizabeth Zwicky follows up with her usual book reviews, that is, reviews that are honest and funny at the same time.

Nick Stoughton takes a stand against the abuse of standards and standards bodies. His target, OOXML, gets revealed not as a standard but as a statement of monopoly control with the sole purpose of providing a large vendor with a continued revenue stream. I personally would be more upset if abuses of power had not become so commonplace these days. But Nick makes a compelling case for the sometimes capricious nature of standards bodies.

The USENIX Notes section fills you in on the latest news about SAGE: The USENIX SIG for Sysadmins, and about the upcoming LISA '07 conference, and asks for your advice and assistance.

This issue concludes with summaries from LISA '06, including two workshop summaries (Configuration Management and Advanced Topics). If you attended a workshop and don't see your summary here, it's not because we aren't interested in reading it—it's just that no one provided us with one to include.

As the world turns, day becomes night, winter warms into spring, and fall cools into winter. The world of computing races forward as well, with ever-increasing complexity. Old-style system administration provides one with a certain comfort, as you can see the changes you make, tests their effects, and easily adjust your changes if necessary. In the new world, that level of comfort is fast disappearing, and the need for abstracting changes to support flexible control of more systems than ever has become a requirement. Don't get left behind.

REFERENCE

[1] List of desktop systems, capacities, and prices from fall 1999: http://www.cs.umd.edu/hcil/academics/courses/fall1999/cmsc838s/Apps/jintong/all.csv.

GLENN FINK AND DEB FRINCKE

autonomic computing: freedom or threat?

Glenn Fink is a Senior Research Scientist at Pacific Northwest National Laboratory (PNNL) in the Cyber Security group. His research interests include the effects of technology on the humans who use it.

glenn.fink@pnl.gov

Deb Frincke is a Chief Scientist at PNNL and the founder of the Intrinsically Secure Computing initiative there. Before coming to PNNL she was a faculty member at the University of Idaho, where she specialized in forensic research.

deborah.frincke@pnl.gov

NO LONGER IS THE QUESTION *WHETHER* autonomic computing will gain general acceptance, but *when*. Experts such as Alva Couch expect autonomic computing to be widely used within 10 years [1]. When it does become mainstream, how will autonomics change system administration and corporations, and will the change be for better or worse? The answer depends on how well we anticipate the limitations of what autonomic systems are suited to do, whether we can collectively address the vulnerabilities of autonomic approaches, and whether administrators, companies, partners, and users are prepared for the transition. In this article, we present some design considerations to address the first two issues, and we suggest some survival techniques for the third.

What Is Driving Autonomic Computing?

Computing systems used to have reasonably well-defined borders, both geographically and logically. Today, corporate, educational, and even government computer-based systems coexist in an open mesh of overlapping infrastructures. The complexity of these networks and of the systems that comprise them have given rise to the need for more automation in their configuration and management. Companies rely on a growing number of increasingly complex systems. Increased interconnectivity has led to increased exposure to attacks from around the world. Speed and frequency of attacks have continued to increase exponentially, with malware capable of saturating the Internet in minutes. The rise in numbers, increase in complexity, and need for quicker protective actions all point to a need for additional automation.

At the same time, because companies cannot afford to hire and appropriately compensate the required number of skilled workers to handle this complexity, they have increasingly resorted to offshore outsourcing and commoditization of system administration in recent years to save labor costs. All of this explains why self-managing, intrinsically secure computing is an attractive notion. The idea of systems that can take care of themselves can be either a dream come true or a nightmare, depending on your perspective. There are many stakeholders, including (1) owners of the systems looking for savings in time or money,

(2) system administrators who face the tension between enjoying the freedom from drudgery that autonomic systems promise and the worry that this freedom will ultimately make their jobs unnecessary, (3) users who are looking for increased work efficiency from the systems, (4) business partners who share the networks, resources, benefits, and risks of autonomic systems, (5) legal counsel who will need to sort out responsibility and liability in the new world, and, last but not least, (6) attackers who will view autonomic systems as either an effective barrier to their access or as a great way to bend the systems to their will automatically and invisibly.

Autonomic Computing: Freedom?

Autonomic computing derives its name metaphorically from the operation of the human autonomic nervous system. Autonomic systems are intended to be self-managing, keeping mundane details of operations hidden from the operator while increasing predictability, speed of response, and reliability. The idea of pervasive computing, where tiny networked computers embedded in the environment will constantly adjust to our needs, requires autonomic computing. IBM has defined the crucial elements of autonomic systems in their autonomic computing manifesto [2]. In a following paper, Kephart [3] describes an inspiring vision of what autonomic computing will do for technology and society.

Autonomic computing promises a more natural boundary around the complexity of the systems we live with today. People don't generally consciously interact on the cellular or atomic level with others; we interact at the natural boundary that separates one person from another. As the internal complexity of computational systems increases, a new boundary between computers and human administrators becomes necessary. People shouldn't have to fiddle with the vagaries of configuration files any more often than they should have to modify their kernel source code. Looked at this way, autonomic computing is a natural and necessary way to internalize and compartmentalize complexity.

If the vision of autonomic computing becomes reality, systems will be self-managing so that the administrator won't have to be summoned on an emergency basis nearly as often. Kephart's autonomic systems may not only patch themselves but also automatically seek updates, new software, and better configurations that will give them better performance (Figure 1). They will find workarounds when services they depend on break. Autonomic systems will negotiate service agreements with external systems, using and providing services whenever it is consistent with system goals, and they will protect themselves when they sense that they are under attack.

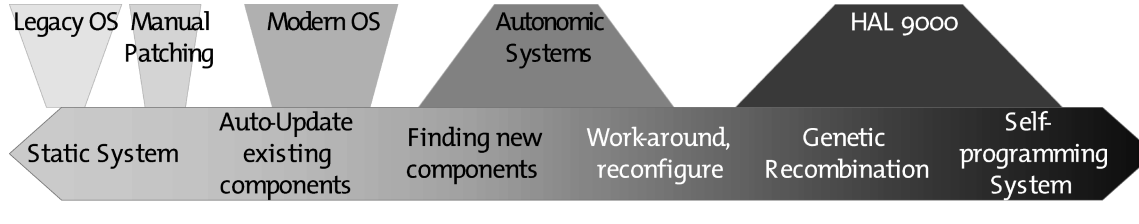


FIGURE 1: THE SELF-MAINTENANCE CONTINUUM

But here we begin to see that the idea of autonomic computing has gone far beyond the metaphor offered by the human autonomic nervous system. There seems to be a subtle difference between what we mean by "autonomic" and the meaning of "autonomous." "Autonomic" traditionally means

that normal operations will continue without interruption or need for intervention. But "autonomous" goes beyond normal operations, implying a sense of self, needs that must be met, and freedom to act. In humans, autonomy implies individual actions springing from intelligence and a will free from conscious external coercion. Kephart's vision for autonomic computing reaches beyond the accepted definition of "autonomic" into territory much better described by the word "autonomous." Is this more freedom than we want our systems to have? Or is it the freedom they *must* have if they are to be what we need them to be?

Autonomic Computing: Threat?

The very freedoms that we expect to gain from autonomous technology may also be cause for concern. An autonomic system providing services to humans or other systems is different from an autonomous system working as an agent on behalf of a human or organization. They are different in the degree of independence they are afforded and in the way accountability and control are administered. Millions daily give eBay's auction software the permission to commit to spending funds within preset limits. This requires eBay users to seriously consider beforehand whether they are willing to buy the products they bid on. Once the user's software agent wins the auction, the user is obligated to pay. Similarly, when allowing systems to autonomously seek new software, enter service agreements, and protect themselves, we must be sure we are able to bound the consequences and that we are willing to pay the potential price.

If autonomic systems are given enough freedom to act and interact, the overall infrastructures may behave in emergent ways we cannot predict. Autonomy means that decisions will be made locally, but the emergent qualities of complex systems demonstrate that local decisions can have far-reaching and unpredictable global results. As Kevin Kelly so poignantly puts it, "Wherever the word 'emergent' appears, there disappears human control" [4]. Automation is one thing, but autonomy is quite another. Once systems become autonomous, by definition they will have a "mind of their own." We will be asking software to make decisions for us that have traditionally been entrusted to humans alone. This is monumental in stand-alone systems, but in the world of overlapping corporate infrastructure boundaries and numerous (and potentially conflicting) stakeholders, the implications are astounding.

Consider a scenario where a self-managing system is empowered to negotiate with an ISP about the amount of bandwidth the owner's commercial storefront Web servers require. The policy enforced by the self-managed system may include "spend the least amount of money that supports peak anticipated access rates based on trends over the previous seven days." If there is a surge in accesses, the system will make the decision to commit corporate dollars to expand bandwidth. If, instead, the access demands have reduced, it may decide to "save money" by reducing the amount of ISP service based on lower access rates and slow sales. Notice that the policy as written fails to take into consideration peak demands such as holidays or the sudden popularity of an item.

The previous situation was fairly straightforward, but weightier agreements between suppliers and purchasers could have more far-reaching economic effects. When autonomous software agents are making the deals, who is legally bound to meet the terms of the agreement, and how can all parties

be made aware of their obligations? How do the humans renegotiate a promise that their automated systems made, and what are the legal, societal, and logistical implications of such intervention? Who decides how much authority to delegate to these systems, and how much can a partner trust the authority delegated? Service-level agreements made by autonomic systems will have economic impacts that imply risks for multiple stakeholders. Before an organization decides to trust a promise made by an autonomic system, it will have to be able to verify that the system's word is worth the risk [5]. And when things *do* go wrong, how will organizations debug the policy language that allowed the problem to occur?

Another aspect of self-management that must be considered is the continuum of potential response to threat—sometimes referred to as active defense or active response [6]. Different organizations have different approaches to defending their infrastructure. Thus, it is reasonable to assume that different defensive policies will be enacted. The impact of executing divergent, autonomously enforced policies within a mixed cyber infrastructure is difficult to express [7]. Consider a virtual community such as the open science grid computing community. Suppose one organization has a policy that it disconnects from the larger network when it detects wormlike activity. What does this do to another organization that may have real-time dependencies on the first organization's services? How about an organization whose policy is to avoid negative effects on others? Given the overlapping nature of dependencies in cyber infrastructures, it will be difficult or impossible to automatically (or any other way) verify that there is an acceptably low level of negative consequences stemming from a policy change. Will this cause autonomous systems to be paralyzed into making no decision at all? Clearly, policies must not be conceived in a vacuum without a consideration of their wider effects on other organizations.

The existence of intelligent attackers brings to light the need to verify that the autonomic system hasn't been subverted and is still acting within the intentions of its owners. Autonomic computing may lead to true complex-adaptive systems whose ultimate behavior is very difficult to predict from initial conditions [8]. One thing about attackers is certain: They will learn to adapt to autonomic systems and to bend them to their will. Another certainty is that attackers will want to keep their activities secret. Autonomic computing brings to attackers the promise that no human will be watching. There must be a way to allow human administrators the ability to inspect the operation of the system and verify that it is still on their side.

Computer programs don't go to jail. They aren't afraid of losing their jobs. But when things go wrong and the cost of a bad decision is estimated, it is certain that some human(s) will pay the price [9]. If system administrators are to be held responsible for the decisions of autonomous systems, then both the responsible persons and their employers will want to ensure that there is the possibility of some human awareness and intervention in these decisions. At the same time, humans will not want to be involved in every decision—that would make autonomous systems pointless. We will need to rethink the meaning and limitations of trust in the new world of autonomous systems.

Design Considerations for Autonomic Systems

The open issues for self-managing systems go well beyond the scope of a single article, and each issue has implications for design. There are, however, a few considerations that we can suggest to manage at least some of the

risks this article has broached. We see five broad areas where design guidelines would help:

- Awareness: Allowing human insight into system activities via cyber analytics.
- Management: Enabling human influence over distributed autonomous systems via hierarchical design.
- Attribution: Certifying the correctness of independent actions of the system.
- Integrity: Ensuring that the system has not been subverted.
- Limits: Stating clearly what the autonomous system may and may not do.

AWARENESS

The emerging discipline of *cyber analytics* (the application of visual and predictive analytics to understanding the workings of computer infrastructures) holds great promise for humans to gain and maintain awareness in the face of overwhelming amounts of data. But awareness alone is not sufficient to control large distributed systems. Coupling autonomous control with the situational awareness of cyber analytics will allow humans the ability to manage an unprecedented number of computer systems. We have shown that human oversight is essential, even if the systems work flawlessly. Thus we propose that designers of autonomic systems keep human awareness and management in mind even as they design their systems not to require human intervention.

MANAGEMENT

Humans need a single point of influence to have multiple points of effect within their systems. We only have at best ten fingers and one brain, but we need to be able to exert consistent influence over large numbers of heterogeneously configured systems simultaneously. This is the point of policy (and, by extension, autonomic computing). But centralized control is not the answer. Large, centralized artificial intelligence becomes brittle and computationally intractable for distributed, highly constrained problems. Other solutions, such as swarming intelligence, are useful for such problems [10], but they are very difficult to understand and control. We suggest that hierarchical deployment of a variety of intelligent agents [11] will provide both the single point of influence and the multiple points of effect needed. The highest-level agents can translate the activities of a swarm of "digital ants" to the human and can implement the user's policy via lower-level agents. We believe that a hierarchy of varied intelligent agents will increase human influence while reducing the need for human intervention.

AUTHORIZATION

Actions that involve agreements across organizational boundaries require some way to distinguish the activities of the autonomic systems from those of the humans who are ultimately responsible for them. Successful delegation of high-level duties will require separate digital identities for the human supervisor and the autonomic systems and digital reputation accounts for the autonomic systems. Systems could be "punished" or "rewarded" via feedback from other systems and from their owners, enabling machines to learn from their mistakes. Similarly, if the system acts outside its authority, its own signature would be on the agreement, allow-

ing the responsibility to be properly allocated. Attribution of responsibility is also a key to debugging these complex systems. These mechanisms certainly don't solve the technical, legal, and social problems raised by an interacting society of autonomous systems, but they do lay some groundwork that may make such problems solvable.

INTEGRITY

Autonomic computing will spare humans many details they simply have no time for, but attackers can turn this information hiding to their advantage. Additionally, adaptive systems will be able to change the way they behave, and possibly even their own behavioral parameters. Autonomic systems must act in a predictable manner, even when portions of their systems are actually subverted by attackers. Thus, it is important that system designers provide an ability to make sure that autonomous systems are acting in accordance with stated policy and the intent of their owners. Cryptographic methods may be employed in a number of ways to check the integrity of autonomous agents, while static code verification may help assure adherence to policy.

LIMITS

Another important facet of autonomous system assurance is making policy limitations expressible in terms that are human-understandable, complete, and translatable down to the machine instruction level and back. Natural language is ambiguous and hard to parse. XML is "human-readable" only in the sense that it is expressed in printable ASCII. Much research in policy languages remains to be done to achieve assurance that policy is expressed correctly and can be executed as expected.

At Pacific Northwest National Laboratory (PNNL), we are designing autonomic systems for computer security. Our Intrinsically Secure Computing [12] initiative embodies three core principles: Trustworthy Engineering, Adaptive Defense, and Appropriate Response. We believe that at least some of what we are learning in the security arena is applicable to autonomic systems in general. As part of this initiative, we are building a human-agent defense grid that will enable greater levels of autonomous behavior in system defense without losing sight of the fact that humans are ultimately responsible for the activities of their systems. We intend to apply our findings broadly to autonomic systems of other sorts in the hope that these systems will be a blessing and not a curse to the administrators who use them.

Conclusions: Thoughts on Life in an Autonomic World

In conclusion, let us consider what the advancement of autonomic computing will mean for system administrators as a profession. Do self-managing systems pose a threat to system administrator job security or a promise of increased job satisfaction? This question has been asked in many forms. In a panel discussion at ICAC 2006 [13], Kumar Goswami stated that barriers to autonomic system administration might arise from concern over lack of trust in the system, loss of hands-on control, and fear that automation would eliminate the system administrator's job.

In the 1820s, during the Industrial Revolution, debate on the "Machinery Question" was even hotter than it is now. People feared that machinery

would replace human labor and result in widespread unemployment and poverty. This fear proved to be short-sighted because, after automation, a skilled workforce was needed to make repairs and manage machinery [14]. Factory jobs that were already highly mechanical were taken over by machines, but new jobs that required human capabilities were created. Industry expanded, with machines doing more than humans ever did before. Human workers had to develop new skills and gain education to remain competitive (see Figure 2), because the rates of job automation and job creation were not tightly tied. While ours is a different age, many similar forces are in play, so the comparison has merit.

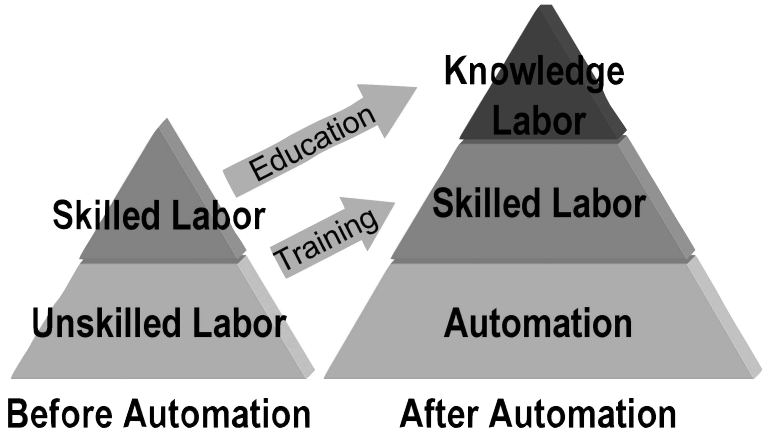


FIGURE 2: EFFECTS OF AUTOMATION ON THE HUMAN WORKFORCE

With the advent of autonomic computing, human system administrators will definitely not become obsolete. However, autonomics will significantly change the way administrators work. Some of the push toward autonomic computing comes from corporations that are unable to hire enough qualified system administrators right now at rates they can afford. We expect that autonomic systems can increase productivity and produce the corporate capital needed to alleviate the existing problem, not put people out of jobs. But this result will require time for society to find a new equilibrium.

During rapid changes in technology, "[p]erhaps the best skill . . . is how to learn (and unlearn) quickly" [14]. Historically, technological leadership has always been hard to sustain, and this will be as true for the technocrats of today as it always has been. Arguably, Britain lost the technological leadership it enjoyed during the first part of the Industrial Revolution because it clung to the products and processes that had made it great rather than adapting to new ways. This is a lesson that all technologists would do well to note: Adaptability is the best defense in a changing world.

Autonomic systems will take over the "plumbing," enabling humans to work at the higher, policy level. As long as humans are responsible for information systems, administrators will be needed to translate operating policies and business practices into clear, complete, and consistent machine instructions. The only difference is that machines are learning to understand language closer to the way humans are accustomed to expressing it. Machines will also learn to find inconsistencies in policy and ask intelligent questions about them. But there is still a place for human system administrators to act as go-betweens for management and machines. And there will always be a place for highly skilled individuals who can "look under the hood" when things *do* go wrong.

Management is good at comprehending business objectives. Machines are good at executing programs. System administrators must learn to translate

business logic into policy logic for machines. System administrators will have to interface well with both humans and machines. The first generation of autonomic systems is already being put into use, and wise system administrators will learn how they work early on. There are two reasons for this: (1) because the technology will be demanded by their employers sooner or later, and (2) so that administrators can join the nascent autonomic computing design dialogue.

Some system administrators see autonomic systems as a threat to their employment rather than liberation from drudgery. We believe this is an unfounded fear if administrators are willing to make some adjustments: Let go of the need to control the details of low-level configuration, trust but verify, and learn to understand the business needs that will drive policy.

System administrators should consider autonomic computing to be a promotion to a position of more responsibility and respect. Administrators will become management consultants rather than technicians. No technology can live without highly skilled troubleshooters, but the numbers of these professions will likely dwindle as autonomic systems improve. We believe the overall number of system administrators will probably not decrease, because the number of machines being fielded and placed on the Internet is increasing exponentially. No matter how smart the machines get, there will always be a place for intelligent, adaptable, human system administrators.

REFERENCES

[1] A.L. Couch, "The Future of System Administration: How to Stop Worrying and Love Autonomic Computing," Presentation at LISA '06. Available at http://www.usenix.org/events/lisa06/tech/slides/couch.pdf.

[2] IBM, "Autonomic Computing: IBM's Perspective on the State of Information Technology," 2001. Available from http://www.research.ibm.com/autonomic/manifesto/autonomic_computing.pdf.

[3] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," *IEEE Computer*, pp. 41–50 (January 2003). Available at http://www.research.ibm.com/autonomic/research/papers/AC_Vision_Computer_Jan_2003.pdf.

[4] K. Kelly, *Out of Control: The New Biology of Machines, Social Systems and the Economic World* (Perseus Books, 1994). Available at http://www.kk.org/outofcontrol.

[5] E.A.R. Dahiyat, "Intelligent Agents and Intentionality: Should We Begin to Think Outside the Box?" *Computer Law & Security Report*, 22(6): 472–480 (2006). Available at http://dx.doi.org/10.1016/j.clsr.2006.09.001.

[6] S. Caltagirone and D.A. Frincke, "The Response Continuum," *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, 15–17 June 2005* (IEEE Press, 2005). Available at http://www.classstudio.com/scaltagi/papers/professional_papers/westpoint05v2_2.pdf.

[7] D.A. Frincke, A. Wespi, and D. Zamboni, "From Intrusion Detection to Self Protection," Computer Networks, 2007.

[8] J.H. Holland, *Hidden Order: How Adaptation Builds Complexity* (Addison Wesley Longman, 1995).

[9] M. Scher, "On Doing 'Being Reasonable,' " *;login:* 31(6): 40–47 (2006). Available at http://www.usenix.org/publications/login/2006-12/pdfs/scher.pdf.

[10] H.V.D. Parunak, " *'Go to the Ant':* Engineering Principles from Natural Multi-Agent Systems," *Annals of Operations Research* (Special Issue on Artificial Intelligence and Management Science), 75: 69–101 (1997).

[11] H.V.D. Parunak et al., "Hybrid Multi-Agent Systems," *Proceedings of the Fourth International Workshop on Engineering Self-Organizing Systems (ESOA '06)* (Hakodate, Japan: Springer, 2006).

[12] Pacific Northwest National Laboratory, Computational & Information Sciences Directorate, "Intrinsically Secure Computing" (2006). Available at http://cisd.pnl.gov/secure_computing.stm.

[13] Panel Report: O.F. Rana and J.O. Kephart, "Building Effective Multivendor Autonomic Computing Systems," *IEEE Distributed Systems Online* 7(9) (2006).

[14] J. Mokyr, "Are We Living in the Middle of an Industrial Revolution?" *Federal Reserve Bank of Kansas City Economic Review*, pp. 31–43 (1997). Available at http://12.154.48.181/PUBLICAT/ECONREV/pdf/2q97mokr.pdf.

CHAD VERBOWSKI

the secret lives of computers exposed

FLIGHT DATA RECORDER FOR WINDOWS

Chad Verbowski is the cofounder of the Cybersecurity and Systems Management research group, where he focuses on reducing complexity and improving the security, reliability, and efficiency of software and integrating the results into the next generation of Microsoft products. Before joining Microsoft Chad worked extensively in the systems management problem space, leading the development of flagship management products at MFS Datanet, Cisco Systems, and Manage.com.

chadv@microsoft.com

WE'VE ALL HAD THE GIDDY EXPERIENCE of setting up a new system and being impressed by our newly acquired performance and capability. Inevitably though, as time goes on, our new system has less time for doing our bidding and assumes a life of its own—hard drives grind for no apparent reason, it is achingly slow or stalls altogether despite available resources, or applications and devices no longer work as they once did. Are these the result of unwanted users or software wooing my system—or did I do something to disrupt the delicate fabric of state stored within? With the ever-increasing spare time gleaned from waiting on my nearly new system, I pondered these issues and decided to put together a plan to spy on the secret life of my computer. What you are about to read may not be the information you need to be the life of your next party, but it will help you win back the attention of your computer.

A New Surveillance Gadget—Flight Data Recorder

To solve this mystery I first need surveillance equipment that is capable of monitoring my multicore system. As a model, I considered the airline industry's success at understanding crashes by analyzing the data contained in the black-box flight data recorders that are now standard on every flight. Designing and building a flight data recorder that tracks *which* process interacted with *what* piece of state as *whom* and *when* proved to be a daunting task. There were three core challenges to overcome:

- Overhead—The first challenge of building such a device for a computer is to accurately monitor the 28 million daily interactions that software running on my computer has with the roughly 200,000 files and 100,000 settings that it contains. Furthermore, to avoid contaminating the results, we must ensure that this equipment was undetectable by the running software and did not impact the resources and availability of the system.
- Volume of Information—The second challenge is to contain the fire hose of information collected without dropping any. Ideally we want to be able to audit up to many thou-

sands of systems and be able to correlate the information across time and across systems to develop an understanding of what they are doing. Tracking 28 million daily events at 250 bytes per event requires 7 GB of space, which is simply too much to handle. Naively, we may apply our favorite byte compressor, such as GZIP; however, we would find that this would only reduce the volume by approximately 90%, still leaving us with 700 MB daily to deal with. We will either quickly run out of storage space or cripple our system from the I/O requirements of writing these humongous logs to disk.

- Analysis of the Results—The third challenge is to be able to make sense of all this data. If we monitored 5 machines for a week, at 28 million interactions per day we would have 1 billion events to contend with. Traditionally we would attempt to cram them into our favorite database and apply SQL queries to discover the golden nuggets of truth. However, this approach is woefully slow and does not scale. The first problem involves the overhead in converting the data from their archived and compressed form into something that can be bulk-inserted into a database. Then there is the time taken to actually insert it. We found that, even with the stars aligned, our enterprise-class database server could spike to inserting events at a rate of 10,000 events per second. This means the insertion of our 1 billion events would take more than 27 hours. Even if we took the time to prepare and insert this data, we would still need several hours to index the tables, and countless hours to run our queries. Clearly, databases are a significant bottleneck that will force us to limit the number of machine days we can analyze or to filter or condense the data before analysis.

Faced with these challenges, many folks have run screaming from the room, thinking there is more likelihood of success to be had working on their perpetual motion machines. However, I remain unfazed—possibly because I am slow on the uptake, but just maybe because I have a key insight. Traditionally, the three challenges were attacked individually, but perhaps we can drastically improve our results if we optimize across all of them.

Based on this insight, Flight Data Recorder (FDR) collects events with virtually no system impact, achieves 350:1 compression (*0.7 bytes per event*), and analyzes a machine day of events in 3 seconds (*10 million events per second*) without a database. How is this possible, you ask? It turns out that computers tend to do highly repetitive tasks, which means that our event logs (along with nearly all other logs from Web servers, mail servers, and application traces) consist of highly repetitive activities. This is a comforting fact, because if they were truly doing 28 million distinct things every day it would be impossible for us to manage them. If we normalize the events as we receive them, rather than storing them in our log as a flat sequential list, we find that normalization removes the redundancy and provides us with a 35:1 reduction in log size. By maintaining the event logs in the normalized form, we make it faster and easier to analyze the log files directly, which saves us the overhead of putting them into a database.

If we GZIP our normalized files, we can squeeze an additional 10:1 compression, providing us with 350:1 overall. However, having GZIP files gives us the unsavory task of decompressing them before we can analyze the normalized tables for our query. Ideally, we want to decompress only the sections of our normalized tables on which we need to perform our analysis. Pondering this problem, we were motivated by traditional operating system page table design. Our solution was to overlay our normalized

tables atop 64k pages that are individually GZIP'd. This enables us to retain our 350:1 compression property, yet provides us with the ability to traverse the tables and decompress only the sections we need for analysis.

With our new FDR gadget we can easily monitor all file, registry, process, and module load activity in about 20 MB per day. Best of all, a single collection server can easily process in real-time the logs from 5000 systems and archive those logs on its available local disk for six months. With our surveillance tool in place, we are ready to begin our investigation of what computers do all day.

What Computers Do All Day—An Investigative Report

In my quest to understand the secret life of my computer I found that many people are often unwillingly forced into solving very similar problems in the course of their daily lives. At one large Internet company, one-third of outages were found to be caused by human error, and three-quarters of the time taken to resolve the issue was spent by administrators scouring the systems to identify what changes needed to be made. Similarly, a large software support organization found that their engineers were able to identify the root cause of only a scant 3% of the calls they received.

Before investigating my own computer's sordid life, I wanted to understand the state of what ought to be well-managed and well-maintained systems. To understand this I monitored hundreds of MSN production servers across multiple different properties. My goal was to learn how and when changes were being made to these systems, and to learn what software was running. Surely machines in this highly controlled environment would closely reflect the intentions of their masters? However, as you'll see in the following, we found some of them sneaking off to the back of the server room for a virtual cigarette.

THE LOCKDOWN PACT

Although rare, there are periods when system administrators like to kick back and enjoy an uninterrupted dinner with their families. The last thing they need is a problem with one of the pesky attention-seeking servers. To avoid problems, administrators form a secret pact they call *lockdown*, during which they all agree not to make changes to the servers for a specific period of time. The theory is that if no changes are made, no problems will happen and they can all try to enjoy their time outside the hum of the temperature-controlled data center. Using FDR, I monitored these servers for over a year to check the resolve of administrators by verifying that no changes were actually made during lockdown periods. What I found was quite surprising: Each of the five properties had at least one lockdown violation during one of the eight lockdown periods. Two properties had violations in every lockdown period. We're not talking about someone logging in to check the server logs; these are modifications to core Line-Of-Business (LOB) and OS applications. In fact, looking across all the hundreds of machines we monitored, we found that most machines have at least one daily change that impacts LOB or OS applications.

ALL RIGHT, WHO BROKE IT?

One of my favorite examples of a troubled computer is summarized in an

email from one system administrator to all other system administrators in that organization. It reads, "Whoever is changing the page-file setting on these computers please stop—you are taking down our site!" What I like about it is the way it shows how even if we know what the root cause of our problem is, we are powerless to understand how it happened and therefore powerless stop it from recurring in the future. What makes finding the culprit of these changes so difficult is the latency between when the page-file setting is changed and when the symptom of the change shows up (a crash from exhausting physical memory). It turns out that the setting does not take effect until the system is rebooted, which can be a long time after the change was made. Once FDR was installed on these systems, we found that this page-file setting is actually modified quite frequently—tens of servers are affected every two to three months. The FDR logs show that this modification is made by a remote Registry call, likely from a rogue administrative script. Armed with this intelligence report, administrators can now quickly undo the change if it happens again, and most important, they have the critical information required to keep this problem from recurring.

ILLUMINATING UNWANTED APPLICATIONS

We would all expect server environments to be highly controlled: The only thing running should be prescribed software that has been rigorously tested and installed through a regulated process. Using the FDR logs collected from the hundreds of monitored production servers, I learned which processes were *actually* running. Without FDR it is difficult to determine what is actually running on a system, which is quite different from what is installed. It turns out that only 10% of the files and settings installed on a system are actually used; consequently, very little of what is installed or sitting on the hard drives is needed. Reviewing a summary of the running processes, we found several interesting facts. Fully 29% of servers were running unauthorized processes. These ranged from client applications such as media players and email clients to more serious applications such as auto-updating Java clients. Without FDR, who can tell from where the auto-updating clients are downloading (or uploading?) files and what applications they run? Most troubling were the eight processes that could not be identified by security experts. Based on their names, some of these could be benign in-house tools (mlconv.exe, monnow.exe, sitreremover .exe); however, others (e.g., lsacacheagent.exe) sound like potential tools for compromising security (since LSA typically refers to the Windows security system).

REMEMBERING TO LOCK THE DOOR

Few of us obsess overly on securing our home and possessions; we tend to content ourselves with a few commonsense tasks routinely followed to ensure our protection. We lock the doors on our car when we park it, and we lock the doors on our house when nobody's home. Although in the back of our mind we know that if someone is determined to get in they probably can, we don't want to make it easier for them. When it comes to servers, there are some similar best practices. One of them is to avoid leaving credentials or primary security tokens on systems. Primary tokens are created with credentials and can be used to hop to another system. The remote system receives secondary credentials, which cannot be used to hop

again. These are used by hackers who compromise a server to hop from one system to the next and spread throughout your network. Using the FDR logs, we found six services (daemons) running on many machines that were using hard-coded credentials, which could potentially be harvested by hackers. We also found that a third of the systems had screensavers running on them from when administrators had logged in remotely and left their sessions active. These remote sessions leave primary tokens on the system for hackers to harvest. By running FDR on these systems we can quickly identify these potential security problems and ensure we are not making it easier for undesirables to break in.

WHY IS MY SYSTEM SLOW?

When applications are running on our system we really have no clue as to whether the amount of resources they are consuming is reasonable or not. Should this monitoring agent be consuming 5% CPU, or 15%? We really don't know. From running FDR we not only see what processes are reading and writing on the system, but we also have the timestamps for each interaction. Using these timestamps, we can easily calculate how many operations (reads/writes) each process is doing per second. We can even tell if it is reading the same thing over and over and over again. In fact, by looking for these patterns we identified processes that were doing just that. An LOB application was reading the crypto settings 240 times per second, and a management agent was continuously reading the 10,000 service (daemon) settings in a tight loop. Without this information, a developer would usually be oblivious to these useless performance costs. Although these may seem insignificant on the surface, consider that losing 10% of your CPU across 100 servers is equivalent to buying 10 extra servers. Furthermore, if one application is unnecessarily dominating the system by reading settings, other applications will perform more slowly.

When the Cat's Away, the Mice Will Play . . .

Peering through our FDR microscope at the daily lives of our computers, we found many unexpected activities. They made us laugh, they made us cry, but in the end they provided us with knowledge that empowered us to improve our systems. For the past 20 years, systems management has been more of a "dark art" than a science or engineering discipline because we had to assume that *we did not know* what was really happening on our computer systems. Now, with FDR's always-on tracing, scalable data collection, and analysis, we believe that systems management in the next 20 years can assume that we *do know and can analyze* what is happening on every machine. We believe that this is a key step to removing the "dark arts" from systems management.

You too can leverage FDR technology for investigating your computer, by using some of the products that incorporate FDR technology. The first wave of products includes Windows Vista, which contains the drivers that expose the file, Registry, process, and module information through the Event Tracing for Windows (ETW) providers. There is also the Application Compatibility Toolkit v5.0, which contains the Update Impact Analyzer (UIA) and monitoring agents that use FDR technology for understanding Windows systems to enable you to better prepare for upgrades and patches.

REFERENCES

[1] C. Verbowski, E. Kıcıman, A. Kumar, B. Daniels, Y.-M. Wang, R. Roussev, S. Lu, J. Lee, "Analyzing Persistent State Interactions to Improve State Management," SIGMETRICS, Saint-Malo, France, 2006.

[2] C. Verbowski, E. Kıcıman, B. Daniels, S. Lu, J. Lee, Y.-M. Wang, and R. Roussev, "Flight Data Recorder: Monitoring Persistent-State Interactions to Improve Systems Management," OSDI, Seattle, WA, 2006.

[3] C. Verbowski, J. Lee, and Y.-M. Wang, "LiveOps: Systems Management as a Service," LISA '06, Washington, D.C., 2006.

BRENDAN QUINN

lessons learned from living with LDAP

Brendan Quinn has more than 14 years of experience as a sysadmin, security engineer, and infrastrucure engineer. He is currently a Senior Infrastructure Project Engineer at London Business School. He is also a musician and audio engineer.

brendan.quinn@gmail.com

LDAP CAN BE A DOUBLE-EDGED SWORD. On the one hand, you have data available on the network, which is easy to access in a standard way. On the other, you have a system that doesn't work or behave like a normal database. At London Business School, we've had a central LDAP repository for years now, and over time it's come to be at the center of our network. We have applications and network devices that use LDAP for authentication and authorization. We have applications that depend on data in the LDAP repository for core pieces of their functionality. We're even using LDAP to route every piece of email that enters our network. Over the past few years, I've learned a few things the hard way about LDAP integration and performance. I'd like to share a few of those lessons, in the hope of saving you some headaches.

"Why Is the Web Site Slow?" or, Thinking About Performance Tuning

Performance can be a serious problem when dealing with LDAP. There are lots of documents out there that explain the mechanics of tuning particular LDAP servers, and I've included a few links to get you started [1, 2]. In the Sun Directory Server, performance tuning consists of building indexes and adjusting cache sizes. Tuning OpenLDAP while using the Berkeley DB backend will be similar. However, it's difficult to tune an LDAP server unless you understand the traffic it's likely to see. So before you dive in and start tweaking, start by asking a few questions.

Is there a particular application or applications that generate large numbers of queries? If so, can you predict how many?

At our site, we route all the email for our 5000+ person userbase using data stored in LDAP [3]. As you might imagine, this generates quite a lot of queries. We can estimate the number of queries the LDAP server will get based on the number of queries needed to route each individual mail message, multiplied by the number of mail messages handled per day. Looking at the mail logs, we can also get the number of messages processed per minute during peak times or mail floods. In this

case, we tested the LDAP architecture by modeling significantly more traffic than this amount of mail would generate. When we were happy with the response times of the LDAP server under the modeled load, we knew that the gating factor for mail performance would be the underlying mail architecture and not LDAP.

What filters are being used? Does the application use programmatically generated complex filters?

The kinds of filters used by directory-enabled applications can often be more complex and slower than you might expect based on the kinds of data being returned. For example, we have one application that always appends a * to the filters it generates, even though it always knows the exact data it's searching for. So although I'd expected an exact match filter, in fact it always used a substring filter. In practice, this meant that the exact match indexes I'd built were useless; I needed to build substring indexes instead. If you're not sure which filters are being used, check the LDAP server access logs. Logs don't lie, and you can usually see exactly which filters the applications are using.

Once you understand the kinds of filters your applications are using and you have some estimates of what kind of traffic your LDAP architecture is expected to handle, you can start tuning your LDAP servers.

I've written a couple of tools in Perl that you may find useful for testing LDAP server performance [4, 5]. They depend on Net::LDAP.

"Why can't I change my address?" or, A Brief Diversion into Objects, Namespace, and Access Control

In the next few sections I'll be talking a lot about how applications use LDAP. Before I do, let's discuss a few key concepts. For a much more complete explanation, it's worth reading Tim Howes, Mark Smith, and Gordon Good's excellent and comprehensive book [6].

LDAP is object-oriented. That's a bit of a confusing term in the context of a data store, because traditional object orientation is about tightly binding data and logic. LDAP objects don't have logic; they consist purely of data. What object orientation actually means in this context is that data in LDAP is structured as objects, which contain attributes and belong to object classes. Attributes name the bits of data and define the syntax of that data. Object classes define what attributes an object can (and must) have. Object classes use inheritance, which just means that they inherit attributes from their parent object classes.

The namespace of a repository is the structure of that repository. LDAP uses a hierarchical data model. Translated, that means that data is stored and accessed in LDAP as objects hanging in a tree. Tree nodes are themselves objects. Every object has an address, or Distinguished Name (DN), which consists of the list of tree nodes that must be followed to get to the object (see Figure 1).

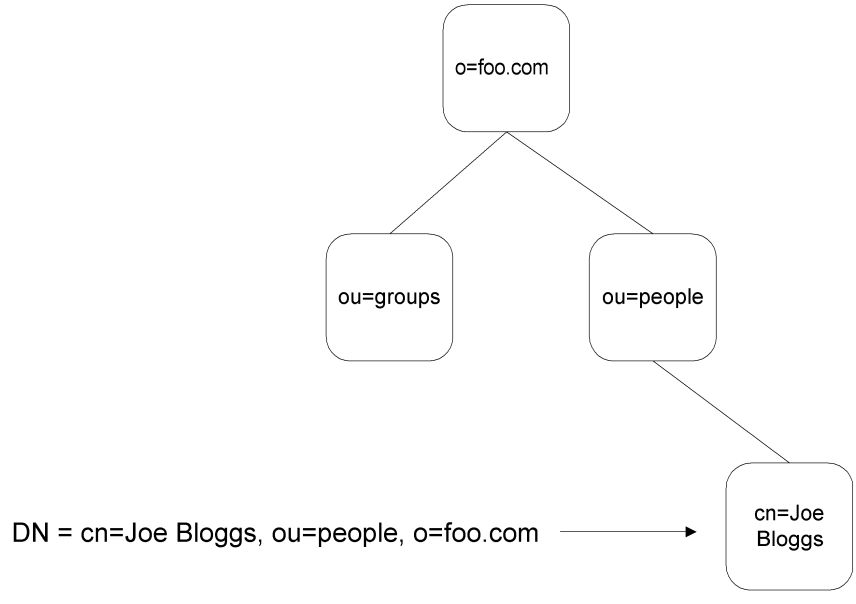


FIGURE 1

The most common way of implementing access control on an LDAP repository is through the use of Access Control Instructions (ACIs). ACIs generally use a syntax similar to that used for LDAP searches (generally called filters). The syntax does differ from vendor to vendor though, so you'll have to look at the documentation for your particular LDAP server for the specifics. The nice thing about ACIs is that they allow you to manage your access control rules in the LDAP repository itself, by setting the "aci" attribute on any object in the tree, nodes and leaves alike.

ACIs also allow the LDAP server to make use of all of the data contained in the LDAP repository when making access control decisions. If the client has provided credentials for an object contained in the repository (called binding), the server can perform internal lookups for group membership, attribute values, and so on.

"But it says on the Web site that it's LDAP integrated . . .", or, Using LDAP for Authentication and Authorization

So you've bought an application or a piece of network equipment, and you want it to use LDAP for authentication and authorization. The vendor claims, "Easy integration with your LDAP repository!" Unfortunately, every manufacturer seems to have a slightly different meaning for this statement.

In general, the Authentication-Authorization (AA) protocol should look something like this:

1. The application performs an anonymous bind against the LDAP server.
2. The application does a search (generally returning only the Distinguished Name) to verify that the username provided matches an object existing in the repository.
3. The application attempts to bind as the DN returned by the previous search, using the password provided.
4. The application, now bound to LDAP as the user attempting authentication, searches for particular attributes in the user's object.
5. The application checks the value of the returned attribute to determine whether the authenticated user is authorized to access the application.

The semantics of the AA process will differ slightly from vendor to vendor, but the overall approach should be fairly similar to that listed here. Unfortunately, there are a few things that can go wrong even with this simple case.

First, anonymous searches (even for DNs) are sometimes forbidden for security reasons, especially in cases where the LDAP repository must be exposed to the open Internet. This will cause the application always to fail to authenticate, because it will be unable to determine which DN to attempt to bind with when validating the user's password. In some cases, the application can be configured with credentials to use for the DN search, rather than using an anonymous bind. Other applications can be configured to construct the DN from the username provided, although this requires that the namespace use the username as part of the DN, which won't always be the case. If the application doesn't support either of these approaches, you'll have to allow anonymous searches for DNs, or not use the application at all. If your site doesn't allow anonymous searches for DNs, it's important to ask the vendor whether the application supports one of these alternatives before purchase.

Second, at many sites, users are forbidden from reading some or all of their own attributes. This will cause authentication to succeed but authorization to fail. The simplest way of dealing with this is to permit the users to read any attributes in their own object that are used for authorization, but this will not always be permitted by the security policies. Some applications will support binding as another user (typically the one used for the DN search) to search the user object for the required attributes.

Finally, which attributes does the application use for authorization? Some applications will allow you to specify an attribute name to check and values to look for. Some will allow you to specify which attribute to check but will require a fixed, vendor-specified syntax to match against. Still others will require that you add a specific attribute or attributes to your schema, generally with vendor-specified syntax. Whatever the case, you'll need to think about where the data stored in these attributes comes from, and how it's maintained and managed.

It's worth noting that some applications that claim to support LDAP authentication don't use the generic process outlined here. In some cases they're designed to bind as an administrative user (with full access rights), retrieve the uid and userpassword attributes, and attempt to validate the passwords themselves. This is poor practice, and you should avoid any vendor who uses such a brain-dead approach.

"I know, we'll just use LDAP for everything!" or, Directory-Enabled Applications and You

The most common use of LDAP is for authentication and authorization, but there are also applications, such as our email system, that make operational decisions based on the data contained in the repository. This class of directory-enabled applications generally makes use of application-specific schema (although possibly making extensive use of one of the standard schema as well). An application of this type will often expect to manage the attributes in the custom schema itself, writing directly to the repository. As many sites don't permit users to modify most of their attributes directly, this generally requires giving the application some kind of administrative access. At our site, we've handled this by setting up a number of

group objects (in ou=Groups), which we reference in the ACIs on different parts of the repository. When an application requires administrative access, we create a user object specific to that application, and we add that user's DN to the group object's uniqueMember attribute. This minimizes the need for rewriting ACIs and brings the number of ACIs needed down to a manageable level.

When you're dealing with applications that perform LDAP writes as well as reads, it's important to make sure that the application developers understand that an LDAP repository is not a traditional database. LDAP servers are highly optimized for reads over writes, and write operations can take a fair bit of time. In extreme cases, too many write operations can slow an LDAP server down to a crawl. For example, at one point we had an application developer who was using LDAP to store session cache information. This was generating an LDAP write for every click that every user made on his or her application's Web site. The solution in that case was to gently explain to the developer why this was such a bad idea. You're less likely to run into this issue with commercial applications, but when applications are being developed in-house it's worth repeating the "write rarely, read often" principle at every opportunity.

One more thing on writes is worth mentioning, before we move on. If you're using a Master/Slave LDAP architecture, when a client attempts a write to the Slave, it will get a referral to the Master. In my experience, most applications only allow you to configure a single LDAP server pool, which is used for both reads and writes. If the application needs to write to LDAP and supports LDAP referrals, you should configure it to use Slaves. If the application doesn't support LDAP referrals, you will have to configure it to use the Master. Most applications that claim to support LDAP referrals work as advertised, but, not surprisingly, there are a few that fail in unexpected ways. If your site uses LDAP referrals and you're having problems with a particular application, try configuring it to use the Master instead. If this fixes the problem, you'll know that the failure most likely is in the way the application handles LDAP referrals.

"But that data was in the repository yesterday . . . ," or, Issues Arising from Data Management

The data in your LDAP repository has to come from somewhere. Some data will originate in LDAP and be maintained directly there. Some data will likely originate in one or more administrative databases. Some may originate in another, special-purpose LDAP server, such as Microsoft Active Directory. All of this data will need to be synchronized. When there are applications making decisions based on LDAP data, synchronization issues can have unexpected consequences.

Let's examine a hypothetical site, where most of the data in ou=people originates in a single administrative database, while passwords are managed directly in the LDAP repository. Data from this database is synchronized with LDAP by using a process that runs once per day. One day, somebody working with the database makes a mistake and accidentally deletes the entire marketing department. This is noticed relatively quickly (within an hour or so), and the database is corrected. Unfortunately, between the deletion and the correction, the daily LDAP update has run. This resulted in all of the users in the marketing department having their user objects deleted. The LDAP administrator doesn't find out about the issue until the call comes in from the head of marketing, who is angry that

no one in the department can get onto the network. Our poor LDAP admin is faced with a quandary. If the database synchronization process is rerun, all the users will be recreated, but they will all need their passwords reset, and some of their usernames may have changed. If the repository is restored by using the last backup, those in the rest of the company who changed their password since that backup will have their password set back to before the change.

What could have been done to prevent this from happening in the first place? You can't completely prevent human error, so there's nothing that could have prevented the initial deletion from the database. However, this type of failure could have been anticipated and planned for. For example, the synchronization process could have been designed to be less automatic, requiring that someone look at an analysis of the changes that will be made and start the process manually. Alternatively, the synchronization process itself might have had a sanity check built in: Build in some logic that would notice that the process was being asked to delete a large number of user objects, and refuse to proceed without human intervention and approval. Did the objects need to be deleted immediately at all? What if the synchronization process had instead marked the object as inactive, say, by prepending a date string to the hash contained in the userpassword attribute with the date that the account had been made inactive? The process could then automatically delete inactive user objects after a specified time period. If you needed to recover the password you could simply strip out the date string.

Making your synchronization processes smarter will help, but not all data problems will be as obvious as this. Bad data is everywhere and isn't going to go away anytime soon. The best that you can do is to try to understand the most likely sources of bad data, and try to minimize the impact of any predictable failure conditions.

One way that LDAP differs from traditional databases is that the data isn't designed for one particular purpose. Once data is in LDAP, people will find new ways of making use of it that can't be anticipated. This is the very thing that makes LDAP so powerful. Unfortunately, this complicates the job of data management. Each application and data source will have a different conception about the purpose of the data in the repository. Understanding what applications are accessing the repository and how they're using the data is an essential step in understanding how to manage the data that resides there.

"conn=37288428 op=81303 RESULT", or, Conclusion

You can tell that the LDAP infrastructure is working well when it's become just like the plumbing. Everyone uses it, but nobody ever thinks about it. Unfortunately, when the LDAP infrastructure isn't working well, it's more like public transportation. Everyone still uses it, but they complain about it all the time. With some thought and attention, an LDAP repository can be one of the most reliable parts of your network infrastructure. I hope this article has given you a few things to think about along the way.

REFERENCES

[1] Steve Lopez, "Solaris Operating Environment LDAP Capacity Planning and Performance Tuning," is a very comprehensive document, which, although primarily focused on Solaris and the Sun LDAP server, provides a

wealth of information useful in the general case. Available at http://www.sun.com/blueprints/0502/816-4829-10.pdf.

[2] "OpenLDAP FAQ-O-Matic: Performance Tuning" provides an overview of performance tuning OpenLDAP. Available at http://www.openldap.org/faq/data/cache/190.html.

[3] Brendan Quinn, "Integrating Exim with LDAP for Mail Relaying—A Case Study." Available at http://www.uit.co.uk/exim-conference/full-papers/brendan-quinn.pdf.

[4] lookup.pl: This is basically a simple LDAP search tool, but with all the search options pushed into a configuration file, and with high-resolution timing metrics added. The advantage of this is that it lets you save models of different types of searches. Available at http://phd.london.edu/bquinn/lookup/.

[5] ldap-load.pl: This is an LDAP load testing tool based on lookup.pl. It simulates load by spawning processes, which then send queries to the LDAP server. It's mostly useful for generating peak loads, but it can be used to generate sustained load as well. Available at http://phd.london.edu/bquinn/ldap-load/.

[6] Timothy A. Howes, Mark C. Smith, and Gordon S. Good, *Understanding and Deploying LDAP Directory Services* (Macmillan Technical Publishing, 1998).

JOHN LLOYD

introducing system engineering to the system admin

John Lloyd is a computer systems engineer at MDA, a Canadian technology company that builds and operates systems ranging from land-title information to space robotics. He has been doing system administration and system engineering on machines ranging from PDP-11s to SGI Altix for nearly 30 years.

John.Lloyd@MDAcorporation.com

THE METHODS OF SYSTEM ENGINEERing (SE) can help solve some typical system construction and operation problems. Using a case study as a demonstration, in this article you will learn to use SE methods to build a simple database system.

A System Administrator's Story

As a system administrator, you are asked to put together a computer system to provide an Oracle database for an Internet Web site. The project leader explains to you that costs have to be low but the system has to be highly available, with better than 99.9% uptime. The Oracle software required by the Web page developers who are providing active (programmed content) Web pages to the system's end users largely dictates the capacity of the system you need to acquire.

You find some used Sun gear, consisting of an A1000 SCSI RAID system and an E450 four-way SPARC computer. The Oracle DBA (database administrator) is very pleased, since this system contains twelve 72 GB disks and has four processors. This seems to be plenty. The project manager is pleased, since you saved lots of money by buying used equipment.

Over the next year the computer system goes through several major and minor crises.

For example, one day a disk in the RAID storage array fails; you notice the yellow light and replace the disk with a spare you thoughtfully ordered with the system. No data is lost. The project manager is happy and publicly congratulates you.

A few months later the software developers phone you at home complaining that the database server has stopped functioning during a software upgrade. After some investigation you discover one of the Oracle filesystems is full. After half an hour of careful and patient probing, you discover the developers have to modify several very large tables in the database and are doing this one table at a time. It turns out this is what consumes disk space—complete copies of the original table are stored in the Oracle "rollback" space. The software upgrade has to be aborted for lack of adequate disk space. Because the half-completed upgrade process has left the database unusable, it takes all night and most of the next day to restore the database, resulting in a full day of downtime.

The next day the project manager gets very angry when he discovers what had happened the night

before. Everyone points fingers at the other groups: The DBA complains about disk space; the developers complain about the Oracle software behavior, and you complain about developers not knowing the consequences of their actions. No points for anybody. But you get budget approval for a disk upgrade.

The third episode revolves around another disk failure. This time one of the system disks fails. You have thoughtfully implemented software RAID 1 (disk mirroring). But upon replacing the failed disk you discover the system will not boot. After several hours of investigation, you discover you've made a basic error in setting up the system disk mirroring. The fix involves a complete system-disk replication and a planned one-hour shutdown has turned into a 12-hour outage. Once more, the project manager is unhappy.

At your annual performance review, the project manager points out this history. He states that as a result, the system performance was not met, since the system was down for two days, resulting in only 99.45% uptime for the year. You respond by acknowledging the system-disk problem, but you refuse to accept responsibility for the Oracle disk-space issue; it was not under your control. You point out the success of the RAID disk replacement. By your accounting, system availability was 99.86% because the system was only down for 12 hours on a Saturday, and everyone knows the customers don't use it on weekends. That's "pretty close" to 99.9%, isn't it?

System Engineering to the Rescue

System engineering is the systematic application of engineering methods to identify the issues and requirements, develop and evaluate alternative architectures and designs, implement and test the selected design, and verify the correctness of a system implementation.

This is a lengthy definition, and you may well ask, "How do all these activities help this situation?" The short answer is that SE enables successful implementation of the features that are important to the customer. For the example given here, this means identifying and meeting the 99.9% uptime requirement given all foreseeable uses of the system.

The hardware failures cited in the case study could not have been avoided. However, the systematic failure to provide disk space could have been prevented by SE processes. System engineering provides the means to design for and test the system response to identifiable situations, including software upgrades.

A Simplified SE Process

SE uses several types of design information, including figures, tables, and text. By using a specified procedure, they are developed and maintained as documents. These will require some effort to generate and maintain. The value of the SE process outweighs the effort required to maintain these documents.

The general outline of the process is as follows. First, identify the system to be built, ensuring that interfaces to outside systems are well defined. Second, define system functionality by writing requirements statements and casting them in a testable form. Third, design the system by methodically placing the list of requirements in the design. After assembly and construction, test the system by showing that it meets the stated requirements.

In terms of SE, identifying requirements is the central goal. Providing clearly defined requirements that are unlikely to change makes designing, implementing, and acceptance-testing understandable and manageable. Additionally, a clear and accurate set of requirements for a system provides stability to the designers, and to the implementers.

System Boundaries

Identifying the system to be built provides the boundaries of the system and the information flows which the system provides or acts on. System requirements can then be stated in terms of these flows.

In our case study, the system admin implicitly identified the system as a computer and Oracle database software. The system admin did not consider the Web server software as part of system admin responsibilities. The software developers understood the system as Java code running on the Oracle database supplied by the system admin. They understood the database in abstract terms, as a data storage system.

Each group had a different view of what the complete Web server system consisted of. Neither fully understood the scope or extent of their views, or the relationship of their part to the organization's real goal of operating a Web site. As a result, the combination of computer and software did not meet management's expectations.

By taking the steps to formally list the actual interactions between these two views of the system, we can define and separate two separate subsystems and the information flows between them. This process provides a way to clearly identify where the database system begins and ends. It also provides the basis for defining the expectations for the database (the requirements).

The interactions between database software and Web application are familiar to most developers and system admins. These interactions, in addition to the usual ones required for any computer system (monitoring, backups, etc.), define the list of interfaces to the database subsystem. By taking this environment into consideration, these interactions define the scope or the extent of the database subsystem. In other words, we identify the database system by defining its interactions with external systems in its environment.

The database system environment is illustrated in Figure 1, as a context diagram. In this figure, each interface with an external entity is identified and the flows of information are shown by the arrows. The intent here is to show all substantial information flows, while excluding uninteresting flows, for example, power and cooling, details of user interfaces, and operating system installations. To supplement this figure, we normally add a table describing the key types of information exchanged on each interface. An example is given in Table 1 (facing page).

The list of interfaces and their data flows will be used as the basis for defining functions and behaviors of the system, also called "requirements." But first, let's take a more detailed look at the figure and table.

Experienced system designers might have some comments about these interfaces as I have described them. The database subsystem is responsible for backups, and opinions may vary on whether the data-model updates are a software upgrade or an information flow. From the system admin point of view, they are data items exchanged with the Web developers and are considered as data.

Interface	Item	Description
Application queries	Oracle queries and inserts	SQL statements operating on the set of application tables
Developer updates	Software updates	Replacement .jar files and .war files comprising the Java application are supplied by the software developers
	Data model updates	Scripts to convert the schema to the next version, and to undo this conversion
Offsite backup tapes	Media	Online Oracle backup tapes, in tar format, are sent offsite by courier
Sysadmin monitoring	Logs	System logs
DBA monitoring	Logs	Database alerts and trace logs

TABLE 1: EXTERNAL INTERFACES

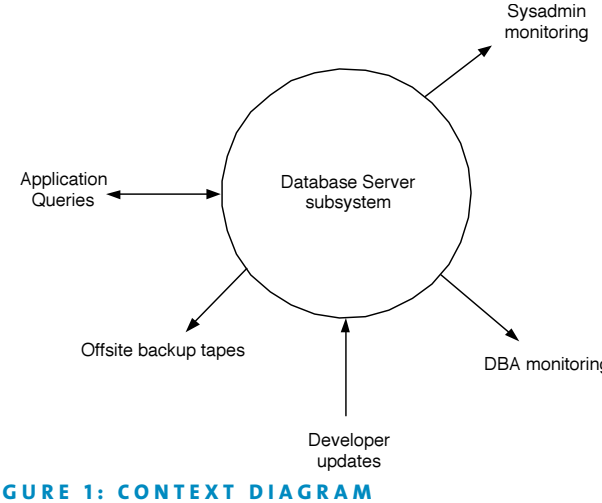


FIGURE 1: CONTEXT DIAGRAM

Here are some simple rules for composing a context diagram:

- Significant information flows, together with directions, are shown.
- Information flows with separate external entities are shown.
- Control flows are excluded.
- Heat, power, light, and space are excluded.
- Software installation, configuration, and management are excluded.

The last rule seems to be broken in my example. In this case, the database subsystem operates on data, and here the data happens to be considered as software by another subsystem.

In addition to the diagram, provide a table (see above) identifying and describing each information flow. The interfaces and flows define the operational context and the extent and scope of the system. Now any functions or information can easily be identified as part of this system, or not.

With the context and information flows between the system and significant externals identified, we can examine functional behavior of the system.

Requirements

With the boundaries of the system identified, we can now address system requirements. These are a set of statements that clearly state all of the

"real" behaviors, functions, and system characteristics; they identify the pertinent functions and capabilities of the system.

Good requirements do not impose or specify a design. Good requirements describe the system behavior; they do not describe the internals of the system. The goal of requirements analysis is to provide objective descriptions of the system. Each requirement, therefore, must also be testable; we need to be able to verify the defined behaviors or attributes. (It is worth noting that writing testable requirements saves time and trouble later.)

Requirements are the most important and critical element of SE. As the basis for the system design phase, they must be complete and understandable. They form the basis for testing, where the tester must show that each requirement is met. Requirements also describe characteristics, such as system reliability or usability, that have to be built into the system design and proven by analysis, examination, or testing of the assembled system. Requirements are also the principal means of communication among developers, sysadmins, and the system users and owners. They are (or should be) mutually understood and agreed upon.

What are "good" requirements statements? Here are some guidelines:

- They can be demonstrated, measured, or verified by analysis.
- They do not contain adjectives that are subjective.
- They do not contain "and" or "or" to separate multiple requirements.
- They do not contain design descriptions or prescriptions.
- They may contain constraints such as preferred vendors or technologies.

Here are a few good examples:

- Shall run Oracle version 10GR2 (this constraint is demonstrable).
- Shall support 120 GB of disk storage (demonstrable).
- Shall perform daily full backups of Oracle data (demonstrable).
- Shall perform daily full backups of operating system (demonstrable).
- Shall support 6 Web logins per second (measurable).
- Shall be available 99.9% of a calendar year (verifiable by analyses).

These examples are clear and concise. Although a software product is specified, this is more of a constraint on the designer than a design requirement, but it is realistic. The sizing statement can be demonstrated by examining results of a command (no arithmetic is needed). The backup functions are separated into two statements, to avoid "and." The availability requirement uses calculations based on vendor-supplied failure rates, and is therefore verifiable by analysis (99.9% of a year is about 8.75 hours downtime per year).

Some examples of bad requirements statements are:

- Shall provide appropriate backup system (adjective is subjective).
- Shall prevent insecure logins ("insecure logins" undefined).
- Ought to be available 7x24x365 (not a yes/no requirement).
- Cannot allow hackers in (unachievable).
- Has to use RAID1 on all database disks (design imposition).
- Shall use Java for writing the Web app (not applicable to this subsystem).
- Should use disk-to-disk backups for Oracle data (design imposition).

These examples show some possible errors of composition or design specification. Some use undefined terms, define impossible goals, or include statements that are not applicable to this particular system.

The last example is a classic requirements error. The "obvious" function to require disk-to-disk backups is stated as a requirement. Fixing this statement requires some detective work to uncover the true purpose: Is the intended meaning to limit recovery time from backups, or is some other objective sought? The requirement should be rewritten to specify the actual need (e.g., "A database restore shall take less than 4 hours").

The benefit of a set of good requirements is simplicity. Good requirements make it easy to understand the functions and characteristics of a system. They also make these statements readily verifiable.

One recurring problem that arises when writing requirements is the risk of missing some "important" requirement. This is a constant issue for SE, and there are a number of ways to reduce this risk. One way is to establish "use cases," as is common in software engineering. These are paper exercises where the functions, inputs, and outputs of the system are considered for different scenarios. These descriptions are checked against the list of requirements. Another method is evaluating the lifecycle of each key data item or information item. We can identify the birth, life, and death (or permanent archival) of each data item and verify that requirements cover every step.

How many requirements statements are enough? If there are too many, the system may not be achievable because of conflicting requirements; too few, and the users have not communicated enough about what is needed. A short answer is: Enough to be clear on the goals of the system.

System Design

The system designer identifies useful components such as computers, networks, and software and connects them in a way that meets the system objectives. In this example, we have already committed the act of design by separating the Web site into a database subsystem and a software subsystem, and connecting them by their data flows. It's not always that simple.

The context diagram describes the database subsystem in its environment, whereas the requirements list identifies the functions and information processed by the database subsystem. The database design provides components to support each information flow in the context diagram and each function and each characteristic identified in the requirements.

Based on experience, research on vendor guidelines, recommendations of peers, or pure invention, the designer identifies the components required to complete the design and composes a figure showing their relationships, as in Figure 2 (next page). In this figure, boxes are the components, circled numbers identify internal information flows, and labeled arrows show flows identified in the context diagram.

This is what is commonly called the design of the system. It shows the computer parts, backup system, two kinds of storage, and related connections. It does not show unnecessary detail such as electrical power, system installation or configuration procedures, cost, etc.

Something very important should accompany this design figure. Each component of the design needs to be matched to one or more requirement statements (see Table 2, next page). This is a very significant (and some-

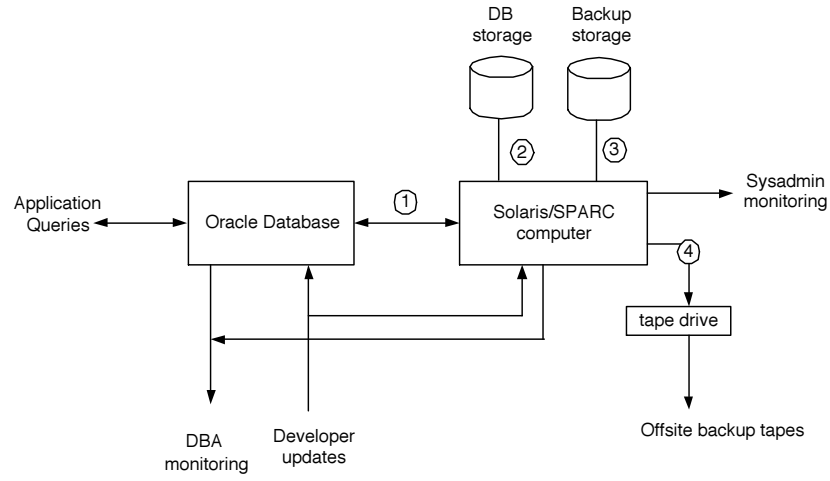


FIGURE 2: SYSTEM DESIGN

Category	Requirement	Allocated Component
Oracle	Shall run Oracle 10GR2	Oracle database
	Shall perform daily full backups of Oracle	Oracle database, Solaris/SPARC computer, DB storage, backup storage
Availability	Shall be available 99.9%	Solaris/SPARC computer, DB storage

TABLE 2: REQUIREMENTS ALLOCATION TO DESIGN

times troublesome) step. Matching components to requirements guarantees two things:

- Each requirement is covered by at least one component of the design.
- All design components are necessary.

By matching a requirement to one or more components, the designer ensures that the requirement can be satisfied by the functions of those components.

If the design contains components not related to official requirements, then there is something funny going on. Possibly, requirements are missing, the designer has made a mistake, or something in the process has broken. This can be a very useful test of the process and a clear quality-assurance check on both the design and the requirements. Getting to this point (clear requirements and consistent and clear design) is a major milestone in the process and has clearly significant benefits for the project.

For the design to be valid and useful, it should follow some rules. First, the design must match the context diagram. Each data flow in the context diagram must match the corresponding flow in the design. The design should maintain a fairly consistent level of abstraction, providing information flows between components rather than reflecting physical connectivity. (This can be a subjective measure, and hard to achieve in some cases.)

As shown in our design, we have identified internal information flows, and so we should identify what goes on in each flow. Some are very simple. In the figure, number 4 is a SCSI or Fibre Channel interconnect, whereas number 1 will reflect all the information flows between a complex database product and the underlying operating system. You should be able to identify each element and describe its functions. The flows of information between each element should be described as well. This combination of (1) a figure showing relationships between components and (2) information flows defines the design.

Interface	Item	Description
1. Oracle-to-Solaris	Data storage for tablespaces	Filesystems for storing tables
	Control-file locations	Distinct filesystems on separate devices to hold copies of Oracle Control-files
	Operating system interface	Process creation, memory mapping, I/O and other standard OS features
2. DB store	Disk I/O	SCSI-compatible disk storage
3. Backup store	Database backups	SCSI-compatible disk storage
4. Tape interface	System backups	SCSI-compatible tape drive

TABLE 3: INTERNAL INTERFACES

Even this figure and accompanying tables, of course, are not enough to fully characterize the design. Performance and scalability are not described in this figure; an analysis of system availability is needed, and so on. In fact, a significant effort is normally made to analyze the characteristics of a design. If an analysis shows noncompliance to requirements, the designer will necessarily have to modify the design or even try to renegotiate requirements.

Readers familiar with software systems design will recognize a pattern here. Requirement statements are formally linked to each element of the system. The links are traced out in both directions, and these are thoroughly checked.

The design must be proven by assembling the components and by checking each requirement. This is the actual work of constructing the system.

Integration and Testing

Once the design is completed (and this means that the requirements set is well defined and that results of various design analyses are satisfactory), the system is assembled and tested.

Most system admins have experience assembling computer systems such as this one. For this small example, it is fairly straightforward to assemble the components, install the software, check vendor Web sites for the latest bug fixes and patches, and complete the patching and configuration of the system. Many might object that formal testing of the system is not really required; they know what they are doing.

The point is that testing applies to the complete system, including interaction with the Web application, not just the database server. Merely testing the database system does not uncover potential systemic errors. Covering all identified requirements makes the complete system much more trustworthy, because it has been tested. Testing should be applied in a controlled, reproducible process to whatever degree of formality is needed by the paying customer, even if it is your own organization.

Preparing tests and performing testing provides three benefits:

- It confirms that requirements are met.
- It provides reproducible tests for future system-integrity checking.
- It confirms valid requirements.

The first two points should be clear. The last point requires some explanation.

The SE process as I've described it includes many references to the requirements set. This dependency is obvious and essential—meeting requirements is the goal. However, a good requirements set has to be valid, in the sense that the requirements must accurately describe the intent of the customer. We all know that customer intentions can change, owing to market conditions, technical advances, or simply improved understanding of the problem being addressed. We can use system testing, or even just the process of writing system tests, as a way to help discover true requirements based on customer intentions. The tests need to be approved or reviewed by the customer: This is their chance to confirm their validity.

Finding the Systemic Problems

Here lies the solution to the two system issues that lay undetected in the Web services system described in the case study. First, the system-disk recovery process was simply not tested; therefore no one discovered it was broken. A formal test with simulated disk failure would have uncovered the problem and resulted in a revised design.

Second, the process of upgrading the Web application software and implementing a data model change (see Table 1) would also have been tested. This test might not have uncovered a storage problem unless the disk usage was measured during the change. It's unlikely anyone would include such a measurement in a test procedure, that is, not unless there is another, real requirement statement:

> Requirement: The system shall support 1 year of online data.

Is this a real requirement? Yes, it is measurable and does not depend on the application or the particulars of implementation. And it is verifiable by adding a year's worth of simulated data to the database and running tests. A suitable test case (perform a data model change with one year of data) would have uncovered the second systemic fault before it was discovered operationally. The operational problem would have been avoided.

Other potential problems can be discovered, such as meeting the Web login rate (6 per second) when the database contains a year of online data and database backups are in progress. Systematic test planning can help uncover this scenario.

Conclusion

We used a case study of a typical database implementation to introduce systematic review and testing of system requirements. Our systematic process is called "system engineering" and meets the goal of providing an understandable, usable process to generate and deploy computer systems that correctly meet the customer's defined goals.

There is much more that can be said: I've omitted some key topics from this description, including the necessity of revising the requirements, designs, and test plans as new insights are gained. I have not described in detail how to document interfaces. The politics of uncovering and documenting requirements from sometimes uncooperative customers has not been described. Also, I have not included a description of document configuration management, which is essential to maintaining control of the SE process.

The thoughtful reader will recognize that the same methods described here for designing a database system can be used for each component of the

database system in turn, resulting in a hierarchy of context diagrams, designs, component context diagrams, component designs, and so on in greater detail. This is a key benefit of SE; once learned and applied, it can be used repeatedly over and over throughout large system implementations.

RESOURCES

The definitive text on SE is *Systems Engineering and Analysis* (4th ed.) by B.S. Blanchard and W.J. Fabrycky (Englewood Cliffs, N.J.: Prentice-Hall, 2005). This is not, however, an easy introduction to the topic. It describes SE for very large systems and includes the application of the SE process to the production of the systems (factories) as well as the final product and long-term support of systems in the field. Earlier editions are useful too.

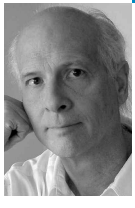
The U.S. Defense Systems Management College's *Systems Engineering Fundamentals* (Fort Belvoir, Virginia: DSMC, 2001) is shorter than Blanchard and Fabrycky's book, but it requires careful reading. It describes SE in 40 pages (of a total of 200), with the rest of the volume describing management of the SE processes. It uses more technical (and somewhat different) jargon and references some U.S. DoD standards. It is freely available on the Web.

INCOSE (International Council on Systems Engineering) publishes a single-volume handbook, available to members. See www.incose.org.

International standards on SE include ISO 15288, EIA/IS 731, and IEEE 1498. These are often costly to obtain, and a textbook (or three) is more likely to be useful. Freely available standards include ECSS (www.ecss.nl), MIL-SPEC 498, and DOD-STD 2167A, although the latter two are more software-oriented.

RIK FARROW

promises, promises: an interview with Mark Burgess



When Rik Farrow is not acting as editor of *;login:*, he is working to improve computer security as a consultant and researcher.

rik@usenix.org

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

Mark.Burgess@iu.hio.no

RIK: SO LET'S BEGIN AT THE BEGINning, shall we? Assume your gentle reader is clueless in these discussions. What led you to promise theory?

Mark: Well [laughs], as always, ideas come by a circuitous route. Promise theory sort of grew in my mind over a few years from thinking about everything I was doing: network graph theory, game theory, host autonomy (as in cfengine), policy-based management, anomaly detection, fault analysis, etc., etc. I think I was getting interested in others' modal logic approaches to policy and was at the same time getting depressed that they seemed to be a complete waste of time. Then, I kind of have this thing where I challenge myself to break with convention and say: The hell with what everyone else is doing; I've got to think again from the beginning! How would *I* do it?

At the time, I was doing some work on network ranking in search algorithms (like Google's PageRank) with two colleagues at Telenor and I was wondering whether there was a way of using those ideas for pinpointing problems in computer networks. It occurred to me that the reason networks succeed or fail to make a working system is all about what the individual nodes do for each other. So that might lead you to thinking about a kind of Service Oriented Architecture, but I'm always trying to look beyond the obvious answer. It occurred to me (from what I know about events and anomalies) that what we *do* or what actually *happens* is far too ephemeral to be interesting. It is not the issue. Rather, it's the *average* behavior of systems that tells you what's of lasting value about it. You know, if we design systems by thinking about mosquito bites we'll wallow in details that will mostly average out to nothing. But what if we could simply describe how individual (i.e., autonomous) components *behave* toward each other on average, and see how that ends up leading to a working system? In other words, stop thinking about the networks as communication, and start thinking about them as *interaction*, or what behaviors the components exhibit toward one another. That was sort of what got me going.

Rik: And you described the nature of these interactions between components as "promises." Why "promises"?

Mark: Yeah, that's important. We're trying to get to a simple summary of how parts in a system will behave when we put them together, and that includes both machines and people—human-com-

puter systems, as I like to say. I think we have to realize that systems don't always do what we want them to, and that we sometimes take it for granted that they should. A promise (if made) is a good way of summarizing how a component will try to make its best effort to contribute to a system voluntarily. Voluntary cooperation, in turn, is an important viewpoint (though you might at first think it's a bit odd) because it says that whatever autonomous decisions have been made by the component, you know either behind the scenes by its owner or programmed within it, these decisions are going to support this behavior that has been promised. So a promise captures the essence of what behavior is advertised and planned for. Now let's say a system does not behave the way we want. This could mean either that it has not promised to behave in that way (i.e., choice was involved) or that it has indeed made a promise but was not able to comply for reasons beyond its control (i.e., a fault).

Without a promise we can't tell the difference between not being willing to cooperate and not being able to cooperate, that is, design error or a fault. So we would miss a vital part of the specification, something like half a contract. If we have all necessary promises to guarantee success, the only reason for failure in a system is an unforeseen fault. So we distinguish between what is promised or "expected" (with inevitable uncertainty) and what is simply "unknown." I think this helps us make an important conceptual step away from believing that we can magically force components to do their jobs.

Rik: I find myself wondering about systems that make promises that they cannot fulfill. I don't see anything in promises that would prevent a system from "lying" about its capabilities and thus tricking other systems into choosing to rely on a system that will fail.

Mark: That's true. It's the way the world really is. Just as there is nothing to prevent any service provider or component in a system from lying about its service delivery. A good example of this is a power supply we bought recently that is rated with a certain current delivery that was simply false. I should be clear: We talk about voluntary cooperation not because it is necessarily desirable but because it is the only realistic viewpoint. Actually, you can't squeeze blood from a stone, or force someone to deliver on something without their voluntary cooperation. At best you might be able to refuse them something in return, if there is a trade of some kind.

Take a peering agreement: If you promise to carry my traffic and then later refuse, I can withdraw my promise to carry yours. In the case of a power supply, there is no trade. The power supply is a "slave" component and if we anthropomorphize, we have no threat of reprisal if it lies about its capabilities—we are simply screwed (pardon my French). Of course, in reality the manufacturer of the power supply lied about its capabilities, so we could either withdraw our promise of money to them or make a threat of litigation (which is a promise to do something that would have a negative value to the other party). The promise paradigm still fits. Promises only help us to manage this uncertainty by indicating an intention to behave in a certain way.

Rik: I can see how promises are a realistic way of representing the relationship between components. How do promises fit into configuration management?

Mark: Configuration management is a service that essentially makes a number of promises about how a system will be configured.

Rik: You and Alva Couch had a paper at LISA '06 that ties promises in with two other concepts: closures and aspects [1]. Can you explain briefly how these three concepts fit together?

Mark: Yes, Alva and I don't get to talk half often enough, but when we do we click on things quickly. We were actually walking around Vancouver at NOMS2006, lapping up some sunshine after I had been really sick in my hotel room for a couple of days. My eye for interesting graffiti was about to take us into "Death Alley" when a guy climbed out of a pile of garbage and tried to sell us drugs (carefully pointing out that we would probably be murdered if we walked down said alley). Anyway, this gentleman then showed us an alternative route through the neighborhood and gave up only when he finally believed that all I needed was aspirin. The promise made by "Death Alley" could be thought of as one of any number of aspects of Vancouver—crime, life, death, violence, drugs, etc. An aspect is a very high-level idea, much higher-level than a promise, but it is nonetheless a thing we use all the time to organize our thinking. Paul Anderson points this out in his SAGE booklet. Think of a Web page, if you like. Its specification makes certain promises about text, palette, images, etc. An aspect of these promises could be "color contrast." If we want to increase or decrease the contrast, we might have to reevaluate promises made about the text, the images, and the colors in a coordinated way even

though they are closed-off and separate categories that we deal with in quite different ways. Aspects often cut through several specific issues.

If we think of aspects of configuration management such as backup, host naming, and service delivery, we can express them (compile them down, if you like) into low-level promises between specific components. The promises are much closer to showing you how to implement these concerns. In fact, cfengine statements are essentially promises (to fix a system in a particular way if anything should go bad). Now, closures are a software idea that represents somehow the autonomous nature of agents in promise theory.

Alva was thinking about closures even before I was thinking of promises. It turns out that these are all very complementary concepts. The agents interact only through the promises they give and receive. Otherwise, they are independent. A closure is essentially an agent that interacts only through its computing interface, that is, in accordance with the promises it makes to others. Like an agent, it does what is inside it; it cannot be forced by an outside influence (for instance, there can be no global variables or shared memory leaking control information in or out). Closures cooperate essentially voluntarily, by virtue of their internal specifications. They can't be obliged to change their behaviors. An agent follows a certain internal discipline, and a promise is a piece of glue that binds closures into cooperative patterns of behavior. These then result in certain "aspects" of configuration being promised.

Rik: Are closures about voluntary cooperation?

Mark: Again, promises go beyond mere change management. They say: Forget about obligations, deontic logic, and all of the barely plausible security models policy people talk about, and think realistically about what happens when you put closed components into a system. You buy a resistor or a capacitor that makes a certain promise, usually printed on its side. You can't force a resistor to be a transistor, so why even try to talk about obligations? The component does what it can, or what it "wants," *voluntarily*. There are no genies in the bottles. When we ask ourselves what promises are required to build a radio, we go beyond one instance or one application to what could be. I am a resistor and I promise to resist by one hundred ohms plus or minus five percent—not by the number you first thought of! That is an obvious but crucial philosophical aspect of management thinking that we seem to have forgotten in computing.

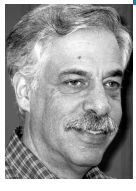
REFERENCE

[1] "Modeling Next Generation Configuration Management Tools": http://www.usenix.org/events/lisa06/tech/burgess.html.

[2] "Introduction to Promise Theory": http://research.iu.hio.no/promises.php.

DANIEL L. APPELMAN

spam and blogs



PART 1: SPAM: A BALANCING ACT

Dan Appelman is legal counsel for the USENIX Association and practices technology law as a partner in the Silicon Valley office of Heller Ehrman LLP.

dan@hewm.com

THE EVER-INCREASING USE OF THE Internet creates new challenges for the system administrator. Many of these challenges have legal dimensions. This is particularly true of spam and blogs. By some estimates, over 75% of all email traffic is spam. In the United States and abroad, laws have been enacted to regulate spam with various remedies and varying success in encouraging compliance. Blogs are increasingly used not just for personal expression but also for commercial purposes. It often falls to the system administrator to design and enforce company policies to protect against spam, to limit personal use of blogs using company facilities, and to ensure that the company is in full compliance with all applicable laws and regulations.

This is the first of a two-part article based on a tutorial that I gave on spam and blogs at the LISA '06 conference in Washington, D.C., in December 2006. The second part, on blogs, will appear in a forthcoming issue.

Federal Inaction and the States' Responses

Spam is unsolicited commercial email. In the United States, Congress was reluctant to enact any legislation that would regulate spam even as its volume increased throughout the 1990s and the protests of the consumer lobby grew more and more audible. A reason why is obvious: The influence of the direct marketing lobby was stronger. Vendors discovered a unique tool for marketing to consumers, one that facilitates focused targeting to discrete market segments and is also ridiculously cheap per targeted recipient. They pressured Congress to do nothing that might increase the cost or impose compliance requirements on this new and very effective medium of communication.

The states were more receptive to the complaints of the consumer lobby. Federal inaction prompted a number of states to enact laws regulating spam. But these laws were inconsistent: They had different requirements and imposed different penalties from state to state. The lack of a uniform set of standards made compliance problematic, particularly for a medium such as the Internet that doesn't recognize state boundaries.

The California legislature enacted a new anti-spam law that was to be the toughest in the nation [1]. It would have become effective on January 1, 2004, and would have prohibited anyone from sending commercial email messages to any recipient who had not "opted-in" by giving his or her consent to receive spam from a particular sender in advance. Although the law could only be enforced in California, it would have had nationwide effect for all practical purposes. This is because spam campaigns can't usually tailor email messages to comply with the laws where each recipient happens to reside. Thus spammers would have had to comply with the most restrictive of the state laws in order to comply with them all—and that would have been California's. Sending spam only to recipients who had opted-in would have killed the direct marketing industry, because only a fraction of all possible recipients would ever be persuaded to opt in.

Congress Finally Acts

The direct marketing lobby found the California law to be intolerable. As a result, they changed their position on federal legislation and began to lobby Congress to pass a spam law that would supersede the very restrictive California law and that would also provide nationwide requirements. The result was the CAN-SPAM Act of 2003 [2].

CAN-SPAM regulates "commercial electronic mail messages" and not "transactional" or "relationship" messages. A commercial electronic mail message is one in which the primary purpose is the advertisement or promotion of a commercial product or service. CAN-SPAM requires that each commercial electronic mail message provide (i) a clear and conspicuous "opt-out" opportunity, (ii) a return email address for opt-outs that must work for at least thirty days after sending spam, (iii) a clear and conspicuous statement that the email is an advertisement or promotion, and (iv) a clear and conspicuous sender name and physical postal return address. It requires the sender to implement any opt-out request within ten days of receipt. It also prohibits false or misleading header or transmission information, subject lines, and content.

CAN-SPAM also contains special provisions for email messages containing sexually explicit material. The law requires senders to include a warning notice in the subject line and prohibits them from including sexually explicit material in the portion of the email that is immediately visible when opened.

Those initiating spam can be sued under CAN-SPAM if they don't fully comply with its requirements. The law also permits suit against companies providing services to spammers, such as those that assist advertisers with their campaigns for a fee. And companies whose products or services are promoted can also be sued if they (i) know or should have known about the noncompliance, (ii) benefit economically from the noncomplying spam campaigns, and (iii) do not attempt to prevent or report the noncompliance. ISPs cannot be sued under CAN-SPAM if their role is merely to transmit and they don't have a role or are not aware of the noncompliance.

CAN-SPAM gives standing to the following to enforce compliance: The Federal Trade Commission (FTC), certain other federal agencies, state Attorneys General, and ISPs. Unlike the California law, CAN-SPAM gives no private right of action. No noncompliant spammer can be sued for damages by any recipients. Recipients must convince one of the aforementioned entities to sue.

CAN-SPAM provides certain remedies in the event of a successful lawsuit. Money damages are available up to $250 per violation to a maximum of $2 million for nonwillful and $5 million for willful and knowing violation of the laws. Injunctive relief that will stop continued spamming is another permitted remedy. Those who violate the criminal provisions of the law can be sentenced to up to five years in prison. And the law provides for "bounty hunter" awards of up to 20% of the money damages assessed in a successful lawsuit.

Despite the availability of money damages, injunctions, and criminal penalties, CAN-SPAM is toothless compared to the California law and many of the other state laws that were passed before Congress finally acted. It allows spammers to send messages to everyone who doesn't opt out, rather than prohibit sending to anyone who hasn't opted in. It doesn't permit those most affected by receiving spam to sue the spammers for damages. And its compliance requirements are not all that difficult to meet. Furthermore, CAN-SPAM supersedes any state law, such as California's, that explicitly regulates commercial email messages. The only portions of the state laws that survive CAN-SPAM are those that prohibit falsity or deception in email messages or attachments.

Regulation of Spam by the FTC

Congress left the implementation of CAN-SPAM to the FTC and, in certain cases, other federal agencies. Congress gave the FTC authority to enact regulations under CAN-SPAM that have the force of law. Thus far, the FTC has developed rules further describing notice requirements for sexually explicit material [3] and establishing criteria for determining whether the primary purpose of an email message is commercial (in which case it is regulated) or transactional (in which case it isn't) [4]. It has also proposed but not finalized rules defining who is a "sender" and clarifying who has primary responsibility for responding to opt-out requests, shortening the deadline for honoring opt-out requests to three days, and prohibiting a sender from charging fees or requiring more information as prerequisites to honoring opt-out requests [5].

CAN-SPAM Compliance: Best Practices for the System Administrator

For the system administrator, there are two sides to the spamming issue: how to protect your system and its users from unwanted commercial email messages, and how to comply with CAN-SPAM if your employer uses email to market its products or services.

Of course you can (and should) maximize the protection of your system and its users against spam by installing the best filters and other anti-spam programs that become available. This is technological self-help. But you or your employer can also take legal action. Although CAN-SPAM doesn't permit spam recipients to sue spammers, you can notify your ISP, your state's Attorney General, or the FTC of any noncompliance and urge them to take action. CAN-SPAM does give these entities standing to sue violators.

The FTC uses its resources to investigate and prosecute the most egregious violators. But often it will not take action against those it views as marginal or not likely to serve as optimal test cases. ISPs are interested in providing good experiences for their customers, but they receive many complaints and have limited budgets for litigation. In my experience, clients often get best results by taking their complaints to their state's department of consumer protection or directly to the Attorney General.

My recommendation is to start by contacting the department of consumer protection in the state in which your company's computers reside. These departments will often have information you can use [6]. They can often be convinced to refer your complaint to the Attorney General, who can actually bring suit against the spammer. But you don't have to choose only one option. You can also inform your ISP and the FTC through their appropriate procedures. However, before you contact any of these agencies, be certain that the spam your system is receiving is actually violating the law, and get authorization from your employer to take the actions recommended here.

For those readers who work for companies that engage in commercial email campaigns, it is very important to comply with the law. Although compliance with CAN-SPAM is easier than with some of the state laws that it superseded, companies can still be fined, people can still be jailed, and employers can still suffer adverse publicity if they violate the law or any of the regulations that implement the law. It is therefore essential that system administrators have some familiarity with the law and those regulations and are able to work with their employers to ensure compliance.

The trend is for system administrators to participate with their employers in developing policy guidelines and templates that will ensure maximum compliance with the requirements of the law. At a minimum, companies should institute effective procedures for systematically implementing opt-out requests and for verifying compliance by its service providers. System administrators need to be aware of these procedures and should participate in their development and implementation. And system administrators should also clarify with their employers the scope of their responsibilities for monitoring and enforcing compliance.

REFERENCES

[1] The California law can be found at http://www.keytlaw.com/netlaw/caspamlaw.htm.

[2] CAN-SPAM stands for "Controlling the Assault of Non-Solicited Pornography and Marketing Act." A version of the CAN-SPAM Act can be found at http://uscode.house.gov/download/pls/15C103.txt.

[3] http://www.ftc.gov/os/2004/01/canspamfrn.pdf.

[4] http://www.ftc.gov/os/2005/01/050112canspamfrn.pdf.

[5] http://www.ftc.gov/os/2005/05/05canspamregformfrn.pdf.

[6] In California, for example, the agency is called the Department of Consumer Affairs, and it has a Web page specifically addressing what to do about spam: http://www.dca.ca.gov/ced/junkmailtips.htm.

THOMAS SLUYTER

all quiet on the western front

A DISCUSSION ON THE NECESSITY OF GOOD REPORTING

In daily life Thomas (a.k.a. Cailin) is a UNIX consultant for Snow BV. Aside from tackling the standard IT fare, he also tries to coach his colleagues on so-called soft skills. Thomas part-times as an Apple Macintosh evangelist and as a board member of the Dutch J-Pop Foundation.

cailin@kilala.nl

WHEN I RETURNED TO THE CONSULTing business back in 2005, I found that a change to my *modus operandi* would have favorable results for the perceived quality of my work. Up to that point I had never made a big point of reporting my ongoing activities to management, trusting that I'd get the job done and that doing so would make everyone happy. And, sure enough, things kept rolling and I indeed got things done. But I won't claim that anyone really knew what I was up to or that they had more than a passing notion of my progress.

2005 provided me with a fresh start; I decided that I'd do things differently this time around. And, indeed, as my reports started coming in, my client's response to both my employer and myself seemed to improve.

What's the Use of Reporting?

> "So, Peter, what's happening? Aahh, now, are you going to go ahead and have those TPS reports for us this afternoon?"
>
> *From the movie* Office Space

Reporting. Reports. Status updates. These are words that most people in IT dread and that conjure up nightmare images of piles of paperwork and endless drudgery with managers. Words that make you shudder at the thought of bosses nagging you about layout, instead of content, and of hours of lost time that could have been spent doing real work.

But seriously, spreading the word about your work and your projects doesn't have to be a huge undertaking and it will probably even help you in getting things done. By spending just a few hours every month you could save yourself a lot of trouble in the long run.

Good reporting has tangible benefits for the customer:

- It records solid and clear agreements about your activities.
- It provides regular updates of your project's current status and progress.
- It gives you a chance to adjust your work if things appear to be going wrong.

Benefits for your employer and yourself include the following:

- Solid and clear agreements about your deliverables are recorded.
- Your employer will have insight into your daily activities.
- You will be able to explain why you did certain things, should your decisions ever be doubted.
- The perceived quality of your work will increase. Reporting is something one expects from people who really take their work seriously.
- You will receive more direct feedback on your activities.
- Getting news about your project out to the rest of the company can create new business opportunities for both you and your employer.

Your First Report: Describe Your Assignment

"A lean agreement is better than a fat lawsuit."

German proverb

It may seem slightly odd, but your first report will be made before you've even done any real work. When you start a new project everyone will have a general idea of what you will be doing, but usually no one has all the details. To prevent delays in the future, you will need to make very specific agreements early on.

To get things started you will need to have a little eye-to-eye with your client to draft your assignment. You will be hashing out every significant detail of what the client expects from you:

- What will you be doing for them?
- How will you be doing it?
- In what timeframe are you expected to deliver?
- Which resources are at your disposal?
- Does the client impose any rules or demands?
- Are there any demands that you put on the client?
- Which milestones can be set and what repercussions will follow if you don't meet them on time?

The good news is that such a meeting usually doesn't take up more than an hour, maybe two. After that you'll need another hour or so to put it all on paper, provided that you have already created a template of sorts.

By putting as much detail as possible into all of these criteria you are creating many opportunities for yourself. From now on everyone agrees on what you will be doing and outsiders can be quickly brought up to speed on your project. At later points in time you can always go back to your document to check whether you're still on the right track. And at the end of everything you can use your original agreement to grade how successful you were in achieving your goals.

So, what if you will be doing "normal" daily management of the customer's servers and IT infrastructure? It doesn't seem like there's a lot to describe, is there? Well, that's when you put extra focus on how things are done. Mind you, even normal daily management includes some projects that you can work on.

Either way, make sure that all demands have been made "SMART": specific, measurable, ambitious, realistic, and time-bound. This means that everything should:

- Have clear boundaries
- Be verifiable
- Answer a specific need
- Be attainable within a certain amount of time

When your document is complete, go over it with your client once more to make sure there is agreement on everything you put onto paper. Then, get the client's approval in writing.

Here are two examples from my recent assignments. The first example was part of a real project with specific deliverables, whereas the second example covers normal systems management.

Example: Task Description, 1

Requirement 1: Improving on the old:

Our current Nagios monitoring environment is severely lacking in multiple aspects. These include but are not limited to the following:

- There is suboptimal design of the infrastructure involved.
- Many services, components, and metrics are effectively not monitored.
- There is suboptimal configuration when it comes to alarming.
- The current configuration is a dirty conglomerate of files and objects.
- There is no proper separation between users. People can see monitors to which they really should have no access.

Example: Task Description, 2

All of these issues should be fixed in the newly designed Nagios environment.

Thomas will take part in the department's normal schedule. This includes the following duties:

- Stand-by duty (being on call), once every five to six weeks
- Daily shifts, either starting his day at 08:00 *or* not leaving the office before 18:00
- The expanded schedule with regard to P1 priority incidents and changes, during which overtime is expected
- The department's change calendar, which involves regular night shifts to implement changes inside specific service windows

Expanding Your Activities

You have done your utmost to make your project description as comprehensive as possible. You've covered every detail that you could think of and even the customer was completely happy at the time.

Unfortunately, happiness never lasts long and your client's bound to think of some other things he or she will want you to do. Maybe there's a hitch in your deadline, or maybe you'll need to install a hundred servers instead of the original fifty. Who knows? Anything can happen! The only thing that's for certain is that it will happen.

When it does, be sure to document all the changes that are being made to your project. Remember, if your original project description is all you have to show at the end, then you'll be measured by the wrong standards! So be sure to go into all the specifics of the modifications and include them in an updated project description.

And, of course, again make sure to get written approval from the client.

Interim Reporting

Most people I've worked for were delighted to get detailed status updates in writing. Naturally, your client will pick up bits and pieces through the

grapevine, but clients won't know anything for sure until you provide them with all the details. I've found that it is best to deliver a comprehensive document every six to eight weeks, depending on the duration of your undertaking.

Each report should include the following topics:

- A short description of your project and of the customer you're working for.
- An overview of your original tasks, as detailed in your project description, and their current status
- An overview of any recent changes made to your goals and tasks, including a status update for these new tasks
- An overview of how you've spent your time over the past few weeks
- A list of problems and challenges that you've run into and how you've gone about solving them (including problems for which you'll need other people's help)
- A list of suggestions for your client
- Predictions regarding the outcome of your project

Example: A Short Description of Your Project

The goal of this project is to improve the monitoring capabilities at $CLIENT by completely redesigning the infrastructure, the software, and the configuration of the Nagios environment.

Example: Original Tasks and Their Status

Automated installation of UNIX servers:

Weeks 26 and 27 (28 June through 8 July) were spent full-time on the implementation of the Jumpstart server. $CLIENT had requested I give this part of the project the highest priority, owing to recent discoveries regarding the recoverability of certain servers.

At this point in time the so-called Jumpstart server has the following functionality in place:

[...]

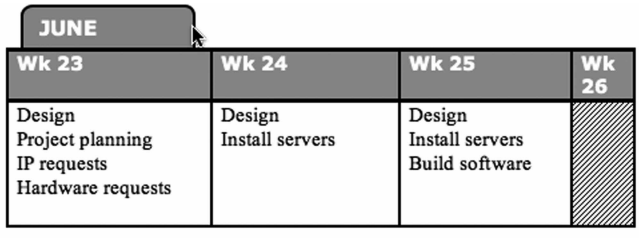
Therefore we can conclude that the Jumpstart server has reached full functionality.

Example: Changes to Your Project

One of the changes made to the project, based on the new technical requirements, is the switch from NRPE to SNMP as a communications protocol. This choice will allow us greater flexibility in the future and will most likely also save us some effort in managing the Nagios clients.

The downside of this choice is my lack of experience in SNMP. This means that I will need to learn everything about SNMP before I can start designing and building a project that's based upon it.

Example: A Simplified Timesheet



JUNE			
Wk 23	**Wk 24**	**Wk 25**	**Wk 26**
Design Project planning IP requests Hardware requests	Design Install servers	Design Install servers Build software	

Example: Problems and Challenges

On 17 July I issued a warning to local management that the project would be delayed because of two factors:

- My unfamiliarity with the SNMP protocol and software
- The lack of a centralized software (and configuration) distribution tool. This lack means that we shall need to install each client manually.

Example: Suggestions and Recommendations

$CLIENT is quite lucky to have a CMDB (Configuration Management Database) that is rather up to date. This database contains detailed information on all of its systems and has proved to be very useful in daily life. However, what is lacking is a bird's-eye view of the environment (maps and lists and such which describe the environment in less detail but show a method to the madness).

Example: Predictions Regarding the Outcome of Your Project

However, as can be seen from the included project planning, I will most probably not be finished with the project before the contract between Snow and $CLIENT runs out.

The contract's end date is set to 16 September, whereas my current estimates point to a project conclusion around 1 October. And that's assuming that there will be no delays in acquiring the backup and monitoring software.

Personal Contact

One of the biggest mistakes I've made in my recent past was to assume that my customers were reading every document I'd been giving them. I'd been sending them email about show stoppers and I'd been providing them with those beautiful reports I've been telling you about. But still something went horribly wrong. You see, some managers really don't care about technical background and thus they'll ignore certain parts of your reports. They figure that since you're not coming to talk to them, everything's hunky-dory.

This is exactly why email and big documents are no substitute for good old face-to-face contact.

Make sure to discuss your progress with your client and to mention any problems you've run into. You could even go the extra mile and request a regular, biweekly meeting! Talking to the customer in person will give you the chance to make sure the customer knows exactly what's going on and that he or she fully understands everything you've written in your interim report.

Everything Comes to an End

> "You can spend the rest of your life with me . . . but I can't spend the rest of mine with you. I have to live on. Alone. That's the curse of the Time Lords."
>
> *From 2005's season of* Doctor Who

Like all masterpieces, your enterprise needs a grand finale.

Now that all the work has been done and your goals have been reached, you will need to transfer responsibility for everything that you've made.

Cross the t's and dot the i's and all that. In short, you'll be writing an expanded version of the interim report.

The composition of your document should include the following topics:

- A review of the items in your task description. Have you met all your requirements and milestones? If you haven't, provide a satisfactory explanation as to why.
- Recommendations for any unfinished parts of the project. These should point your client in the right direction to finish the work at hand.
- A summary of how you spent all your resources. Think pie charts. Think graphs. Think budgets.
- A list of all of the issues and risks that you have noticed over the course of your project.
- A checklist detailing everything that you and the client need to agree on before the project can be truly considered "finished."

On the last page of your document, leave room for notes and signatures from your client and the lead technicians. Go over the document with everyone who'll need to take responsibility for your project. When they agree with everything you've written, have them sign that page. You will end up with a page that contains all the autographs you'll need.

Example: Task Review

Solaris Automated Installation Server:

[...]

Current status:

Finished December 2005. Unfortunately, there are a few small flaws still left in the standard build. These have been documented and will be fixed by $CLIENT.

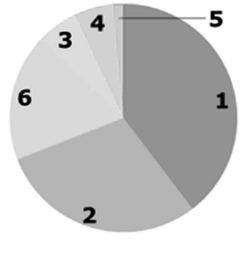
Example: Project Recommendations

A basic list of applications to be entered into phase 2 was delivered a few weeks ago. Now we will need to ascertain all the items that should be monitored on a per-application basis.

Once those requirements have been decided on we can continue with the implementation. This includes expanding the existing Nagios configuration, expanding the Nagios client packages, and possibly the writing of additional plug-ins.

Example: Resource Expenditure

	Activity	Hours
1	Nagios renewal	750
2	Daily management	550
3	Syslog server	110
4	Jumpstart server	100
5	Inventory of improv.	28
6	Everything else	300



Example: Risks and Pitfalls

These are areas identified by Thomas as risk areas that need addressing by the $CLIENT team:

- Limited knowledge of Nagios's implementation of SNMP
- Limited knowledge of Perl and shell scripting in lab team
- Limited knowledge of SDS/SVM volume management in lab team
- Limited knowledge of Solaris systems management
- Only one engineer in lab team able to support all aspects of UNIX

Example: Checklists

Nagios project items	Status
Upload all documentation to Sharepoint	Transferred to $CLIENT
Provide copies of all documentation and project files on CD-ROM	Finished
Perform high-level review of Nagios renewal project	Finished
Create and verify user accounts for SH and DR on new Nagios servers	Finished
Have SH and DR assume full responsibility of the project	Finished

In Conclusion

I've found that many of my customers were pleasantly surprised to receive detailed project reports. It really is something they're not used to from their IT crowd. So go on and surprise your management! Keep them informed, strengthen your bond with them, and at the end of the day take in the compliments at a job well done.

DAVID BLANK-EDELMAN

practical Perl tools: these inodes were made for walkin'

David N. Blank-Edelman is the Director of Technology at the Northeastern University College of Computer and Information Science and the author of the book *Perl for System Administration* (O'Reilly, 2000). He has spent the past 20 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

SINCE THIS IS THE SYSTEM ADMINISTRATION issue of *;login;*, I thought I'd take a break from covertly writing about sysadmin topics and overtly write about one instead. Today we're going to talk a bit about using Perl to walk filesystems. As filesystems continue to grow in size and complexity it helps to know what tools are available to make navigating these filesystems easier and more efficient.

(Ohhhh, I remember the good old days back when our filesystem had just two files: dot and dot-dot, and we were happy! . . .)

Best not to dilly or dally; we've got a lot of walking to do, so let's get right to it.

Putting One Foot in Front of the Other

The simplest way to begin to walk filesystems is to use the directory and file functions that are built into Perl:

```
opendir()
readdir()
closedir()
chdir()
```

Open a directory, read the contents of that directory, close the directory again, and then change directories to one of the subdirectories you just found. Repeat as necessary, most likely via recursion. Can't get much simpler than that.

This sort of approach appeals to the do-it-yourself crowd and those people who don't get enough recursion in their life (..their life..life..their life...in their life). In general I don't find using bare metal code like this particularly productive. It works fine for directory walks, where you might do something like this to find all of the files in a directory:

```
my $path = "/Users/dnb";
opendir my $DH, $path
      or die "Can't open $path: $!";
my (@files) = grep { -f "$path/$_" } readdir $DH;
closedir $DH
```

Here's a quick related aside that may keep you from banging your head against a wall someday: When first writing walking code of any sort, many people find themselves becoming very frustrated because their code only partially works. Close examination under a debugger shows that the

opendir(), closedir(), and even the readdir() all appear to be doing the right thing but for some reason their -f test doesn't return any results.

In my experience the most common reason for this is that the code they have written looks like this:

```
opendir my $DH, $path;
my (@files) = grep { -f } readdir $DH; # probably broken
```

This certainly looks as though it should work, since readdir() is correctly returning a list of the contents of $path. Pity we're not testing items in that directory, though!

We're performing the -f test on some (probably nonexistent) name in the script's *current directory* by mistake. If you want to perform a test such as -f on a file in some other directory, you need to specify that directory, as in our original code sample ("$path/$_").

If using the Perl built-in functions isn't the most productive way to spend your time, what is? Avid readers of this column know just where I'm going. Yup, it's module time.

File::Find

For a fairly long time, the File::Find module was the only game in town. And it is still a pretty good one. File::Find is shipped with Perl ("in the core"). It has steadily improved over the years, remaining a decent option for filesystems walking tasks.

Here's how it works: File::Find provides one subroutine, find(), which you call to start a walk from a specific path. Each time File::Find encounters something (e.g., a file or a directory) on its journey it calls a user-specified subroutine. This subroutine is responsible for doing all of the selection and disposition. It is the code that decides which items should be worked on and what to do with them. Here's a very simple example to make this a little clearer:

```
use File::Find;

find( \&wanted,'.' );

sub wanted { print "$File::Find::name\n" if ( -f $_ ); }
```

The first line says to begin the walk from the current directory and to call a subroutine called "wanted" each time it finds something. The wanted() subroutine checks to see whether the item in question is a file and, if it is, the full path for that name is printed. File::Find makes several useful variables such as $File::Find::name available for your wanted() subroutine while it is running, including $File::Find::dir (the current directory at that point in the walk) and $_ (the name of the item that was just found).

Your wanted() subroutine can be arbitrarily complex. Here's a contrived example that attempts a DNS lookup based on the name of every file found during the walk:

```
use File::Find;
use Net::DNS;

our $res = Net::DNS::Resolver->new;

find(\&wanted, "/data/suspect_hostnames");
```

```
sub wanted {
        next unless -f $File::Find::name;
        my $query = $res->search($_);
        print "ok $_" if defined $query;
}
```

This subroutine prints out the names of the files that successfully return an answer to the query. Notice that I said that wanted() *can* be arbitrarily complex, but here's the rub: It shouldn't be. This subroutine gets called for every single item in your filesystem, so it is incumbent upon you to write code to exit the subroutine as fast as possible. The code in the last example is a bad example of this, because DNS queries can take a relatively long time. If you can write a quick test that might lead to an early exit from the subroutine, by all means do it.

Before we move on to a better way to use File::Find, I do want to mention that the File::Find way of doing things was so compelling that Guido Flohr wrote a module that emulates it for use with complex data structures. In Data::Walk scalars are treated like files and lists and hashes are treated like directories. With Data::Walk you start a walk() of the data structure and a user-specified wanted() subroutine is called for each item in that structure.

File::Find::Rule

File::Find by itself makes the walking pretty easy through its simple interface. But this interface can be a bit too simple. It can make you work too hard to perform common tasks. For example, to collect a list of the files that are bigger than 2 MB you need to work out your own way to accumulate these results between calls to the wanted() subroutine. This means using global variables (yuck!), a caching/shared memory/persistence mechanism (pant, pant, hard work), or creating a closure ("Too much thinking give Oog a headache!"). To help make standard tasks like this easier there are a number of File::Find wrapper modules available. My favorite by far is the File::Find::Rule family of modules. This is the module I reach for most often for this sort of work (followed as a close second by the technique we'll see in the next section).

File::Find::Rule uses an object-oriented calling structure with a procedural interface also available if you'd prefer. This means you get to type lots of little arrows (->). To start off slow, here's the code that returns a list of all of the items in /var:

```
use File::Find::Rule;
my @items = File::Find::Rule->in('/var');
```

We can retrieve just the files or directories by chaining the appropriate method:

```
use File::Find::Rule;
my @files = File::Find::Rule->file()->in('/var');
my @dirs = File::Find::Rule->directory()->in('/var');
```

And that task we mentioned above—all the files above 2 MB in size—becomes as easy as pie:

```
use File::Find::Rule;
my @bigfiles = File::Find::Rule->size('>2M')->file()->in('/var');
```

Much more complex filtering expressions are also possible; see the File::Find::Rule documentation for more details.

If File::Find::Rule only provided a better interface to File::Find-like operations that would be enough reason to use it, but there's even more yummy goodness inside. I mentioned the File::Find::Rule *family* of modules before because File::Find::Rule provides an extension mechanism. Other File::Find::Rule::* modules can provide new predicates. A sample plug-in is File::Find::Rule::Permissions, which allows you to write things such as:

```
use File::Find::Rule::Permissions;
@files = File::Find::Rule::Permissions->file()->
        permissions(isReadable =>1, user => 'dnb')->in('/');
```

to determine which files in the filesystem are readable by the user "dnb." There are other extension modules that let you easily exclude CVS/SVN administrative directories (.cvs/.svn) based on image size or MP3 bitrate and so on. If you want to get really crazy, there's a module that can detect every time you've used a certain Perl operator in all of the scripts found in your filesystem:

```
# example from the documentation for File::Find::Rule::PPI
use File::Find::Rule ();
use File::Find::Rule::PPI ();

# Find all perl modules that use here-docs (<<EOF)
my $Find = File::Find::Rule->file
        ->name('*.pm')
        ->ppi_find_any('Token::HereDoc');
my @heredoc = $Find->in( $dir );
```

The Iterator Gang

File::Find::Rule may be one of my favorite tools for filesystem walking, but its default mode isn't always the best choice. For situations where you are dealing with a huge number of items, iterator-based modules can often offer a far better approach. (Note that File::Find::Rule also offers iterators.)

Iterators are well championed in the Perl community by Mark-Jason Dominus. The best exposition on the topic I have seen is in his excellent book *Higher-Order Perl*. In this book he says:

> An *iterator* is an object interface to a list.
>
> The object's member data consists of the list and some state information marking a "current" position in the list. The iterator supports one method, which we will call NEXTVAL. The NEXTVAL method returns the list element at the current position and updates the current position so that, the next time NEXTVAL is called, the next list element will be returned.

If you've used a module where you've called a next() method to get the next item back from some operation, for example an LDAP search, you've already dealt with iterators. In his book Dominus details a number of reasons why iterators are cool (and even shows you how to turn recursive code into iterator-based code). For our purposes iterator's most compelling argument comes into play when dealing with a huge filesystem. If you need to operate on the list off all of the files in a multi-terabyte filesystem, you don't want to use a module that will try to fill up all of your machine's memory with a massive list of names. An iterator-based module will let you work on that list one element at a time without having to store the whole thing at once. Clearly, this is a big win.

There are a number of Perl modules for filesystem walking that fit into this category; these include Iterator::IO, Find::File::Iterator, File::Walker,

Path::Class::Iterator, and File::Next. In the interests of time and space we'll only look at the last one.

File::Next is interesting because it describes itself as "lightweight, taint-safe . . . and has no non-core prerequisites" and also because of a new tool that makes use of it. We'll mention that tool in a second, but here's how File::Next gets used:

```
# example from the Find::Next documentation
use File::Next;
my $iter = File::Next->files( '/tmp' );

while ( my $file = $iter->() ) {
        print $file, "\n";
}
```

Simple, no? To make the iterator do more complex filtering, more parameters can be passed to the files() method. There is a similar dirs() method (although no methods are available to return special file types such as fifos).

It is possible to do some pretty powerful stuff with this module. The module's author has built a utility called Ack based on it. Ack (or App::Ack if you want to install it from CPAN) is a souped-up grep-like program that knows how to search filesystems in a more intelligent fashion. It knows to ignore certain files by default (e.g., backup files and core dumps) and can be told to search only certain types of files. Ack help types shows:

```
--[no]asm       .s .S
--[no]binary    Binary files, as defined by Perl's -B op (default: off)
--[no]cc        .c .h .xs
--[no]cpp       .cpp .m .h .C .H
--[no]csharp    .cs
--[no]css       .css
--[no]elisp     .el
--[no]haskell   .hs .lhs
--[no]html      .htm .html .shtml
--[no]java      .java
--[no]js        .js
--[no]lisp      .lisp
--[no]mason     .mas
--[no]ocaml     .ml .mli
--[no]parrot    .pir .pasm .pmc .ops .pod .pg .tg
--[no]perl      .pl .pm .pod .tt .ttml .t
--[no]php       .php .phpt
--[no]python    .py
--[no]ruby      .rb .rhtml .rjs
--[no]scheme    .scm
--[no]shell     sh .bash .csh .ksh .zsh
--[no]sql       .sql .ctl
--[no]tcl       .tcl
--[no]tex       .tex .cls .sty
--[no]tt        .tt .tt2
--[no]vim       .vim
--[no]xml       .xml .dtd .xslt
--[no]yaml      .yaml .yml
```

Ack offers a number of handy features such as color highlighting and full use of Perl regular expressions in addition to this filetype recognition.

Now that you've seen three approaches to filesystem walking and an application built on one of those approaches, I think it is time for you to hit the filesystem and start walkin'. Take care, and I'll see you next time.

ROBERT HASKINS

VoIP in an Internet service provider network

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

rhaskins@usenix.org

IN THIS ARTICLE, I TAKE A LOOK AT how "traditional" Internet Service Providers deploy Voice over IP (VoIP) [1] in their networks. VoIP gets a lot of press these days, and there seems to be no end to the hype surrounding it, especially when it comes to service provider networks. This article makes no attempt to cover the basics of VoIP (see Heison Chak's excellent column here in *;login:*), as this is a huge subject and there are plenty of Web sites available for learning more.

There are at least two ways "traditional" Internet service providers could use VoIP in their networks. The first (and arguably the oldest) method is as a replacement for voice trunks (trunking, either point-to-point or long distance) in their voice networks. This might be done to reduce cost or enable or enhance new voice services. The second way VoIP is typically deployed is as a replacement for a traditional landline phone (for example, a Vonage [2] or Packet8 [3]) type of service).

VoIP Trunking

Trunks are telephone lines that are used for point-to-point or long distance calling on provider networks. They carry calls from one point-of-presence (POP) to another, or from the POP nearest to the recipient to the recipient.

The use of VoIP connections as trunks has actually been around for a relatively long time. Many traditional long distance providers started using VoIP on their *dedicated* IP networks 10 years ago (or more) in a bid to lower cost. The key word here is "dedicated," because without a quality-of-service component, running VoIP on highly utilized non-QoS–enabled IP networks is usually a recipe for trouble. Once a QoS mechanism was designed and implemented, it made running VoIP on non-dedicated networks a more feasible proposition.

Providers might route calls onto their VoIP network in order to get them closer to the recipient's end office, thereby reducing cost. Once the call is closer to the recipient, the call would then be converted to a traditional voice trunk and handed off to the local carrier who has the callee (recipient) as a customer.

More recently, providers can route calls directly to VoIP trunks by purchasing VoIP trunks from a provider such as Junction Networks [4] or Covad [5]. Alternatively, with the proper interconnection (peering) agreements and facilities, providers can route calls directly to their caller's destination network with little (or no) cost. Of course, the interconnection (peering) agreements in question may or may not cover VoIP. These peering agreements are usually made in cases of relatively equal traffic. A provider with a small amount of traffic probably cannot make a peering agreement (for VoIP or anything else, for that matter) with a much larger network unless something else changes (for example, money changes hands).

End-User VoIP Service

A provider might want to provide traditional Plain Old Telephone Service (POTS) access for all inhabitants of a large, multi-unit building. One way to do this would be to deploy a VoIP PBX such as Cisco CallManager (CCM) [6] to support local phones in the building. The CCM would terminate VoIP calls for subscribers and send the calls (via traditional closet RJ11/RJ45 wiring) to the recipient's subscriber phone. This allows the provider to aggregate and oversubscribe traditional voice trunks, either locally or across their WAN. This model is closest to the traditional POTS most subscribers are used to.

One important point to bring out regarding CCM is the fact that it uses Cisco's proprietary Skinny Client Control Protocol (SCCP) [7] to communicate between CCM and the VoIP phone. This might cause interoperability issues if the provider were to change PBX platforms, although other vendors do support this protocol.

In the Packet8/Vonage type of service, the end subscriber uses what is known as an Analog Telephone Adapter (ATA), which converts a traditional analog phone to VoIP. This type of service gives the end subscriber the most services and flexibility, as it pushes the VoIP functionality and benefits as close as they can get to the subscriber. These services use the Session Initiation Protocol (SIP) [8].

The benefits to a Vonage-type service are that additional services can be offered to the subscriber and the service is portable. To move service from one location to another, the ATA is simply moved. The downside to this method is that the adapter requires 110 volt power and 911 service can be difficult, if not impossible, to offer with current technology in widespread use.

Provider-Side Equipment

Of course, there must be equipment on the provider side to process and route calls initiated by subscribers and provide other services such as voicemail. If a provider has an existing "legacy" telephone switch (such as a Lucent 5ESS), it can support VoIP call termination with an appropriate card installed. Although this works, typical traditional wireline switches don't have the functionality in terms of end-subscriber features found in more modern softswitches [9] such as Asterix [10] or Metaswitch [11].

Asterisk is an open source softswitch sponsored by Digium that runs on Linux and handles POTS and trunk (T1) lines with appropriate hardware [12]. Asterisk is widely used because of its large feature set and its low cost, running on Linux and similar open source platforms. The one down-

side to Asterisk is the difficulty in configuring the software for use, as it must be done by text file and the process can be quite time-consuming. There are a number of GUI interfaces for Asterisk; voip-info.org has a very comprehensive list [13] of Asterisk front ends.

Metaswitch is a carrier-based class 4/5 switch replacement typically deployed in a provider environment. Softswitches are very attractive to providers owing to their lower space, power, and cost requirements (not to mention subscriber feature sets) compared to legacy telephone switches. In provider networks that already have a class 4/5 switch, they are deployed alongside the existing switch. This is usually done for financial reasons, as the class 4/5 switch has paid for itself many times over and is cheap to run.

Billing

No article on deploying VoIP in a service provider environment would be complete without a mention about how to collect money from the subscriber. The requirements of the billing system are directly dependent upon the plans offered by the provider. Of course, if the provider is only offering a flat-rate service with no international calling component, then billing is much easier, as no call detail records (CDR) need to be processed.

If a service provider has a billing system that handles existing voice services, then it is not a huge issue to add a time-based VoIP service offering. All major softswitches provide the standard CDR that is easily processed by a voice-based billing system such as Oracle Infranet [14]. See voip-info.org for a nice listing of VoIP billing systems [15].

Summary

VoIP is a great way for traditional Internet service providers to lower costs and offer new services to their subscribers. VoIP trunks are a good way to reduce more traditional T1 long distance and point-to-point trunk costs while maintaining an acceptable QoS level, either across their own net or on shared networks. Providing end users with telephone service can be accomplished by using a VoIP PBX for multiple dwelling units or via the ATA adapter. Equipmentwise, providers can use open source solutions or commercial class 4/5 softswitches, depending on their level of comfort with open source and VoIP feature requirements. On the billing side, VoIP integration is dependent upon what plans the provider opts for and what billing system the provider currently has in place. Most softswitches have standard CDR support, making billing integration relatively easy.

I wish to thank Jeff Manning and Pete Carey for their input into this article.

REFERENCES

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

[2] Vonage: http://www.vonage.com/.

[3] Packet8: http://www.packet8.net/.

[4] Junction Networks: http://www.junctionnetworks.com/index.php.

[5] Covad: http://www.covad.com/.

[6] Cisco Unified CallManager: http://www.cisco.com/en/US/products/sw/voicesw/ps556/index.html.

[7] SCCP: http://www.javvin.com/protocolSCCP.html.

[8] SIP: http://en.wikipedia.org/wiki/Session_Initiation_Protocol.

[9] Softswitch: http://en.wikipedia.org/wiki/Softswitch.

[10] Asterix: http://www.asterisk.org/.

[11] Metaswitch: http://www.metaswitch.com/.

[12] Digium cards for Asterisk: http://www.digium.com/en/products/.

[13] Asterisk GUIs: http://www.voip-info.org/wiki-Asterisk+GUI.

[14] Oracle and Portal Infranet billing software: http://www.oracle.com/portalsoftware/index.html.

[15] VoIP billing systems: http://www.voip-info.org/wiki/view/VoIP+Billing&view_comment_id=13159.

HEISON CHAK

Asterisk appliance

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

heison@chak.ca

WITH COMMUNITY EFFORT TO enhance Linux and other UNIX-based embedded systems, the creativity and ingenuity of these users are driving hardware manufacturers to deliver more robust and ever improved products. Both the Linksys WRT54G and the NSLU2 (a.k.a. "slug") have attracted developers and users to modify firmware and hardware to suit their custom needs and gratification. This article will focus on running the Asterisk PBX on Linksys NSLU2 with the Unslung firmware.

What Is NSLU2?

Linksys NSLU2 is a NAS (Network Attached Storage) unit designed to share USB storage via the SMB protocol. NSLU2 stands for Network Storage Link for USB 2.0 disk drives. The NSLU2 features an Intel IXP420 (ARM) CPU, 32 MB of RAM and 8 MB of flash, an Intel IXP425 Ethernet interface, and a dual-port USB controller. Under the hood, it has a Linux-based embedded OS driven by a Web front end for users to manage and share the two USB-connected devices.

The devices can be USB memory sticks, USB card readers with a removable medium (CF, SD, etc.), or USB hard drives. Although NSLU2 features a 266-MHz ARM processor, it is limited to operating at half the speed of a 133-MHz processor, perhaps because of heat dissipation requirements in its small footprint. Despite the slow response and low throughput, the NSLU2 is an inexpensive way to add a hard drive or two to a network.

With community efforts, the NSLU2 can be flashed with different firmware depending on your interests:

- Linksys (original firmware, driven by Web interface, based on Linux 2.4.22)
- Unslung (modified firmware for beginners with little Linux knowledge, based on Linux 2.4.22)
- SlugOS (for experienced Linux users who need software from different repositories, based on Linux 2.6)
- OpenSlug
  - OpenDebianSlug
  - SlugOS/LE
  - GentooSlug
  - UcSlugC
- Debian/NSLU2 (based on SlugOS kernel patches, Linux 2.6)

NSLU2 Firmware Replacement

The original Linksys firmware is based on a 2.4.22 Linux kernel and uses RedBoot as its bootloader. RedBoot expects to find the kernel in the 8-MB flash at address 0x50060000 (/dev/mtdblock2).

Flash memory is partitioned into four "/dev/mtdblock" devices, as shown in the following table:

Linux device	Type	Start address	Length	Description
/dev/mtdblock0	RedBoot	0x50000000	256 KB	Contains code from which the IXP420 boots
/dev/mtdblock1	System	0x50040000	128 KB	NSLU2 configuration Config, e.g., IP address
/dev/mtdblock2	Kernel	0x50060000	1 MB	Linux kernel
/dev/mtdblock3	RAMdisk	0x50160000	6.625 MB	Ramdisk image for /

Unslung is one of the many firmware replacements for NSLU2; it is based on the same firmware that the manufacturer ships except that it enables support for telnet and unslinging. Unslinging refers to the procedure of copying the filesystem to an attached USB device (e.g., a 2-GB USB stick is used for testing) and booting off the external device. Additional space on the root device allows installation of extra software and other custom packages.

The Unslung 6.8 firmware, along with instructions on how to flash the firmware, can be found at http://www.slug-firmware.net/u-dls.php. It is crucial to note that the firmware *must* be flashed when *no* USB device is connected. Once the firmware is updated, telnet can be enabled and then one can run the unsling script (/sbin/unsling). This divides the target USB device into three partitions (ext3 root, ext3 conf, and swap).

Although using a flash drive is often an attractive way to run an Unslung NSLU2, the low-power, low-cost, and quiet appliance may be exposed to the danger of failing owing to wearing of the flash drive. To extend the life of flash devices, it is best to disable swap and mount the ext3 filesystems so that access time is not updated.

If a file named /.ext3flash exists, the Unslung boot scripts will remount the root and conf partitions with the appropriate options (`-o noatime` for both and `sync` for conf) and disable swapping to the root drive by running swapoff.

Installing Packages on NSLU2

Similar to Debian apt-get, an Unslung NSLU2 can install ported packages onto the external USB flash by using the `ipkg` command. There are 900+ packages available for the Unslung firmware. For those who are more adventurous, OpenSlug is a another firmware to explore, as it has more ported packages and is better suited for compiling and building packages.

The update and upgrade options of ipkg work in similar fashion as in Debian; it looks for package lists in the archives found in /etc/ipkg/[cross|native]-feed.conf and performs upgrades to installed packages and their dependencies:

```
# ipkg update ; ipkg upgrade
```

For the purposes of this article, the NSLU2 is intended to run as an Asterisk server. Thus, the following packages are installed:

```
# ipkg install openssh ntp asterisk14 asterisk14-core-sounds-en-ulaw
    asterisk14-extra-sounds-en-gsm asterisk14-extra-sounds-en-ulaw
    asterisk14-gui
```

Packages installed by ipkg live under /opt; otherwise they reside in their normal path. In the case of Asterisk, the configuration files can be found under /opt/etc/asterisk and the sound files are located in /opt/var/lib/asterisk/sounds. Running Asterisk on the NSLU2 isn't much different from running it on x86 or AMD CPUs.

Connecting NSLU2 to the PSTN

Once Asterisk registers itself to a SIP or IAX provider, VoIP handsets and soft clients provisioned to use the NSLU2 can accept calls from the PSTN through the VoIP service provider:

```
SIP.conf: register => username:password@voip.service.provider
```

This is probably good enough for those who are planning to run Asterisk as a pure VoIP application. For those who want to connect Asterisk to the PSTN via FXO (Foreign Exchange Office) or prefer to connect an analog phone to Asterisk, NSLU2 doesn't quite meet the bar.

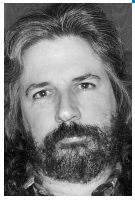
One can go about writing one's own driver for USB-based FXO/FXS interfaces and worry about fitting all those drivers in only 32 MB of RAM. But why reinvent the wheel when there is an inexpensive and quick solution around the corner?

The Grandstream GS-488 ATA and the Linksys 3102 ATA (analog telephone adapters) are examples of SIP-based media gateways that feature both FXO and FXS interfaces. Both are affordable low-power, low-cost, and quiet VoIP appliances. (The GS-488 has been reported as unable to pass inbound callerID to SIP. Other than this small glitch, the two pieces of equipment are comparable.) The ATAs will take the burden of POTS line and analog devices (phone or fax) out of the NSLU2. Incoming calls from the PSTN to the POTS line will come through the ATA, then get converted to SIP UDP packets destined for Asterisk (e.g., an IVR menu). If the PSTN incoming call ends up going to an analog phone (as a result of the IVR menu), it can be sent as SIP UDP packets from Asterisk to the ATA and ring on its analog port.

If you terminate inbound VoIP calls from the service provider and originate VoIP-based calls to the PSTN via the service provider, you now have the ability to do this on either the NSLU2 or the ATA. You can also use the ATA as the interface for other Asterisk installations, such as the one I described in my April 2007 column which runs within a virtual machine without needing to install device drivers.

ROBERT G. FERRELL

/dev/random



Robert G. Ferrell is a geek-without-portfolio currently residing in an apartment with a wife and two cats waiting on the glacial gears of government, which is akin to watching paint peel, only not so engaging.

rgferrell@gmail.com

The author's podcast of this article can be heard at http://www.antithetical.org/restlesswind/plinth/Podcasts/login_0407.mp3.

FEW DENIZENS OF THE COMPUTING wilderness have been more revered, reviled, rebutted, and reconciled than the wily system administrator. From Simon Paul Travaglia's legendary *Bastard Operator from Hell* to my own hapless Jake in *Chasing the Wind*, this species has gotten more attention than perhaps any other single deity in the digital pantheon. Apart from their tendency to be autocratic and possess an overriding obsession with config file tinkering, however, what do we really know about these often elusive creatures?

I happen to have been a student of sysadminology for, lo, these past two and a half decades, and as a result have a number of pointed observations to share on the subject. Most of my observations are pointed, come to think of it. I think it's because the pointed ones have better penetration.

The first of these pithy and poignant postulates concerns the lair of the beast. If you're wondering whether there is a WORKSPACE, SYSTEM ADMINISTRATOR'S, STANDARD, ONE EACH, the answer is, "negative, soldier." Sub-basements to penthouses, from dark 'n' dank to lofty 'n' looming, the habitat of the sysadmin is as variable as the campaign rhetoric of a congressional incumbent. There was a time when the unifying element of virtually all lairs was that they were chilly, owing to thermal requirements of the server rooms where sysadmins usually lurked, but even that has largely fallen by the wayside for modern practitioners of the art as more efficient heat dissipation systems and cooler-running components have evolved.

Whether considered as oubliette or *haute couture*, the digs of the sysadmin have certain commonalities, perhaps the most ubiquitous of which are spare parts. Scarcely exists there a system administrator's lair (slair) that has not within easily observable radius a plethora of software boxes, cables, trays, drives, stripped wires, adapters, backplanes, and a host of less readily identifiable fiddly bits. The very essence of the system administration experience can be reduced to the simple phrase "make it work." An intrinsic byproduct of the pursuit of that noble goal is the amassing of a sizeable collection of electronic gewgaws, which then tend to pile up on every available horizontal surface as the eons crawl inexorably by.

Another class of paraphernalia one often encounters whilst strolling through the slair habitat is that of the fecund reference publication. Ordinarily at least two full shelves will be devoted to this esoteric collection of apparently self-replicating printed material, despite the fact that the vast majority of it will be available on and consulted, when consulted at all, via electronic media. This is because sysadmins never throw anything away. The largest regiment of this cellulose battalion will consist of badly translated installation and troubleshooting guides that are of limited use in any capacity except perhaps as exhibits in a study of the relationship between linguistics and major military actions. The seasoned sysadmin will have survived a number of instances of encountering, when alacrity was of the essence and lucidity of instruction therefore an absolute necessity, admonitions such as, "Plugging board into slot not recognized, do not create difficulty when pushing down with top insertion." One common battle scar of the grizzled veteran of multiple system administration campaigns is a chronic scalp abrasion resulting from too much time spent head-scratching over this sort of literary surrealism.

System administrators have suffered under the onus of being perceived as oppressive information technology resource hegemonists for as long as multiuser computing environments have existed. To a certain degree this reputation is justified, inasmuch as the enforcement of chafing restrictions such as access control and disk quotas falls squarely on the sysadmin's shoulders and he or she must therefore play the "heavy" on occasion. However, the popular notion that sysadmins enjoy, relish, covet, or revel in this disciplinarian role is unfair and wholly in error. The system administrator is a humble servant of the users, dedicated to providing them with the best possible computing experience at all times. To suggest otherwise is a great way to get your privilege level reduced to "invertebrate."

Love them or loathe them, were it not for the perseverance of these dedicated professionals the topology of business and academic computing as we've known it simply could not exist. In light of this basic truth, I thought perhaps a few tips regarding the care and feeding of the person upon whom your computational happiness depends might be in order.

Never, ever give a sysadmin gooey confections in the workplace. Tech folks tend to have sweet teeth and often won't be able to resist tearing into the box of melt-instantly-in-your-hand chocolate-covered cherries right then and there. The havoc this can wreak with keyboards, network equipment, user account request forms, and general datacenter surfaces is simply not worth it. Similar caveats apply to crumb-generating snacks, any plant materials that could be rolled into a cylinder and smoked, or liquor. In fact, it's safest not to bestow upon your friendly neighborhood sysadmin any substance the immediate consumption of which might contribute to systems outages. Unfortunately, that narrows the prospective consumable gift list pretty much to carrot sticks and turkey jerky, the presentation of either of which may tend, once again, adversely to affect your user privileges (see "invertebrate," above).

The sysadmin does not, as a rule, thrive in strong sunlight. A dimly lit cubbyhole punctuated by blinking LEDs is the natural environment of the species, and any misguided attempt by management to "brighten up the place" will only send the resident scurrying away into the shadows, there to plot horrific vengeance. Similar failures will accompany initiatives centering on improving the decor by taking down the graphic novel posters, MMORPG screen shots, and obscure indie music CD covers scattered hither and yon throughout the slair. The feral sysadmin is a territorial creature that doesn't react well to intruders upon its domain. Any forced alterations to the ecosystem are likely to be met with considerable eye-rolling, if not downright aggression.

It has been suggested, albeit not by credible authorities, that the sysadmin is related to the wolverine. They both snarl rather viciously and tend to dig up the garden at night. I may be confusing wolverines with weimaraners, though.

book reviews

ELIZABETH ZWICKY, MING CHOW, AND RIK FARROW

CODE CRAFT: THE PRACTICE OF WRITING EXCELLENT CODE

Pete Goodliffe

No Starch Press, 2007. 558 pages.
ISBN 1-59327-119-0

It doesn't take that much programming to discover that programming is harder than it seems. It's not really that hard to turn out code that works, for some definition of "works." For instance, I once worked for a programmer who insisted that her task was done when the code compiled without error, and anything further was debugging, which could be undertaken by somebody else—an attitude I didn't discover until she declared something done when it did not in fact run at all, even a little bit. In a happy twist of fate, she later ended up working for a friend's least favorite boss, and in our more twisted and bitter moments we take joy in imagining their working relationship.

If you want to be a good programmer and not end up on the wrong side of stories like this, you need to worry about a lot more than getting the code to basically fulfill its intended function. You have to be able to pass it on to another programmer. It has to survive rounds of modification. It has to grow and change. You have to work with other programmers. *Code Craft* is meant to help you achieve all those extra goals, and from my point of view, it does a pretty good job. It also has a monkey cartoon in every chapter, just to sweeten the pot (and although the topics are dense, the writing is clear and vivid enough to keep you going without the monkeys).

This is a great book for somebody who has basic technical competence but wants to learn how to make that fit into the bigger picture. I don't agree with every detail, but I agree with all of the author's main points, and I think a programmer who absorbed them would be much more fun to work with than one who disagreed with them.

The book does not mention installation and administration issues, which are close to the top of my "Ways programmers become loathed" list, but it's primarily concerned with keeping other programmers from loathing you. (It doesn't suggest firmly that unless you are unusually talented or a GUI designer moonlighting as a programmer, your user interfaces will be horrible unless you get help, either, and that's something lots of programmers need to hear.) It does mention that security is important and requires thought, but it omits the all-important warning "Don't design your own crypto." Perhaps what it's really missing is a big list headed "Things you might think you know something about but almost certainly don't."

HEAD FIRST OBJECT-ORIENTED ANALYSIS AND DESIGN

Brett D. McLaughlin, Gary Pollice, and David West

O'Reilly, 2007. 589 pages.
ISBN 0-596-00867-8

One of the blurbs for *Code Craft* is from a four-year-old (with a last name suspiciously like the author's), who likes the monkeys best. Let me tell you, she'd love the Head First series. In fact, I had to hide while reading it, because my daughter is currently passionate about dogs and speech bubbles (or thought balloons), and there are a lot of examples that combine the two. It turns out that she keeps asking you to read what people are thinking even after it turns out to be "Look at all those Strings! That's terrible! Shouldn't we use objects or constants instead?" or even less comprehensible stuff, over and over again. She was entranced. My husband, however, was instantly suspicious. Dogs and thought balloons are not a combination that makes him think "Serious programming book." (I haven't enquired about monkey cartoons. I think it is entirely likely that he finds them more suitable. The relationship between monkeys and programmers is all too evident to anyone who deals with programmers regularly. Don't throw things! I program for a living at the moment.)

Supposedly, all this hullabaloo that delights the small child and makes the adults wary is scientifically supported as a way to keep you from turning off your brain and doing the sort of "smile and nod" routine that gets you through cocktail parties but leaves you at the end of a technical book without actually having learned anything. I'm not sure, but I didn't find it unfortunately distracting, and I was occasionally moved to actually do parts of the exercises, and I am the world's worst person at doing exercises. Threats and pleading on the part of the author almost never move me to pull out a pencil—normally only the hope that

I will discover some terrible error will move me to even read exercises. But apparently in the war between "Don't do homework" and "Do puzzles," doing puzzles does sometimes win.

Along the way, I did absorb some of the principles and terminology of object oriented design and analysis; I probably would have gotten more if I'd stuck with the exercises and I weren't currently programming in an in-house language whose eccentricities preclude the use of many lovely design principles. This would make a particularly nice introduction for somebody who doesn't see the point. It does a good job of showing why design matters and how design issues relate to real-world programming.

ESSENTIAL CVS, 2ND EDITION: VERSION CONTROL AND SOURCE-CODE MANAGEMENT

Jennifer Vesperman

O'Reilly, 2007. 395 pages.
ISBN 0-596-52703-9

I admit to a certain sense of relief that this is a very straightforward book. No monkeys. No dogs. Either you have a burning need for it (e.g., you are trying to figure out which end is up while using CVS) or you don't. If you need to know more about CVS, this is a good, explanatory reference that will keep you from the sort of hazy hand-waving incantations that go on all the time at my place of work. (I blush to admit that I'd been using the contradictory option combination "-d -P" for quite some time because, well, that's what somebody told me to do and nothing bad happened. Also it was keeping the tigers away from my desk, apparently.) The book does a nice job of explaining differences between releases (which is important in a heterogeneous environment, where my CVS has all the latest flashy features, the servers have almost all of them, and alas, half my colleagues are in the dark ages).

If you aren't deeply interested in CVS, go read something else.

THE ART OF SOFTWARE SECURITY TESTING: IDENTIFYING SOFTWARE SECURITY FLAWS

Chris Wysopal, Lucas Nelson, Dino Dai Zovi, and Elfriede Dustin

Symantec Press, 2007. 250 pages.
ISBN 0-321-30486-1

It's nice to see a book that talks about software security testing in a sane and sober way. This book is intended for people testing their own software, and it discusses attack tools in the testing context, with a brief discussion of how software security and its testing fit into the development model. Oddly, it doesn't talk much about what I've always found to be the hardest part, convincing developers to care, although it does address it indirectly by talking about the cost of fixing security flaws at various points in the process (e.g., it's a lot cheaper to notice you need encryption before you ship a million copies and end up on the front page of a newspaper).

On the whole, I found the book unsatisfying. It's got a bunch of good information, but I'm not sure who the audience is. It seems to be meant for testers who don't have a security background, but there's really not enough information about security to let them test things effectively and know what's an important vulnerability and what's not.

There's a completely gratuitous ad for a Symantec product thrown in; I'm sure that Symantec's vulnerability database is a lovely thing, but what exactly has it got to do with patch management for your product, where you're trying to release patches, not figure out what patches you need to install?

HOW TO BREAK WEB SOFTWARE: FUNCTIONAL AND SECURITY TESTING OF WEB APPLICATIONS AND WEB SERVICES

Mike Andrews and James A. Whittaker

Addison-Wesley Professional, 2006.
240 pages. ISBN 0321369440

REVIEWED BY MING CHOW

As the number of Web applications increases, so do the risks of exposing the most critical business and personal data. Despite the growing acceptance of application security, security testing in the QA process is still largely lax. Glancing at this title, one would have the impression that it resembles one of the "Hacking Exposed" books. However, Andrews and Whittaker did a very good job in not doing exactly that.

This book covers all the most current attacks and issues pertaining to Web applications and services. The attacks are well organized into specific categories: client-based, state-based, user-specific input, language-based, server-based, and authentication. Attacks such as SQL injection, cross-site scripting, buffer-overflows, and even fake cryptography are discussed. For each attack, the "when to apply attack," "how to conduct attack," and "how to protect against attack" are described with very good illustrations using code or screenshots. The book is largely independent of language choice. However, the book delves into using a plethora of tools (e.g., Witko, NitkoParos, SSLDigger) for security testing. Finally, the book concludes with a discussion of privacy issues and threats in Web services.

The primary focus of this book is on testing. This is not a book on

how to exploit servers or applications in gory detail. This is also not a book on how to write secure code, or how to design secure software. However, it gives a concise overview of all the current problems in Web applications and services. This book is handy for anyone working in software QA. It also serves as a good introduction to Web application security for new developers.

INSIDE THE MACHINE: AN ILLUSTRATED INTRODUCTION TO MICROPROCESSORS AND COMPUTER ARCHITECTURE

Jon Stokes

No Starch Press, 2006. 320 pages.
ISBN 1-59327-104-2

REVIEWED BY RIK FARROW

Have you ever wondered just what Intel has been doing to make its processors run faster and cooler? Stokes provides an illustrated view into key areas of processor technology that explains the big issues in processor design. I had a good idea of what pipelining and superscalar meant before I started reading this book; I had actually learned about pipelining in a hardware design course in the late 1970s. But Stokes makes it easy to understand what these terms mean, and what effect they are supposed to have on processor performance.

Much of this book is based on articles published previously by Stokes at arstechnica.com, and if you have been following his articles, some of this material will be familiar. Chapters have been added to the beginning to fill in some basics and terminology, for example, as well as providing some explanation of the purpose of cache. The writing is clear, and the four color diagrams make a difference in coming to grips with a complex topic.

Stokes focuses on just two processor families, Intel x86 and IBM Power RISC (used in earlier Apple Macs). AMD barely gets mentioned during a discussion of 64-bit features. But that explanation did make it clear why you must have a 64-bit operating system to use these features and that you can still run 32-bit programs, that is, you are not locked into 64-bit-wide datapaths, in x86-64.

By the end of the book, I understood a primary difference between Core Duo and Core 2 Duo (the latter have wider internal data paths for doubles, which were missing in previous Intel desktop-targeted processors), as well as what MMX, SSE, and SSE2 are supposed to do. Reading this book should help you choose the right processor, as manufacturers, such as Intel, target their designs for particular workloads, and buying the right processor, with the right amount of cache, can really make a difference.

As Stokes himself writes, this is *not* a college textbook on computer architecture and related software (compilers and APIs). For that, you want Patterson and Hennessy's *Computer Organization and Design: Third Edition* (Morgan-Kaufman, 2004).

NICHOLAS M. STOUGHTON

why standardize?

USENIX Standards Liaison

nick@usenix.org

Descriptive or prescriptive? Should a formal International Standard describe a single existing product, or should it prescribe how future products should work? Is it an abuse of the process for any one company to force a document describing its product through the standards mill?

Most of the formal standards on which I work describe some sort of an interface. They act as contracts among multiple producers of the interface and multiple consumers of its services.

For example, POSIX describes the interfaces provided by a wide variety of different operating system implementations (e.g., Solaris, AIX, HP-UX) and portable applications that should compile and run on any of those implementations. The C standard describes the language a compiler must be able to translate and provides a guarantee that an application that strictly conforms to the language standard will always run the same when translated with a strictly conforming compiler.

As such, standards such as POSIX or C are *prescriptive*; they dictate behavior on both sides of the interface. They tell an implementer what must be done in order to conform, and they tell an application developer how to write portable code that does what was intended.

Occasionally, however, a *descriptive* standard is needed. The Linux Standard Base (LSB) is one such standard. It describes the binary interface between an application and a Linux distribution. It is still prescriptive in the sense that a distribution that fails to ship the versions of the libraries or symbols required does not conform, but it does not tell implementers how to write interfaces. Instead, it describes what the implementation does. Linux hopes to be POSIX conforming (and almost is), so POSIX can tell implementers how to write interfaces; the LSB only describes the size and shape of the binary interface itself.

When we are developing the POSIX standard, however, every attempt is made to take into account existing and historic practice (including Linux behavior). In that sense, even POSIX ends up being somewhat descriptive. POSIX tries to specify the things that every implementation must do (by looking at what every implementation actually *does*), and it leaves unspecified those areas where implementations may differ.

But when a standard is descriptive you have to start asking yourself, "Who is the intended user?" and "What is the intended purpose?" If the standard simply describes something that *is*, a single product with a single implementation, rather than the way a conforming implementation should work, why bother? The whole purpose of the standard is to allow competing implementations and to allow portable applications. In the LSB case, the answer to the question is easy; the intended user is any application developer targeting the Linux platform. The purpose is to ensure binary compatibility among distributions. It does prescribe what distribution vendors must implement in their products, and it does promise conforming application developers

binary portability across conforming distributions.

But let's look for a few minutes at another standard wending its way through the process at present: Office Open XML (OOXML). The name is (deliberately?) confusing: It is Office Open, and *not* Open Office! This is a standard developed by Microsoft for their Office product.

At this point, OOXML is an approved standard within Ecma International (the people who brought you Ecmascript). It is sometimes known as Ecma Office Open XML, or EOOXML. The Ecma standard (Ecma-376) is being submitted to ISO/IEC JTC 1 via the "Fast Track" process. If this process were to be completed, Ecma-376 would be an international standard that describes a file format used by Microsoft Office. No other product could ever fully conform to it. There is no good technical purpose for this standard. Of course, the Ecma committee doesn't actually *say* that anywhere, but should you attempt to read the 6,000+ pages, you would find numerous places with phrases such as "This element specifies that applications shall emulate the behavior of a previously existing word processing application" (Microsoft Word 95), but nowhere will you find a description of what that behavior is. If you are trying to write an application that reads an OOXML file, what chance do you have of implementing this? In Microsoft's defense, most of these requirements are marked as "deprecated" and describe "compatibility" settings. However, they are still a mandatory part of the standard for conformance. How else are they going to achieve their goal of having every document ever produced by MS Office capable of being expressed in OOXML?

What the Ecma committee *does* say is:

> The goal of the Technical Committee is to produce a formal standard for office productivity applications within the Ecma International standards process which is fully compatible with the Office Open XML Formats. The aim is to enable the implementation of the Office Open XML Formats by a wide set of tools and platforms in order to foster interoperability across office productivity applications and with line-of-business systems.

Sounds great, right? But it is a lofty goal that they have singularly failed to achieve or even approach. In their press release announcing that the standard had been approved by Ecma International, they say, "The new open standard safeguards the continued use of billions of existing documents." And that's the point: Safeguard MS Office documents. Any legacy MS Office document can be converted, by MS Office, into the OOXML format, and any conforming application must be able to handle the result.

The charter for OOXML expressly locked the standard to Microsoft Office. (Recall that bit about "fully compatible with the Office Open XML Formats.") So the entire standardization process within Ecma International was tied up by the language that expressly forbade any attempt to deviate or improve on the Microsoft format. The process was not open, and it does not represent industry consensus (both stated goals of the ISO/IEC process).

OOXML describes a file format. Granted, a file format standard is somewhat different from a straight interface type of standard. But it's not that different: It should be possible to write an application that produces such a file and know that any other conforming application can read the same file and produce the same results. In the OOXML case, there can only be one application that can produce or consume the file: Microsoft Office. To pretend that this is a standard that will increase document portability is an outright lie.

And to make matters worse, ISO/IEC already has a standard that is in exactly the same space, is well written, and is well adopted: Open Document Format (ODF). As I write this, OOXML is at the start of a six-month ballot. The first thirty days of that period is to note "any perceived contradiction with other JTC 1, ISO or IEC standards." National bodies (e.g., those of the U.S. or the U.K.) can submit comments on such "perceived contradictions," and if any are received, the five-month remainder might be put on hold. Groklaw has been compiling a list of contradictions to submit. We shall see what happens.

There can only be one purpose in Microsoft attempting

to push the OOXML standard through: to be able to state that it has a standard conforming product. It is a simple knee-jerk reaction to ODF, the accepted standard that has locked them out of some major government contracts. And there can only be one user of the standard: Microsoft.

Now, don't get me wrong; I'm glad that Microsoft has published a specification that tells others the nitty-gritty details of one of the file formats used by Office. It just doesn't need to be an international standard. The only possible beneficiary of the effort to make it one can be Microsoft. The company has even managed to get some very clever wording into the legal license arrangements that appears to prevent unauthorized (by Microsoft) implementations.

This controversial standard is not without its supporters, and not entirely without merit, at least at the theoretical level. There's more than one programming language, so why shouldn't there be more than one office document format? And ODF is not perfect in every respect.

The OOXML document really sums up everything that is wrong with the ISO "Fast Track" process, and it may indeed even lead to the downfall of that process. The Fast Track process was originally designed to allow a standard developed through an open process in another standards development organization (SDO) to speed through the ISO process. In theory, the document had already accepted adequate comment and review from that SDO itself.

However, some SDOs (and Ecma is a particularly notable one here) have simply served as backdoor ways to get a poorly reviewed, proprietary document published by ISO. If OOXML gets through these next six months unscathed, the Fast Track process will be seen by many as utterly worthless. The process is used as a sort of shell game: The message to Ecma members is, "Oh, don't worry if this isn't perfect . . . it will be reviewed again by ISO," while saying to ISO members, "Oh, you don't need to put any effort into this document in ISO . . . it has already been reviewed by Ecma."

An interesting perspective (and my motivation for this article) can be found at http://www.robweir.com/blog/2006/01/how-to-hire-guillaume-portes.html.

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, VoIP, and operating systems, book reviews, and summaries of sessions at USENIX conferences.

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

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

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

DISCOUNTS on registration fees for all USENIX conferences.

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

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

TO JOIN SAGE, see www.usenix.org/membership/classes.html#sage.

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

Communicate directly with the USENIX Board of Directors by writing to *board@usenix.org*.

PRESIDENT
Michael B. Jones,
mike@usenix.org

VICE PRESIDENT
Clem Cole,
clem@usenix.org

SECRETARY
Alva Couch,
alva@usenix.org

TREASURER
Theodore Ts'o,
ted@usenix.org

DIRECTORS
Matt Blaze,
matt@usenix.org

Rémy Evard,
remy@usenix.org

Niels Provos,
niels@usenix.org

Margo Seltzer,
margo@usenix.org

EXECUTIVE DIRECTOR
Ellie Young,
ellie@usenix.org

SAGE UPDATE

JANE-ELLEN LONG
jel@usenix.org
ALVA COUCH
alva@usenix.org

LISA '07 NEEDS YOU!

Has the time come for you to participate in LISA? Refereed papers, invited talks, Guru sessions, workshops, poster session, WiPs, BoFs, tutorials, and [your cool new idea for LISA goes here] all offer opportunities for you to join in, either as presenter or just suggesting your ideas. See the many ways you can contribute to LISA '07: 21st Large Installation System Administration Conference: www.usenix.org/events /lisa07/cfp/.

SAGE READING

You'll notice some new content on www.sage.org. Not only are the last of the LISA '06 White Papers online, but we have yet another SAGE booklet. *A System Engineer's Guide to Host Configuration and Maintenance Using Cfengine,* by Mark Burgess and Æleen Frisch, is now available in PDF format for all SAGE members to download. If you want a print copy, order online: www.sage.org/pubs/short_topics.html.

Some new books have been added to the Recommended Reading list. We have sections for books on general sysadmin, networks, programming, security and firewalls, soft skills, and the Web. Please check them out at www.sage.org/books/books.html. We crave suggestions for new books, notice of new editions, and areas of interest, as well as the occasional "Dude, this book has all the current value of last month's Caesar salad." Send your comments and suggestions to suggestions@sage.org.

2007 USENIX ANNUAL TECHNICAL CONFERENCE

Santa Clara, CA • June 17–22, 2007

USENIX

Don't miss the latest in groundbreaking research and cutting-edge practices in a wide variety of technologies and environments.

6 days of training by industry experts, including:

- Richard Bejtlich on TCP/IP Weapons School, Layers 2–3
- Peter Baer Galvin on Solaris 10 Security Features
- Æleen Frisch on Administering Linux in Production Environments
- Steve VanDevender on High-Capacity Email System Design
- And over 30 other full- and half-day tutorials

3-day technical program, including:

- The latest research in the Refereed Papers Track
- Keynote Address by Mendel Rosenblum of Stanford University, "The Impact of Virtualization on Computing Systems"
- Expert-led Invited Talks
- Guru Is In Sessions
- BoFs, a Poster Session, and more

Join us in Santa Clara, CA, June 17–22, for the 2007 USENIX Annual Technical Conference.

New in 2007: SANS
SANS Security Training

USENIX '07

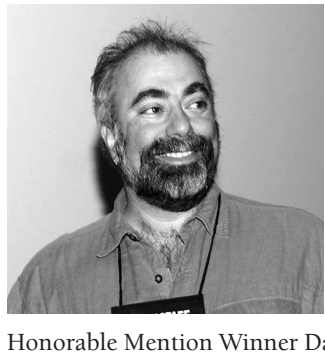
Register by June 1 and save! • www.usenix.org/usenix07/login

conference reports

THANKS TO OUR SUMMARIZERS

Leah Cardaci
Marc Chiarini
Chris Cooke
Beth-Lynn Eicher
Raj Kumar Gurung
Nathaniel Husted
Sumeet Jain
Kevin L. James
Robert Marmorstein
Sanjai Narain
Will Nowak
Alex Polvi
Josh Simon
Gautam Sinjaraju
Ning Wu

Best Paper Winner Melanie Rieback receiving her award from Program Chair William LeFebvre



Honorable Mention Winner Daniel Klein

LISA '06: 20th Large Installation System Administration Conference

Washington, D.C.
December 3–8, 2006

KEYNOTE ADDRESS

Hollywood's Secret War on Your NOC

Cory Doctorow, science fiction writer, co-editor of Boing Boing, and former Director of European Affairs for the EFF

Summarized by Alex Polvi (alex@polvi.net)

Cory Doctorow discussed many startling issues in his keynote about contemporary digital rights. The talk emphasized how the PC, the Internet, and crypto were once tools that helped us. Now, those tools are used to control us. The ex-EFF cohort and science fiction writer went on to discuss many issues, including the spyware in Amazon Unbox, the DVR disabling broadcast flag, and the crippling of TiVo.

The session brought out many idiosyncracies. For example, if you download music for free from Kazaa you will not get a rootkit; instead, you have to pay $15 for Sony to give you one. Or in the case of a Sony PSP, where the user hacker community is providing extra value to the device for free, Sony continues to obfuscate and defect the device.

The disheartening continued with consideration of End User License Agreements (EULAs). Today a EULA is often accepted without the user even doing anything. A concerned audience member was quick to ask what could be done as an individual and as a company. Cory responded that all EULAs must be addressed as a policy in the organization. Such policy will help raise awareness about the issue.

Cory concluded by reminding all sysadmins to make the right choice for digital freedom.

ELECTRONIC MAIL

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

■ ***Privilege Messaging: An Authorization Framework over Email Infrastructure***

Brent ByungHoon Kang, Gautam Singaraju, and Sumeet Jain, University of North Carolina at Charlotte

Brent Kang said it was his belief that email infrastructure is restricted by disk, CPU, memory, policy, and a large set of security concerns. He touched on how much time users spend sorting through email they do not want and identifying

falsified emails, such as phishing attacks. Dr Kang drew the audience's attention to the lack of an infrastructure to guarantee where a message is coming from and filtering based on that system. He drew an analogy between unauthorized email and unsigned software; allowing anyone to create a message that will appear in anyone's mailbox is akin to allowing anyone to write software and have it be trusted on any system.

The talk was not based on a new problem; rather, the idea of creating an email authorization framework has been around for some time. The solution that Dr Kang and his team created is called "Privilege Messaging" or "P-Messaging" for short. The system seems to be a cross between message authentication and a tagging structure. It layers on top of already existing email systems by adding a header to the email message. Utilizing these P-Messaging signatures, one can use them as one would tags in the "Web 2.0" sense, creating email filters and classification based on the sender's identity and "privilege-tag" information. The privilege-tag can be associated with a group, a department, or an individual, allowing for complex filtering schemes, which provide the ability to accept email on a white-list basis.

See http://isr.uncc.edu/pmessaging.

- ***Securing Electronic Mail on the National Research and Academic Network of Italy***

Roberto Cecchini, INFN, Florence; Fulvia Costa, INFN, Padua; Alberto D'Ambrosio, INFN, Turin; Domenico Diacono, INFN, Bari; Giacomo Fazio, INAF, Palermo; Antonio Forte, INFN, Rome; Matteo Genghini, IASF, Bologna; Michele Michelotto, INFN, Padua; Ombretta Pinazza, INFN, Bologna; Alfonso Sparano, University of Salerno

Alberto D'Ambrosio came to LISA on behalf of GARR, the National Research and Academic Network of Italy. As with other large sites, the GARR network faces the challenge of dealing with spam. Following a proposal in November of 2003, the SEC-MAIL team was formed to find a solution to dealing with the spike in spam delivery. The SEC-MAIL team started with a common base, Spam Assassin. The team worked to tune Spam Assassin's scoring, but it moved to work with a central Bayesian classifier. The SEC-MAIL team was seeing up to a 95% success rate with the Bayesian filtering. Mr. D'Ambrosio discussed SEC-MAIL's success by using many unique SpamAssassin plug-ins and a network of DCC servers in Italy.

Overall, the GARR SEC-MAIL team noted that the most effective tools for combating spam were Bayesian filtering, powerful DCC server networks, and greylisting. A combination of these tools, with other standard tools and best practices, led the GARR network to a steady decrease in delivered spam.

- ***A Forensic Analysis of a Distributed Two-Stage Web-Based Spam Attack***

Daniel V. Klein, LoneWolf Systems

Awarded Honorable Mention

Dan Klein presented a detailed analysis of a typical way a spammer can abuse a CGI-to-email gateway. The most interesting part of the talk involved the methods used to identify that such an attack was taking place. A series of RRDtool graphs illustrated trends that were not visible through on-demand statistics. A simple thing such as a sudden drop in spam, owing to an increase in overall volume and a high percentage of nonspam (compared to normal operation), can lead to the discovery of a small underlying issue. Klein's approach was twofold: to alert the system administration community to possible vulnerabilities that can lead to spam and to demonstrate how good reporting can lead to the discovery of such issues.

Dan emphasized that looking at your reporting technology is the most valuable way to detect these kinds of attacks. Simply looking at one log or graph would not have tipped him off as to what was happening or not happening; it was a combination of information fed to him through the comprehensive coverage of reporting. Because of this daily review he was able to notice a small (5000-message) spam attack, through his exploited Web form. There was a question from the audience of whether or not watching for short, sporadic hits to your Web form could help prevent this type of attack, to which he replied that it would not, on its own, indicate that an attack was happening. This is because short, sporadic hits to your Web site exhibit exactly the kind of behavior you expect. If you look at that data in conjunction with email volume, you can paint yourself a better picture of what is happening. Dan sent us off with an exhortation: We should check our scripts and our reporting tools.

INVITED TALK

- ***Teaching Problem Solving: You Can and You Should***

Elizabeth Zwicky, Acuitus

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

Zwicky gave an interesting and genuinely funny talk about why and how to teach problem-solving skills to system administrators. She began by speaking about the "noncontroversy" surrounding

the teaching of such skills. The general academic consensus is that excellent tutors are usually able to improve a student's skills by up to two standard deviations. This implies the possibility of pushing mediocre administrators into the highly employable "very good" category. There are of course a few barriers: People need to believe they can be taught this stuff; the system administration business selects for natural talents (i.e., not many admins get formal instruction); and, in general, although school systems know that problem solving can be taught, they don't tend to teach it. Once these hurdles are overcome, there are at least two good reasons to teach problem solving: First, people who cannot do it well tend to behave like bozos (the definition of which is left to the reader's fertile imagination!). Second, being able to solve problems naturally improves people's lives.

Throughout the talk, Zwicky offered examples of how a little bit of problem-solving skill can avoid (or could have avoided) disasters. For example, after three days of rain, a windowless third-floor machine room fills with muddy water via a blocked A/C drain. The first shift of A/C maintenance techs stem the flood by putting a plunger in the drain while a team of people work in a bucket line to empty the room out of a first-floor window. When the A/C repair shifts change at 8 a.m., a new A/C tech sees the water and the plunger and promptly removes the plunger with no forethought, creating a gigantic plume that causes damage to the ceiling and at least one machine. A bit more problem-solving skill would have led to the question, "Have I ever before seen a plunger jammed into an A/C drain, and if not, is there a good reason for it to be there?"

For most of the rest of her talk, Zwicky expounded on general problem-solving approaches as well as techniques for being a good tutor. In general, there are six steps (give or take) involved: identify the problem, analyze it, find solutions, choose a solution, implement it, and verify it.

Identifying a problem is sometimes more difficult than it sounds; one receives complaints such as "the elevators are slow" or "the Internet is broken." These are *symptoms* of a problem, not the problem itself. In the first case, once mechanical problems are ruled out, the key becomes recognizing that elevator riders have absolutely nothing to do while waiting. The *problem* is that riders are bored, and this has many good solutions. Likewise, in the second case, the symptom may be that the user cannot access cnn.com. As it turns out, the Web site itself is down, and nothing can be done from your end. No matter. The *problem* is that the users need their news (or, really, they're just bored), which again admits of many satisfactory solutions.

Analyzing a problem involves knowing what you are allowed or not allowed to do to investigate (do you have root?), and also how things look when they are working. Diagrams and guided questions are important in this phase. Don't be afraid to think "outside the computer." When you (and others) are satisfied with your analysis, it is helpful to come up with more than one solution and weigh them against each other and analyze their side-effects and long-term consequences. Again, don't hesitate to consider less costly solutions (by whatever measure) that fundamentally alter or eliminate altogether the process that led to the problem. Finally, verify your solution. Did the problem go away, and was it your considered solution that made that happen? Is the fix permanent? What would you have done differently if you knew then what you know now?

Now we know about general problem-solving approaches, so how do we teach these? Zwicky suggested several best practices: *Scaffolding* is doing the absolute minimum to allow somebody to reach a higher level than is possible without help; if done correctly, the student doesn't even notice the help. *Spotting* is the practice of being unobtrusive and letting the student make some mistakes, but catching any errors that would lead to disaster. Providing *conceptual focus* is another key; looking for repeated errors (having the wrong model or no model) that indicate misunderstanding and asking the student to verbally explain the concepts helps to keep this focus. *Praise and support* are also essential, but don't overdo it; people usually smell perfunctory back-patting. Finally, make sure to have learning environments that are as safe as possible; where practice is encouraged, mistakes go unpunished, and the teaching machines can be easily reset to a golden state. Zwicky offered the caveat that this is considerably harder to do in a workplace context.

Zwicky's references for tutoring and puzzle and problem solving can be found at the end of her online presentation at http://www.usenix.org/events/lisa06/tech/slides/zwicky.pdf.

INVITED TALK

Sysadmins, Network Managers, and Wiretap Law

Alex Muentz, Geek and Corporate Counsel, Cornerstone IT
Summarized by Nathaniel Husted (nhusted@iupui.edu)

Alex Muentz currently works as Project Manager for Onsite3. His talk covered how the various

wiretapping laws affect a system's administrator during his or her time on the job. The various laws include the 4th Amendment of the U.S. Constitution, the Wiretap and Electronic Communications Privacy Act, the Stored Communications Act, and various state laws. CALEA was also an important part of this talk, as were the pen register and trap and trace devices. He also made it be clearly known that this talk was not legal advice, that it was U.S. federal law and not state law, and that the precedents he would be discussing are not set in stone.

The first subject of the presentation was the 4th Amendment. Basically, the 4th Amendment protects against unreasonable search and seizure. This also does not affect any private actors, only public actors. In the private sector, certain states have "intrusion into seclusion" laws that can be the basis for a civil lawsuit. The current 4th Amendment view was decided in the Katz v. United States court case in 1967. This case stated that there is a privacy right. A person has the right to privacy when two conditions are met. The first condition is that society at large decides if the person has a right to privacy during an activity. The second condition is that the person thinks he or she is being private during the activity. In terms of communications, this means that a person is protected in whatever communications that he or she receives, but not in what he or she transmits. Also, information given to third parties for a delivery is not protected except in certain situations.

The second subject discussed was the Wiretap and Electronic Communications Privacy Act. This was originally enacted as Title 3 of the Omnibus Crime Control Act of 1968, then updated in 1986, and finally in 2001 by the Patriot Act. Alex also mentioned that FISA does not modify this, but he did state that nothing should interfere with the president's right to gather intelligence about foreign powers. The Wiretap and Electronic Communications Privacy Act states that interception is the acquisition of the contents of oral communications through the use of any device. The term device is not specific and can include sniffer software, laptops, or other mechanisms. Interception only happens if the information is read on the wire. Penalties include five years in prison and fines greater then $10,000 per incident. One exception to this law is that recipients may intercept their own messages. Another exception is that server provider agents, employees, and contractors may intercept communications to protect the rights or facilities of the service providers, to comply with a court order, to troubleshoot customer problems, or with the permission of the user.

The third subject discussed was the Stored Communications Act. According to the Stored Communications Act, accessing stored communications without permission or exceeding the granted permissions results in criminal penalties. This also includes someone who alters or prevents authorized access to the stored information. If the law is violated for profit, the penalties include five years in prison for the first offense and ten years for each subsequent offense. If the law is violated without a profit motive, then a person can receive up to five years in prison. Fines can also be issued by the courts. There are various exceptions, however. One exception is that the owner of a service, for any reason, can view stored material. Another exception is that providers may divulge content to the information's recipient or forward it to a third party if certain conditions are met. These conditions include a court order, inadvertent discovery of criminal evidence, or a reasonable belief that death or physical harm may fall on someone mentioned in the letters. The term *provider* is not clearly defined by the law. A basic definition is a maintainer or owner of some system that transmits electronic communications.

The fourth subject discussed was pen register devices, trap and trace devices, and customer records. Pen register devices traditionally are devices that list all phone numbers dialed along with the duration of calls from a single phone. Trap and trace devices are traditionally devices that list all the numbers that have dialed a single phone number as well as the duration of those calls. Customer records consist of names, dates, times, payment methods, and addresses (real or IP). What constitutes customer records is defined vaguely except for phone records. The only restrictions on getting pen register or trap and trace warrants are that law enforcement must show the courts that there is a legitimate need. Providers must only use informed consent, or the usage must be for billing, maintenance, testing, protection of services, or compliance with an issued wiretap order.

CALEA comes into play when discussing network design. According to CALEA, any network must allow the federal government to tap in. Alex said that the Department of Justice has stated that it must be able to do these tapes remotely without the NOC having knowledge and without the help of the network administrator.

Other interesting cases that were talked about include Steve Jackson Games v. U.S. Secret Service (1995), Garrity v. John Hancock, Muick v. Glenayre, Konop v. Hawaiian Air, IAC v. Citrin, and Councilmen v. United States. The most notable is this last case, in which the judges decided that sniffing is more than what is on the wire, but they did not define what sniffing is. Alex stated that, just to be safe, businesses should operate under the wiretapping law when reading information in their own storage. This only affects individuals in the First Federal District (New England and Puerto Rico, minus Vermont and Connecticut).

Alex said that the best way to protect yourselves is to make sure you have the user's consent in writing. He also said that it would be best to have a sniffer policy and that said policy should be clearly laid out in the employee handbook. Also, businesses need to make sure they are aware of any application that is sniffing networking traffic.

A number of questions were asked during this session, and most revolved around what is acceptable under Councilman v. United States. Other questioners also asked about CALEA. Alex continued to stress his point that businesses need to ensure that the customers are notified about what is going on. One thing Alex mentioned about CALEA is that if you do not own the keys to any encrypted data that needs to be accessed, you are not responsible for providing those keys to the authorities.

BOUNDARIES

Summarized by Ning Wu (ningwu@cs.tufts.edu)

Firewall Analysis with Policy-based Host Classification

Robert Marmorstein and Phil Kearns, The College of William and Mary

Firewall policies are susceptible to many configuration errors. The difficulty of analyzing these policies prevents their rigorous analysis. Active testing of policies may not catch all possible errors; passive testing produces output that is too unstructured to be useful.

One analysis method that avoids these problems is to classify hosts into groups by deriving equivalence classes from firewall policy. One first converts the policy into a multiway decision diagram: a directed acyclic graph in which each path through the graph represents a firewall rule. From this graph, equivalence classes can be computed that represent mail server, workstations, Web servers, etc.

Reviewing these classes for possible configuration errors allows the user to catch typos, shadowed rules, out-of-order rules, and other configuration errors. This analysis method is available via the tool ITVal at http://itval.sourceforge.net.

Secure Mobile Code Execution Service

Lap-chung Lam, Yang Yu, and Tzi-cker Chiueh, Rether Networks, Inc.

Yang Yu presented SEES (Secure Email Execution Service), a commercial system that secures the execution of mobile code that arrives at a host as an email attachment or as a downloaded Web document. Because signature scanning cannot detect any new malicious code, to protect the user the mobile code is moved to an isolated playground machine containing no valuable data and executed there.

The document containing mobile code is intercepted from Web browsers and email clients. The local version of the document is saved with a special mark, and local read operations are intercepted. After the document is sent to the playground server, a terminal is started by using remote desktop activeX control. The GDI events and bitmaps are sent back to the user's computer. The accounts on the playground machine are autogenerated, and the profiles are reset. There are also firewalls isolating the playground server.

Yang also discussed the limitations: Currently, mobile codes are intercepted from email and Web browsers only; terminal servers may be not the best choice (because of scalability and license issues).

FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs

Adam Slagell, Kiran Lakkaraju, and Katherine Luo, NCSA, University of Illinois at Urbana-Champaign

Adam Slagell presented FLAIM, a multi-level anonymization framework for computer and network logs. There is an urgent demand from the research and education community to obtain real-world logs of various kinds of system events, but data owners are concerned about exposing sensitive information contained in the logs. Anonymization, defined as the process of removing information about personal and organizational identity from the logs, is absolutely necessary to protect the data owners who are willing to share.

FLAIM attempts to bridge this gap by anonymizing the data but keeping enough information in the data to make it useful to analysts. A key part of this is to define a diverse set of anonymization

algorithms and to separate format parsing modules from the core anonymization module, so that one can make policy decisions about how to anonymize without having to understand how data is parsed. FLAIM software is available at http://flaim.ncsa.uiuc.edu.

INVITED TALK

■ ***Site Reliability at Google/My First Year at Google***

Tom Limoncelli, Google

Summarized by Alex Polvi (alex@polvi.net)

Tom Limoncelli gave an overview of Google's production infrastructure. Tom covered how Google does upgrades, gave a simple view of the network, and showed some of the impressive benefits of the distributed system. Upgrades are done in a multistep process. Every feature in a Google application has a corresponding flag. When a new flag is ready to be turned on, the upgraded application is put into production, with the feature flag off. The site is then tested for regressions. After the upgraded site is proven to be equivalent to the original, the feature is turned on. Emphasizing the benefits of this approach, Tom noted, "once you have good failure management, upgrades are free."

Tom also brought light to the Google WAN. Google will distribute applications across many different data centers, but not all of them. For that reason there is often a choice between sending user requests to the closest data center and using the Google backbone or simply having the user send all requests directly to the application-hosting data center. Tom also discussed some of the stats associated with distributing all data via the Google File System. For example, in 2004 their infrastructure was able to grep 1 TB of data in 100 seconds using 1800 machines. Concluding, Tom mentioned that lots of machines, lots of data, and lots of services make for a sysadmin's playpen.

INVITED TALK

■ ***Leveraging the IT Community***

Patrick McGovern, VP Community and Services, Splunk

Summarized by Gautam Singaraju (gsingara@uncc.edu)

Patrick McGovern has been the director of the successful Souceforge.net Web site for five years. McGovern introduced the idea of building an IT troubleshooting community Web site. The vision behind Splunk is to develop a centralized repository for logs and IT data information. The goals as described were to provide logs and IT data for availability, security, and compliance. With the ability to share the logs, Splunk builds a community where users share and solve each other's problems. Spunk would be a centralized repository for information about all IT troubleshooting information. McGovern demonstrated that combining the viral nature of Wikipedia with the technical knowledge base from SourceForge.net would bring forth a community-based IT troubleshooting Web site for system administrators.

Started in 2001, Wikipedia, ranked as the 12th most popular Web site according to Alexia, was based on Nupedia. Wikipedia decentralized the content by using a simple wiki for uploading the contents. Following its tremendous growth, Wikipedia today holds about 1.5 million articles. The growth was attributed to the fact that Wikipedia simplified the content review process by allowing the community to collaborate.

SourceForge.net has about one-million registered users, with about one hundred thousand projects. Ranked 81st by Alexia, it has the world's largest user base and code repository. SourceForge has provided support for community-based open source development by allowing team building, collaboration, and fast development cycles. Each project can build a micro-community with a bug-tracker, ticket tracker, feature requests, mailing list, cvs, file release, and mirrors. Some of the successful projects developed at SourceForge are SugarCRM, Mailman, JBoss, SquirrelMail, and Gaim.

McGovern introduced Splunk as a technology and not as a Web site. Data centers generate a lot of logs; as a software stack, services on the machines at a data center have considerable interdependency. It is important to locate and fix the source of a problem as soon as possible. Finding the source of a problem requires many users to come together to identify the problem in the system, with 30–70% of the time spent looking through the logs. The problem in the Linux distribution is not obvious, as it installs and runs about 50 services at startup. McGovern pointed out that it is difficult to write tools to parse the data, as there are multiple log formats that change for each release of the software. The events need to index the events in real-time.

Splunk is an Ajax-based search engine for machine data. Splunk's powerful algorithm dynamically indexes data and discovers relationships among them to troubleshoot data across multiple applications, systems, and technologies. Splunk also provides all facilities for collaborative devel-

opment, just as in SourceForge. McGovern introduced Splunkbase so that the community can edit based on a wiki. Splunkbase provides an interface where people will enrich the data by introducing tagged words.

To a question about anonymization of the logs, McGovern stated that Splunk provided tools to anonymize the IP address. McGovern suggested that the logs to be put up on Splunkbase should be small, so that the community could go through them to help in troubleshooting the problem.

SECURITY

Summarized by Robert Marmorstein (rmmarm@cs.wm.edu)

Centralized Security Policy Support for Virtual Machine

Nguyen Anh Quynh, Ruo Ando, and Yoshiyasu Takefuji, Keio University

Nguyen Anh Quynh presented an architecture for implementing Mandatory Access Control (MAC) on a collection of virtual machines. This is work his team has done at Keio University in Japan. After a quick explanation of MAC and a summary of how it differs from Discretionary Access Control, Nguyen pointed out that virtual machines pose unique problems for mandatory access security policies and described a framework for addressing these problems.

Nguyen argued that implementing MAC on a collection of virtual machines (VMs) is difficult with a traditional architecture, because the security policy must be managed separately on each VM. He also discussed the challenges of collecting and correlating security logging data in a VM environment.

To address these problems, his team created VMAC, which uses shared memory to centralize logging and policy enforcement. Their approach uses a client/server model in which one VM serves as a policy manager and the other VMs retrieve the MAC policy from the server, which also coordinates security logging data. The VMAC architecture is designed to support any kind of VM and any MAC scheme. To prove the concept, they implemented VMAC using Xen with 3 MAC schemes: AppArmor, LIDS, and Trustees. In the future, they want to expand their implementation to allow other kinds of VMs and to support a broader variety of MAC policy models such as SELinux.

A Platform for RFID Security and Privacy Administration

Melanie R. Rieback, Vrije Universiteit Amsterdam; Georgi N. Gaydadjiev, Delft University of Technology; Bruno Crispo, Rutger F.H. Hofman, and Andrew S. Tanenbaum, Vrije Universiteit Amsterdam

Awarded Best Paper

The best paper of the LISA conference lived up to its high billing. According to Melanie, the proliferation of RFIDs poses several significant and dangerous privacy and security issues, ranging from identity theft to illicit tracking of passports. To address these concerns, her team has developed RFID Guardian, an embedded device for managing RFID tags and readers.

Melanie was very up-front that the Guardian could be employed for nefarious as well as benevolent uses. In addition to allowing users to identify, jam, and spoof RFID tags, the Guardian can be used to implement RFID auditing, authentication, key management, and access control. This extensive range of powerful features can be used to protect your privacy, but it could also be employed by malicious users to steal private information, launch denial of service attacks by jamming valid RFID transmissions, or circumvent anti-shoplifting measures. However, Guardian can also be used to protect your personal privacy and improve the security of RFID transactions. These benefits can be extended by employing Guardian-aware RFID readers.

The device is portable (battery-powered) and has much greater range than RFID tags. It actively transmits on the side-bands, rather than relying on power received on the carrier frequency. By generating carefully timed collisions, the Guardian provides selective RFID jamming, leaving some RFID tags accessible while blocking others.

Future versions of the Guardian will reduce the cost of the components, improve the range of the device, and potentially include a Bluetooth interface for communicating with cell phones. The question and answer period led to some lively discussion. When asked by Brian Trammel whether multiple Guardian units can interfere with each other, Melanie admitted that no such testing has been tried, but such tests are planned once the unit goes into mass production. With careful design, they may be able to make Guardians play nicely with each other. Cory Doctorow pointed out that perhaps a better solution to the problem is to get rid of RFIDs altogether. Melanie replied

that there doesn't seem to be a way to stop the proliferation of RFIDs. She argued that making Guardian widely available will make people more aware of the privacy issues surrounding the use of RFIDs.

Guardian supports both of the ISO standards for RFID. More information on the Guardian is available at http://www.rfidguardian.org.

INVITED TALK

■ ***Open Source Software and Its Role in Space Exploration***

DJ Byrne, Software Engineer, Jet Propulsion Laboratory
Summarized by Alex Polvi (alex@polvi.net)

DJ Bryne talked about open source at NASA's Jet Propulsion Lab (JPL) at Caltech. The JPL is dedicated to expanding knowledge, observing the universe, and analyzing data. DJ discussed many reasons why open source software is fit for such purposes. First, community-based bug fixes and feature additions are generally much better than those of proprietary vendors. Also, there is much higher confidence in the future of a particular software if it is community-supported. Community-based open source software is used at the JPL in its OS, software management, communications, visualization, compilers, databases, and various other places.

The JPL also releases its own open source software. The CLARAty project provides a framework for researching autonomous mobile robots. Having this software released under an open source license has helped the JPL collaborate with other researchers.

INVITED TALK

■ ***Virtualization: The Good, the Bad, the Ugly***

Mark Baum, Splunk
Summarized by Kevin L. James (kevljame@cs.iupui.edu)

This talk was given by Mark Baum, CEO of Splunk, a startup company that produces a system administrator's "search engine for IT data." He was joined by Mike Beck of the Emerging Technologies Group, the 2006 System Administrator of the Year, and Mark Cohen, Senior Technical Support Representative at Splunk.

Baum began by giving an overall view of the good, the bad, and the ugly of the current crop of virtualization technologies. The good news is that virtualization greatly simplifies the administration of many diverse systems. Existing resources are better utilized, allowing for improved and more cost-effective scalability. The bad news, according to Baum, is that resources also become more complicated and therefore harder to manage, making virtualized systems more prone to failure and performance issues. These complications are compounded when troubleshooting must take place, as pinpointing the source of failure becomes blurred: Whom do you contact, the software vendor or virtualization technical support/open source community, when your application fails? Even before virtualized resources are in place, the decision of which type and method of virtualization in which to invest can cause headaches.

Next Baum discussed several types of virtualization. One of the more recognizable types, server virtualization, aims at masking resources of servers (physical machines, processors, and host OSes) and configuration management from users. There are two methods being promoted: hosted and hypervisor. The hosted method runs as an application, making it easy to set up, while locking it to a specific platform. According to Baum, this leads to better performance for the virtualized resources. In contrast, the hypervisor model provides a leaner kernel capable of running on different architectures. Baum has found that there is currently less support for this method and performance is actually less than that of hosted mode, but of the two, hypervisor is growing faster.

Turning to Mark Cohen and Mike Beck, he asked them their opinions of server virtualization. In Beck's company, the primary benefit their exploration has turned up is that many solutions have built-in, hands-off failover support for virtual resources. According to Cohen, Splunk uses server virtualization in development, product testing, and support, as it saves money over implementing every platform supported by their product in hardware. When called upon to troubleshoot their product, problems are solved more quickly because the many possible user platforms can be emulated on a single system.

Baum's poll of the audience revealed that about one-third have implemented server virtualization in their work.

The next virtualization type presented was storage virtualization, which allows the transparent pooling of storage resources. The immediate downside to this technology is the expense of current offerings and the noticeable lack of open source implementations in this area. Mark Cohen has found

that virtualized storage is very useful when implementing business architecture; he uses it to prevent their SANs from being overrun by greedy applications. The last type discussed was network virtualization, a technology that Baum remembers was used heavily by his previous employer, Yahoo!. When dedicated routes, channeling, and finer grained control are desired, implementing them virtually makes these easier to control. According to Baum, this promising technology is still too new.

Next the talk shifted into a description of how virtualization is used at Splunk. Virtualization solutions are implemented for all supported platforms, simplifying development, quality assurance, and customer service tasks while providing great savings from having to buy equivalent hardware. Current solutions used at Splunk run on Linux, Mac OS X, and Windows. According to Baum, his people can run 16 to 20 operating systems on a single dual-core laptop. Unfortunately, there are bad and even ugly sides to all of this great promise. One problem is that although virtualized implementations are able to be scaled easily, this can hide too much information. Users have no knowledge of where their "physical sites" are located or what resources are available to them. Because the physical infrastructure behind these virtual servers is obscured, it is difficult to determine the cause of slowing or failing systems. The increased scale can also make patch management a nightmare because of the lack of tools written to work in a virtualized environment. This proves especially troublesome at Splunk, as they must constantly update their virtual systems to keep up with the machines of their users.

Another task made difficult by virtualization is backup operations. When backing up a system running several virtual ones, it is difficult to restore a single virtual platform. According to Cohen, because of how backup software often runs, it cannot be used to back up individual files within a virtualized system; instead, the entire virtual disk image must be archived. Cohen also finds troubleshooting to be difficult, because virtualization can have unwanted effects on platforms. One problem Splunk found was that time becomes skewed within virtual environments. Because the company's products deal with various system logs and files, using its software to pinpoint problems can become impossible in a virtualization setup, since time is inconsistent across the virtual platforms.

Baum's final observation is that although virtualization has been great in Splunk's development, testing, and QA shops, it is not so great for production environments. The available management tools are not robust, and automated provisioning of resources to virtual systems under heavy load is nonexistent. Therefore virtual servers often experience high utilization rates, says Baum, even when the underlying hardware is not being taxed.

THEORY

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

Specification-Enhanced Policies for Automated Management of Changes in IT Systems

Chetan Shankar, University of Illinois at Urbana-Champaign; Vanish Talwar, Subu Iyer, Yuan Chen, and Dejan Milojicić, Hewlett-Packard Laboratories; Roy Campbell, University of Illinois at Urbana-Champaign

Shankar began his talk by describing the highly heterogeneous device makeup of many modern networks and computation grids. Most of these systems are subject to frequent changes that are managed by somewhat fragile automated scripts. This management approach does not scale well and is prone to many kinds of errors, not the least of which is the untested effects of different orders of script execution. Policy-based management, in contrast, is much more scalable and tolerant of mistakes. One traditional method of implementing such a management solution is via an Event-Condition-Action (ECA) framework. In this approach, events serve as triggers that, upon certain prespecified conditions being met, initiate some kind of action (which could be corrective or just a normal operation). An example of an ECA rule is "When checkpoint store is full (*event*), if backup store is running (*condition*), assign backup store as new checkpoint store (*action*)." The drawback to ECA is that when system changes occur, rules can trigger simultaneously, resulting in a nondeterministic ordering of actions that does not achieve the desired end state. Shankar gives a simple but very convincing example of this uncertainty when ECA is used to manage a computation cluster that has failing aggregator nodes.

To address the weaknesses of traditional ECA, Shankar proposes a specification-enhanced rule framework called ECPAP (Event-Condition-Precondition-Action-Postcondition). A *precondition* is a first-order expression representing some desired partial state of the system before any action is taken. Similarly, a *postcondition* represents a desired partial state after the action is completed. While events and conditions are normally specified by the system administrator, preconditions and postconditions are designed by the *action de-*

veloper, who may be an entirely different entity. An analogy can be found in management scripts (pre/action/post) distributed by a vendor; the sysadmin may determine why (event) and when (condition) to use them. The ECPAP framework applies algorithms that can identify dependencies between conditions and build Boolean Interpreted Petri Net (BIPN) workflows. BIPNs are very useful in modeling and reasoning about concurrent action execution. A workflow is constructed by analyzing each pair of actions to determine whether one enables the other (*enablement analysis*). Following this, an appropriate ordering of rules can be enforced by a workflow execution engine, subject to one of three different enforcement semantics: *random*, which executes rule actions in random order, is the type of workflow generated in traditional ECA; *Maximum Rule* semantics guarantees that the management system enforces rules in an order that ensures as many rules are successfully enforced as possible, provided no other errors cause enforcement to fail; *All-or-None* specifies that rule actions must be executed only if all actions can eventually execute. This is accomplished by reachability analysis on the BIPN, and it provides the strongest guarantee.

Shankar went on to briefly discuss the algorithmic complexity of ECPAP's various analyses and showed some of the research team's successes in applying the framework to managing changes to HP OpenView and Ganglia performance-monitoring infrastructures.

■ ***Experience Implementing an IP Address Closure***

Ning Wu and Alva Couch, Tufts University

Ning Wu presented a prototype "IP address closure" that provides integrated DNS and DHCP services. A *closure* is a self-managing "black box" that implements and protects the functionality of one part of IT infrastructure while making its needs known to other closures. In this case, Wu and Couch propose placing such a closure on a USB device that can be used to bootstrap a network. The procedure is fairly simple when compared to other methods for maintaining IP address assignments in larger networks: The devices communicate using a simple pull-only gossiping protocol in a peer-to-peer network and are configured via a process of *seeding*. Each device is initialized by physically plugging it into the same subnet as an already seeded device. The new device discovers the existing device, clones its configuration, and receives an idea of network topology, policy, and locations of peers. It is then moved to its final location, after which it can serve to seed other closures. This approach has many advantages, including the ability to provide automatic failover and information backups, as well as to enable quick policy change propagation.

In addition to the technical details of the research, Wu also talks about the system administrator's role when interfacing with self-managing systems. The only input to the closure is a policy file (in two parts, low-level and high-level) describing the desired relationships among IP numbers, network names, MAC addresses, and subnets. This would ideally be determined (automagically in part) by a lower-level routing closure, but such a closure has not yet been built. Still, the sysadmin is relieved of managing superfluous aspects and "incidental complexity" that has no behavioral impact (e.g., having to ensure agreement between DHCP servers on the location of routing gateways). This approach also avoids common human errors during bootstrapping by, for example, automatically replicating configurations. It's important to note that human sysadmins do not become obsolete in all this. Instead, their role moves up a level, to managing policy, ensuring that the physical architecture matches policies implemented by the closures, and intervening when certain closures discover mismatches between, for example, physical or virtual subnets and desired operating characteristics.

During the Q&A session, Cat Okita asked whether the approach was just making things more complicated. Wu answered that a lot of complexity is internalized, freeing the sysadmins to fulfill their new role. Another question concerned ideas for implementing a closure for the presumed routing architecture. The answer was taken offline. Marc Chiarini commented on the necessity of a reporting facility being built into the closure magic so that system administrators start to trust the technology.

■ ***Modeling Next Generation Configuration Management Tools***

Mark Burgess, Oslo University College; Alva Couch, Tufts University

By his own admission, Dr. Couch commits sacrilege in this paper by presenting a model of configuration management without tying it to any practical tools. He tells us about aspects, closures, and promises as three essential tool-invariant pieces of the same puzzle: Paul Anderson's *aspects* model dependencies between entities; Couch's *closures* model behaviors of entities; and Mark Burgess's *promises* model interactions. Couch attempts to enlighten the audience as to how this "grand unified theory" can be applied to practical concerns

and how future tools may be built with this in mind. He wants to dispel the myth that the tools and technologies we use define the cost of config management. In fact, most IT managers know (even if they won't admit it) that it is *the way we think about the problem* that defines the cost. We currently have two ways of thinking about config management: the prescriptive approach (BCFG2, Puppet, LCFG) and the convergent approach (cfengine). The former applies a "rule with an iron fist" methodology that makes the problem too big. The latter exhibits a more laissez-faire attitude that makes everything too small. Neither of these is sufficient to tackle the entire problem, and Couch claims we are reaching the limits of our current conceptualization.

Couch goes on to explain the three pieces of the puzzle in detail: An aspect is a set of configuration parameters whose values are interdependent (e.g., all the locations in which the hostname of a machine appears). Many people understand aspects in an intrinsic way as the coordination of parameters needed to achieve a certain effect (e.g., have a running Web server). Aspects are important because they're a tool-independent way of describing interaction and complexity and they allow some approximation to the difficulty of a management task. If one can partition parameter sets into subsets in such a way that constraints between those subsets are minimal, one can then use aspects to begin to talk about desired behavior. Closures, in a mathematical sense, are deterministic mappings between configuration and behavior. One doesn't so much build closures as *find* them. For example, we can *discover* a Web service closure by identifying and controlling all aspects that determine Web service behavior. Current prescriptive tools actually forcefully create aspects, but they don't find closures, because they don't comprehensively encapsulate desired behavior. Convergent tools also end up only managing configuration as opposed to behavior; we need some way of composing closures to describe larger and larger spheres of behavior, and for that we require promises. Promise theory can be used to describe communication between closures when they are viewed as autonomous subsystems that minimally overlap.

The most important point is that the new theory provides an efficient way to talk about system behavior within the configuration management community and also to build next-generation tools to validate behavioral models (and therefore configuration) by using closures and promises.

INVITED TALK

Everything You Know About Monitoring Is Wrong

Mazda A. Marvasti, Integrien Corporation

Summarized by Robert Marmorstein (rmmarm@cs.wm.edu)

According to Mazda, recent trends in architecture, such as the proliferation of virtual machines, have provided administrators with extremely large quantities of logging and other data. Dealing with such massive amounts of data impairs the effectiveness of traditional strategies for monitoring systems and networks. The talk discussed ways to address this problem by adopting new paradigms for monitoring. Whereas some parts of the talk seemed more appropriate to a vendor BOF than to an invited talk, much of the presentation focused on new ways to think about data and monitoring.

After discussing reasons why many admins feel they need to collect more and more data, the speaker suggested that collecting "tons of data" is the wrong goal. Instead, the focus should be on getting data that has a direct business impact.

The distinction between data (something measurable) and information (something useful) formed the basis of much of the talk. Mazda argued that collecting more data is not always helpful in identifying potential failures. Often, an overwhelming amount of data makes it difficult to distinguish important events from trivial ones.

Much of the talk focused on a white paper the speaker had written on a simulated IT environment. From the data he collected, he concluded that using 40% of the available metrics is optimal. In his analysis, he found that using this proportion of the available metrics eliminated 98% of potential problems from consideration.

The remainder of the talk focused on the benefits of integrity management. Mazda advocated the use of self-learning for determining the "normal state" of the system and for predicting faults. Although this has the drawback of requiring some time for training the system, it provides better post-mortem analysis after the first occurrence of a failure and helps reduce the duration of subsequent faults by identifying problems earlier than other techniques.

Mazda also argued that using dynamic thresholds rather than static thresholds provides more accurate measurements when discovering deviations from "normal" and allows for earlier prediction of faults. He argued that monitoring solutions must deal with change and that static thresholds are in-

adequate for this task. He also argued for increased sophistication in statistical techniques.

During the Q&A period, Mazda described the learning period of his tool. It took four weeks to achieve daily resolution of events. It took nine weeks to achieve hourly resolution. He admitted that gradual changes can sneak past the self-learning paradigm and advocated using static threshold conditions and SLAs to detect these kinds of faults.

INVITED TALK

Is Entropy Winning? Drowning in the Data Tsunami

Lee Damon, Sr. Computing Specialist, University of Washington; Evan Marcus, CTO and Founder, Aardvark Technologies, Ltd

Summarized by Sumeet Jain (jain.sumeet@yahoo.com)

According to the speakers, we're drowning under a wave of data and are oblivious to it. As data space expands we will start losing track of, and thus losing, our data. Archival backups add complexity to this already confusing situation. Then we toss in security and availability issues for some spice. Where is this going, and how can we handle it in the face of millions of gigabytes of "old cruft"?

The speakers explained existing problems in data archives and then discussed some ways of solving these problems:

- Disk is cheap but the information is expensive.
- Long-term storage is easy but retrieval is difficult.
- Time is more expensive.

Many threats exist even if we store every bit of data: The storage media can wear out; media readers may not be available or can't decrypt the data; and even if everything is present there remain difficulties in finding a small piece of information in such an ocean of data.

In ancient times data was stored on media such as papyrus or rocks, which are still readable but storing on these media was hard and expensive. Then there was an era of handmade books. Entering data in these formats was easy but it involved a high cost of ownership, and few people could read or write. Johannes Gutenberg's invention of printing made it easy to publish data in the form of books but the cost of ownership was very high. In the initial stage of evolution of computers, data was stored on punch cards, which were very bulky and had limited memory. Later on, data was stored on magnetic media, which could store whole roomsful of punch cards on a few tapes. Associated with magnetic media were some new problems such as unlabeled tapes and long-term storage. As the size of storage started increasing people started keeping a lot of data backups. Today we can get 4.5 TB+ for only $7000, but how are we going to back this much data up? And long-term storage is still a big problem. The SSLI Lab has grown from less then 1 TB to over 13 TB of backed-up storage in 5 years, along with 100s of GBs of scratch space on every disk. Most data is transitory and in limbo space.

Archives have three basic functions: ingestion, preservation, and access. With ingestion many questions need to be answered: Is this the right archive for the record? Are there duplicate records? Do records need to be stored on-site or remotely? Data preservation relates to the current condition of records, environmental needs of records, ensuring that what we store is what we retrieve, and security controls for record access. Accessibility of data relates to access policies, arrangement of records, and searching for and locating the desired piece of information.

Can we say that librarians are the best people to handle our data archives? They have thousands of years of experience in data collection and cataloging. They deal with finished goods more often than us. But they have their own problems of data finding and indexing.

Several solutions are available for libraries:

http://digital.lib.washington.edu/staff.html
http://www.lockss.org/lockss/Home
http://www.contentdm.com/

Several solutions are also available for indexing, change-tracking systems, and document management systems (e.g., Google).

Lee and Evan stated that people view their short-term "being busy" state as more important then the long-term ability to recover, restore, search, and identify data. People should decide what data is important to them and how long they should keep such data. One should keep data for six months; if you don't use it in that time, throw it away.

ANALYSIS

Summarized by Ning Wu (ningwu@cs.tufts.edu)

- ***Windows XP Kernel Crash Analysis***

  Archana Ganapathi, Viji Ganapathi, and David Patterson, University of California, Berkeley

  Archana Ganapathi presented the analysis of Windows XP kernel crash data collected from volunteers who contribute to the Berkeley Open Infrastructure for Network Computing (BOINC) project. During a year, the authors overcame the challenges of collecting user data and collected over 2500 crashes. The collected data was carefully analyzed to obtain temporal patterns in the crash history. One of the goals was to determine which organizations are responsible for causing crashes.

  Analysis shows that seven organizations caused nearly 75% of the crashes in the data set. The data is also categorized based on fault type, image name, etc. Archana also reminded the audience that this result is only derived from subscribed hosts, and no information about the installed software and their frequency of usage is available. However, it is clear from the data that Microsoft is not solely responsible for crashes.

  This paper also introduces a customer-centric kernel crash analysis framework that will help users evaluate their current practice (e.g., compared to the average crash rate) and provide useful information on how to improve.

- ***SUEZ: A Distributed Safe Execution Environment for System Administration Trials***

  Doo San Sim and V. N. Venkatakrishnan, University of Illinois, Chicago

  V. N. Venkatakrishnan presented SUEZ, a distributed safe execution environment that allows an administrator to "try" new changes before they are "committed" or "aborted." Currently, the tests for new changes are tried either in the real environment or in a testbed that is constructed to be similar to the real environment. However, it is risky to try changes in the real environment and testbeds often do not reflect the real environment. Another approach is to change the operating system itself to allow testing and commitment of changes. The authors propose a distributed safe execution environment (SEE) that implements one-way isolation between the SEE and the host OS. The processes in the SEE can access the host OS; but the host OS cannot access the processes in the SEE.

  A SUEZ environment consists of host monitors and a network redirector. Host monitors use system call interposition to provide host-level isolation, and network redirectors provide network-level isolation through static or dynamic redirection of network services. The performance impact of SUEZ is carefully analyzed with several applications to show that the performance is acceptable.

- ***WinResMon: A Tool for Discovering Software Dependencies, Configuration, and Requirements in Microsoft Windows***

  Rajiv Ramnath, National University of Singapore; Sufatrio, Temasek Laboratories, National University of Singapore; Roland H. C. Yap and Wu Yongzheng, National University of Singapore

  Often system administrators feel that we need more history or dependency information when making decisions regarding management of the Microsoft Windows platform. For example, "Can I safely remove this DLL file?" or "What programs are using this registration key?" Roland Yap introduced WinResMon, a tool that can help administrators answer these questions. WinResMon does this by tracing the history of access of resources through intercepting system calls. The trace information is then generated and stored in a database.

  Accesses to resources including file, registry, network, and synchronization objects are recorded in the log database. I/O operations are not logged because of privacy concerns. The logs can be queried with a customized query analyzer by using a customized query API à la SQL. The log database can also be maintained through log APIs. The overhead of WinResMon was analyzed using micro-benchmarks and the results show that it is comparable to that of other tools. The volume of data is also acceptable; Roland described that, in his environment, the analysis rate of raw data before compression could be as high as 18 MB per hour.

INVITED TALK

- ***Perfect Data in an Imperfect World***

  Daniel V. Klein, Consultant

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

  Dan Klein related the current increase in the collection and preservation of data and the implications of this change. Klein first looked at the impact of long-term data retention. He illustrated why this is a problem using the example of trying to explain his college years to his hypothetical children. One solution to this problem is personal digital rights management, which would allow control of how long the data could be used, who could use it, and how it could be used. He went

on to provide other examples of how retention and dissemination of information about someone's past could be damaging.

Klein next discussed how to handle the current state of data exposure. On an individual level, you can protect your personal information by not doing that which is illegal, not foolishly publishing incriminating information, not publishing anything you'll regret later, caring what others think, and not sabotaging the collection of data. On a societal level, you can isolate your information, legislate for privacy protection, mitigate the impact of data retention, and understand how privacy can be threatened.

In addition to the long-term retention of data, another problem is the widespread collection of data. Data is collected in credit card logs, RFIDs, and loyalty programs. In addition, such information can potentially be misused. This is compounded by the fact that people are often willing to trade privacy for convenience. Many collection practices have a potential good use, but they also can be used to violate privacy. This brings up the question of whether data must always be stored completely to serve the purpose for which it is preserved. In some cases, it may be desirable to store data completely in the long term, in some cases it is the trend in the data that is important, and in some cases it may desirable to destroy the data after a given time. However, it is not always obvious what complete, long-term data might be useful to others.

Besides the abundance of data and the lack of control that individuals have over their data, there are also problems associated with everyday attitudes toward digital information. There is a tendency to blindly believe in data, without considering the fact that it can be incorrect. There is also a tendency to abandon established social niceties when dealing with electronic information about friends, employees, and acquaintances.

People have no idea what data about them is being made public, and how it can be used to track them. The use of proprietary data formats makes it impossible to tell what data is actually being recorded and shared. This includes the embedding of camera type in images from digital cameras. One way of controlling information when you are the one publishing it is to use techniques such as not sharing the information digitally, cryptography, steganonography, and shared secrets. Another option is to use a personal privacy statement and choose what information to disclose based on the privacy statement of the third party. However, these precautions cannot mitigate the fact that a person is not the only one who controls what information about him or her is being made available. It can be difficult to use information for good purposes without inadvertently violating someone's privacy.

There is a tendency to believe a perceived authority without thought, and the Web can be perceived as an authority. In addition, people tend to trust themselves, despite their own abilities to make mistakes. These problems could be handled by verifying the correctness of data, but that is not a common practice.

Overall, technology can be used for both good and bad purposes and the abuse of technology is common. Klein suggests that information handling "makes it easy to be good" and "makes it hard to be perverse." As an example of this type of information-handling system, he mentioned an ancient contract system that involved an inner, unmodifiable contract inside a publicly visible copy of the contract. He promoted the use of open source code and open standards to ensure that both the amount and the nature of data published are visible.

INVITED TALK

QA and the System Administrator

Adam Haberlach, Google

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

Haberlach's talk focused on the questions "What is quality assurance (QA)?" and "How does QA fit in?" As "Internal Systems QA" scaled at Google a dedicated "Operations QA" team was spawned. Adam encouraged the audience to look into the question of what QA is, and what it is not, giving definition to the scope of his work.

He gave some general examples of QA at Google, then drilled down to illustrate eight use cases for the Systems Operations and Network Operations groups. Key points were in the performance testing of LDAP directory services, to ensure that global performance is up to par. Another role of the group was to test the desktop platforms, to ensure that each hardware platform globally, performed as expected with the software applications engineers need to do their daily jobs. Adam mentioned that a GUI-focus-based UI testing tool, Eggplant (http://www.redstonesoftware.com), was effective in ensuring that Windows machines behaved as expected. Simple repetitive tests help to make sure that new changes to an OS environment do not break core user functionality.

The end of the talk focused more on traditional software QA, ensuring that internal applications have unit tests and that developers use their own software. Adam had some tips for QA teams to fit into the big picture, making sure that QA is committed to making software and services for customers and clients. Getting started in QA is easiest if you can fit into a large, long-term project and sell yourself via viral marketing. In the end, the Operations Quality Assurance team at Google helps to ensure that the software and services that the Operations group oversees run smoothly on a day-to-day basis.

SYSTEMS AND NETWORK MANAGEMENT

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

LiveOps: Systems Management as a Service

Chad Verbowski, Microsoft Research; Juhan Lee and Xiaogang Liu, Microsoft MSN; Roussi Roussev, Florida Institute of Technology; Yi-Min Wang, Microsoft Research

Chad Verbowski presented several new approaches to handling the management of large networks of Windows machines. Often, with a large site, you cannot keep track of every change made on a machine to figure out which single change impacted the system. The LiveOps system inserts itself as a kernel-level driver to passively monitor what is happening on the machine, keeping logs of transactions happening on that machine. It is possible, with low overhead, to monitor what process forked another, at what time, and by what user. Chad presented the example of discovering how eMusic was installed on an MSN server. By backtracking through some intuitive Web interfaces, he could see that the eMusic installer was launched from a Winamp Media Player installer, which was launched by Microsoft Internet Explorer.

Chad highlighted another example, one of installing service packs on a large number of machines. It is hard to determine whether the service pack has done everything it was intended to do. By utilizing a "Stale Files" feature of the LiveOps service, one is easily able to see that a subset of the total affected machines did not get properly updated, owing to the lack of a reboot. The LiveOps service attempts to help administrators track the number one cause of system administration problems: unexpected user changes. The reporting features present in the LiveOps system enable the administrators to discover the relationships between changes and events on their systems.

Managing Large Networks of Virtual Machines

Kyrre Begnum, Oslo University College

Kyrre Begnum made a key point to his audience: Setting up a complex virtual machine environment for a class or lab is difficult and arduous. Kyrre introduced a tool developed at Oslo University College called MLN (Managing Large Networks). MLN allows for the use of a simple configuration syntax to build reasonably complex virtualized scenarios. These scenarios, or projects, can consist of many virtual machines and switches, and they can be used with UML or Xen. The configuration syntax allows for the creation of superclasses and is extensible through plug-ins. Illustrated was a sample plug-in, one that would autoenumerate parameters such as an IP address for a large quantity of hosts.

Kyrre gave two prerecorded demos for the audience: One demonstrated the creation of a virtual machine LAN from a configuration file, and the other was a demonstration of cold migration from one Xen host to another. The MLN tool recognizes the project as a whole and will migrate everything needed to run the project on another host, by changing only one parameter. MLN is a cool tool to work with large virtual networks, enabling one to think more about the key factors involved and less about how to accomplish them.

More information on MLN can be obtained from http://mln.sf.net/.

Directing Change Using Bcfg2

Narayan Desai, Rick Bradshaw, and Cory Lueninghoener, Argonne National Laboratory

Narayan Desai presented a interesting paper about change management. Using the bcfg2 tool, developed at Argonne National Laboratory (ANL), Narayan and his team applied supplemental features to allow for integrated change management. A Subversion repository backend was added to bcfg2, and the server was modified to allow a site to pick any revision available in the repository, not just the version available in the HEAD revision. The team also chose to modify the client side, so that they could track reporting information alongside the associated revision.

The speaker covered some theory behind how changes are made and applied. The scenario described involved coordinating changes to the configuration repository to indicate current and future applied configurations, then mapping out a schedule for those changes to be applied, because of the ordered approach that configuration takes. Change management is not a new idea, but the

implementation that ANL provides helps administrators take advantage of a tight integration between change management and configuration management.

More information on bcfg2 can be found at http://trac.mcs.anl.gov/projects/bcfg2/.

INVITED TALK

High Availability: From Luxury to Commonplace Necessity in 10 Years

Eric Hennessey, Group Technical Product Manager, Symantec Corp.

Summarized by Ning Wu (ningwu@cs.tufts.edu)

Eric Hennessey reviewed the history of high availability (HA) and how HA solutions evolved from life before HA, to server-centric HA, and then to application-centric HA. In the "old days" HA was implemented via standby servers that shared the same storage and provided failover capability. After improvements in storage technology, the "N+1 architecture" became popular: These involved N applications and one standby server. When one application failed, the application was moved to the standby server. Later, the "N-to-N architecture" emerged, in which there was no spare machine and excess capacity on each server was used to provide failover. Eric also talked about data replication (DR) technology, which has become an integral component of local HA and also provides failover across wide area networks.

Currently, HA solutions are application-centric. Applications can now run on virtually any machine with access to appropriate storage. The challenges this brings include server proliferation (more and more servers) and decreased server utilization (with some using only 15%). Applications have also become more and more complex; layered structure brings more dependencies. As IT provides more services, customers are demanding more. Customers want HA for more applications. A Gartner report shows that 58% of applications are considered critical, while an informal survey shows that 5%–10% are protected by HA. The main reason for the gap is cost.

Facing increased complexity and higher SLA requirements, server proliferation, and limited staff and budget, we need integrated solutions. In the next few years, through comprehensive data center automation (configuration management, provisioning management, and application management), high availability will become a matter of routine, not exception, and each application will get as much HA as it needs.

INVITED TALK

What Do You Mean, Identity 2.0?

Cat Okita, Earthworks

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

Cat Okita provided an introduction to the concept of Identity 2.0, the motivations behind it, and the current state of the Identity 2.0 movement. Okita began by discussing general identity management concepts and history. A digital identity is defined as a collection of claims attached to a digital subject. She related four standard ways to treat identity management: the traditional user account method, the per-environment centralization of user accounts, the data management view, and the marketing view.

Okita discussed the different suggested properties of an identity management system and went on to relate her own recommended properties. An identity management system should be:

1. Minimal, designed for selective rather than permissive sharing of aspects of identity.
2. Verifiable, providing a means to verify assertions made by a subject.
3. Unlinkable, preventing the ability to take one aspect of a person's digital identity and link to other aspects of that person's digital identity.
4. Usable, making it something that will actually be used.

In order to provide users with control over their data, an identity management system must be designed to provide anonymity by having a default deny policy for sharing information about someone. The system can then choose who is allowed to access what information about one's digital identity. This is critical because, once it is shared, information cannot be made private again.

Problems with identity management involve the inability to know who has your information, what is being done with the information, and how it is being shared.

Identity 2.0 is an identity management scheme designed to allow individuals to control the aspects of their digital identity by limiting how it is shared. Identity 2.0 involves three key controlling entities: a digital subject, a relying party, and an identity provider. A digital subject has multiple digital identities, which are stored by an identity provider. When digital subjects want to interact with a relying party, they can choose what digital identity they want to use, based on what type of credentials the relying party accepts. They will

then send those credentials to the relying party, and the relying party will select the appropriate identity provider to confirm that the credentials are valid.

Okita then provided an overview of the current state of Identity 2.0. She mentioned important players in Identity 2.0, including standards, protocols, frameworks, and Web applications related to the movement. She then discussed the current areas of development in the movement.

Identity 2.0 looks to solve various identity management problems. Individual concerns include the management of many multiple identities, keeping track of the associated passwords, and controlling the flow of information. For those responsible for the digital identities of others, Identity 2.0 can ease the process of managing the information, sharing it within the organization and between organizations, and meeting security audit compliance. For commercial interests, it facilitates sales, helps to track habits, and promotes customer confidence. For the government, it helps reduce complexity and improve manageability.

Okita concluded that the Identity 2.0 movement is developing in several promising areas. However, progress remains to be made if it is to meet its goals.

VISUALIZATION

Summarized by Robert Marmorstein (rmmarm@cs.wm.edu)

NAF: The NetSA Aggregated Flow Tool Suite

Brian Trammell, CERT/NetSA Carnegie Mellon University; Carrie Gates, CA Labs

Brian Trammell presented a tool for aggregating and displaying network flows. The tool he and Carrie have developed is like a Swiss army knife: It has tools for handling many different kinds of inputs (including IPFIX, Argus, Silk, and even pcap!) and provides a wide variety of filtering and analysis operations.

The tool consists of three utilities which together provide a comprehensive netflow aggregation suite. The "nafilize" utility allows the user to apply aggregation and filtering expressions to a set of flows. The "nafscii" utility converts the binary output of the other utilities into a human-readable format. The binary output can also be converted into a graphical plot of the aggregation data. The "nafilter" utility is a lightweight filtering component with no aggregation capability.

In addition to aggregation and filtering, these utilities allow the user to sort on any key or value field. They also provide a "top-N" listing feature that can, for instance, show you the top 15 most common source addresses of all SSH packets sent in the last hour. The tool can manipulate either unidirectional or bidirectional flows and can even combine related unidirectional flows into bidirectional flows.

The tool is available from http://tools.netsa.cert.org.

Interactive Network Management Visualization with SVG and AJAX

Athanasios Douitsis and Dimitrios Kalogeras, National Technical University of Athens, Greece

This paper focused on ways to allow administrators to create network-related visualizations without suffering through the complexities of modern graphical APIs. The framework that Athanasios presented provides a simple but flexible API for depicting important network data. Using this framework, developers can create interactive visualizations for observing and managing the network.

The framework is designed to be modular, interactive, and reasonably secure. It provides functions for displaying and manipulating both unidirectional and bidirectional graphs. The framework uses a client/server architecture. Data is collected by a management server, which formats it as a set of XML documents and transmits it to a Javascript client, which renders it by using scalable vector graphics.

The tool is not yet available, but it will be released when it is considered stable enough for production use.

Bridging the Host-Network Divide: Survey, Taxonomy, and Solution

Glenn A. Fink and Vyas Duggirala, Virginia Polytechnic Institute and State University; Ricardo Correa, University of Pennsylvania; Chris North, Virginia Polytechnic Institute and State University

Glenn presented HoNe, a network visualization tool he developed as part of his dissertation research. Unlike existing tools, HoNe can correlate network connections with processes on the sending or receiving host. This makes the tool particularly useful for visualizing security-related information.

The main display window categorizes hosts into categories based on whether they are inside or

outside the enterprise and whether they are "hosts we manage" or "hosts someone else manages." Other windows provide connection filtering and display time-related data about currently selected connections.

Glenn also described the challenges in obtaining data about the relationships between processes and network connections. After trying to obtain this information using various userspace tools, he finally decided that modifying the kernel was the only effective and accurate solution. A usability study of the tool found that the packet-process correlation was a novel and helpful instrument for both novice and expert users.

INVITED TALK

The Last, Best Hope: Sysadmins and DBAs as the Last Guardians of Privacy

Danny O'Brien, Activism Coordinator, Electronic Frontier Foundation

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

Danny O'Brian began by discussing his organization. He briefly covered the EFF's activities, funding, makeup, and goals. He mentioned three areas in which the EFF is involved: technical research, legal representation, and publicity and advocate work.

O'Brian then moved on to cover the need to update constitutional rights over time and the difficulties involved in this process. In particular, he focused on the 4th Amendment and the need to update the amendment to reflect new technology. To illustrate the typical slow pace and intricacies involved in such a revision, he detailed how the 4th Amendment was applied to conversations over the telephone. In 1927, the Supreme Court ruled that the 4th Amendment did not apply if no physical trespass occurred and the items gathered were not tangible. This was slowly changed, as the technology advanced and various cases began to show the problems in that decision. Eventually, the court did decide that the amendment applied to people and not places. However, it did not protect information given to a third party and then shared with the government.

The modern way of life involves private information being shared in way it would not be in the past, which requires a change to strengthen the constitutional protection of privacy. The way the data is stored should not affect the privacy rights accorded to that data, but that is the current impact of the third-party exception. To handle this problem, the EFF advises courts and judges about the need for change, advises users about how the current laws can affect them, and advises companies about the implications of their actions under the current laws.

O'Brian went on to discuss how system administrators can help this process. There are three aspects involved in the development of a civil liberties law: law, running code, and culture. One area of the culture that could be adjusted to better support privacy is the idea of logging by default. Instead, it would be preferable to consider when, what, and how much actually needs to be logged to serve the desired purpose. One current change to system administrator code is to use data storage techniques that will restrict the use of the data to its original use. An example of this is in the book *Translucent Databases,* by Peter Wayner, which describes mechanisms to protect the stored data from being used in any other way than its original purpose. This involves encryption, ignorance, minimization, misdirection, stunt data, equivalence, and quantization.

The use of these techniques can be supported by pointing out how it would be in the company's best interests. This includes the need to follow privacy policies that state that data will only be used in one way. Another factor is reducing the cost of trying to discover data when it is used in a lawsuit. These approaches can also help to avoid the issue of the company becoming a target of government agencies because of the amount of data stored. Finally, there is the ability to reduce the cost of storing the information.

The ultimate goal is a change in the law. However, the changes in code and culture are important because judges tend to look at existing practices when interpreting the law. As these changes are made in the realm of technology, the idea of the importance of privacy of electronic data will spread to the larger culture. This will help to change the law, as judges try to reflect the existing culture when deciding the meaning of the law.

O'Brian was asked about the Electronic Frontier Foundation's degree of collaboration with the ACLU and organizations such as moveon.org. He replied that they work closely with the ACLU. He added that the EFF is nonpartisan and tends to work with organizations on both sides of the political aisle.

INVITED TALK

Command and Control: System Administration at U.S. Central Command

Andrew Seely, Global Command and Control System Lead, HQUSCENTCOM-J6/Northrop Grumman Defense Mission Systems

Summarized by Kevin L. James (kevljame@cs.iupui.edu)

Andrew Seely is a system administrator for U.S. Central Command (CENTCOM), the Command and Control (C2) arm of the U.S. Military tasked with operations in the Middle East and surrounding areas. He set out to both show how his job is similar to system administration in other industries and how this unique niche sometimes requires unorthodox problem solving.

CENTCOM is responsible for managing and distributing information concerning force deployments and the resources needed to support these forces and protect them. The planning, logistics, and intelligence of missions are also its responsibility for not just war but also rescue and disaster relief efforts. To deliver these services, CENTCOM provides information to the Global Command and Control System, GCCS-J, a family of systems facilitating everything from planning and intelligence gathering to the tracking of global readiness and ballistic missiles. GCCS is also used by coalition partners and is being integrated into many homeland security organizations.

Interoperability is the system's strength, as GCCS is able to integrate raw data from many diverse sources and present it in a cohesive manner. But, according to Seely, this comes at a cost: glacial technology migration, many levels of bureaucracy to gain required approval, and months (even years) of testing. For instance, he expects that GCCS will be using Windows XP by 2007, though most likely later. Thus, CENTCOM's C2 capability is made up of systems considered by industry to be outdated, even ancient, because stability is paramount.

To accomplish these goals, C2 system administration requires constant vigilance to maintain reliability and flexibility to accommodate ever-changing requirements in order to fulfill a much broader mission. Although these requirements may sound familiar, the unique "tactical conditions" of performing these duties, which include live battlefields, present interesting challenges. Among those Seely described were power, communications, and resource shortages caused by mortars and adverse weather. These were coupled with locally hired staff who often do not speak the same language and the high turnover resulting from personnel rotation. In addition, the systems and applications that he supports are not chosen or even configured by him when he receives them. Everything comes through the Defense Information Systems Agency (DISA), a government clearinghouse that vets and tests applications and preconfigures machines to exact specifications and allowances. Many of these come from contractors working on classified contracts, so support is difficult to obtain, if even possible, and thus they are virtual black boxes that operate without regard for other applications or systems. Yet all must be integrated into this monolithic system for the continued success of the mission. All in all, Seely said, the most important requirement is that "a wide range of expertise is needed at a moment's notice: You have to be sharp and learn fast."

After describing the environment, Seely gave examples of problems he has faced and steps taken to solve them. One such problem involved an attempt to save setup time on the installation of two new machines, each of which, because of accreditation requirements, requires a specially tasked team two weeks to set up. Another wrinkle was that one machine was at CENTCOM Headquarters in Tampa and the other was in Qatar, a small country in the Arabian Gulf. An attempt to simplify the setup of one machine, make a backup, and then restore the backup on the other machine was balked when the tape containing the backup never arrived in Qatar. Communications between Tampa and Qatar were shaky, so the entire backup couldn't be sent by FTP. Consequently, everything was sent file by file, but larger files failed repeatedly. Because of the tight configuration control required by the government, simple tools such as compilers and utilities such as split are unavailable on his systems. To solve the problem, Seely decided to implement split and cat functionality himself, in Perl.

Inventive solutions such as this are often the rule in C2 environments because of the controls placed on those working in them. To solve a problem, the solution not only has to work but must be built with the few tools available to the administrator. Although he acknowledges the obvious problems with this, Andrew's approach is to assume crisis to be the norm because, as in all our jobs, often it is.

Black Ops 2006: Pattern Recognition

Dan Kaminsky, Doxpara Research

Summarized by Nathaniel Husted (nhusted@iupui.edu)

Dan has spoken at the Black Hat Briefings for the past six years and is also the coauthor of several books. He is also a member of the "blue hat hackers" Vista audit. Dan discussed various topics, including network neutrality and how to detect that it is being violated, a better way for users to recognize SSH keys, finding structures in hex dump files for use in fuzzing, visually showing file structures at a binary level, and flaws in SSL communications.

The first subject discussed was determining when network neutrality was being violated. Dan suggested that exploiting a behavior in the TCP protocol was a useful way to determine when service providers slow various packets down. The specific TCP behavior being exploited is the protocol dropping extra packets when the channel is saturated with information. Based on this behavior, Dan suggested that if someone sends 100 KB on one channel, one can tell what link is causing a drop in speed based on dropped packets. By spoofing source IPs, a person can then determine which providers are acceptable, and which providers are not. One can also spoof various payloads to determine what content is acceptable, and what is not. Other protocols useful for this purpose include RTP and RTCP.

The second subject Dan discussed is the weakness in SSH key validation by the user. Every time a user sees an SSH key she or he doesn't recognize, it should not be accepted. Generally, this doesn't happen. To improve the user's ability to recognize SSH keys that belong to the user's servers, Dan suggested replacing the hex values with names from the U.S. Census report. Five names provide the same amount of entropy as the hex keys that are normally used. Dan also mentioned some other methods that have been thought of, including using abstract art and people's faces.

The third subject discussed was how to make hex dumps more usable for fuzzing by finding structures in the files. Dan stated that hex is generally hard for people to read. This makes it especially troublesome during a process called fuzzing. This is when a user tries to input various types of data to make a program do something that isn't desired. The two types of fuzzing include smart fuzzing and dumb fuzzing. Smart fuzzing is when a user requires knowledge of the underlying structure of a system and then has a specific attack to exploit that system. Dumb fuzzing is when the user will input random data into areas that he or she thinks might cause the system to stop working. To improve the dumb fuzzing process, Dan suggested using the Sequitur function to determine and highlight any structures in a hex dump. This technique was based on a paper by Craig Neville-Manning written during his Ph.D. research. The major benefit to using this technique is that it scales very nicely, even if it is not the best way to generate grammar.

The visualization technique Dan talked about next was somewhat related to the Sequitur function. This technique is based upon Jonathan Foote's paper "Visualizing Music and Audio Using Self-Similarity," as well as "DotPlot Patterns: A Literal Look at Pattern Languages," by Jonathan Helmans. This visualization technique uses white, black, and grey pixels to determine whether various file bits are similar, dissimilar, or semi-similar. This technique then creates a patterned image with an equality line running diagonal from the top left to the bottom right of the image file it produces. What is special about this technique, Dan says, is that it is actually useful. When using the equality line as a reference, anyone analyzing the file can then pinpoint where exactly in the file a point in the image occurs. Color was also developed for this application based on suggestions from http://colorbrewer.org.

In a last-minute addition, Dan also discussed flaws in various SSL implementations. The first thing Dan suggested was not to put the same SSL key on different boxes. Also, he suggested that if you want to keep some DNS names secret, you need to be careful about what certificates users are allowed to scan. Finally, Dan suggested that banks need to rethink how they provide user authentication. Most banks do not have users enter login information on a secure SSL page. This can allow a hacker to hijack your session. Dan suggested that banks use iframes to cache the SSL secured page, and then switch to the protected page via JavaScript when a user goes to enter his or her password.

Unrelated to anything else, Dan also mentioned that SSH works very well as an extremely flexible VPN solution. However, he noticed that it had a tendency to leak DNS requests from remote users onto the local LAN. To resolve this problem he found a way to tunnel the DNS requests by basically going from DNS to SSH back to DNS.

All the slides from this presentation can be found at http://www.doxpara.com/ and at http://www.usenix.org/events/lisa06/tech/slides/kaminsky.pdf. Any released code can also be found at the Doxpara Web site. Dan has stated that he will respond to emails requesting the code talked about in this presentation.

Most questions revolved around whether the programs Dan discussed in his presentation were released or not. Others asked what he used his programs for. Dan stated that currently his hex program will not properly display files that have structures on bit boundaries and not byte boundaries. He also suggested that it may have potential uses for system administrator data sets, but he has yet to really test that.

INVITED TALK

Seriously, Tape-Only Backup Systems Are Dead

W. Curtis Preston, Glasshouse

Summarized by Nathaniel Husted (nhusted@iupui.edu)

W. Curtis Preston has been a data protection specialist for 13 years and is known to his friends as "Mr. Backup." According to Preston, if you're performing LAN-based backups directly to today's tape drives, you're doing nothing but shooting yourself in the foot. The problem with tape drives is that they are much faster then our networks and a streaming tape drive cannot write slower than its minimum speed. Matching the speed of networks with the speed of tape drives is not possible these days, because there has been a 27% increase in the speed of tape drives every year. Before 2001, tape drives and networks had the same speed, from 2001 to 2003 networks were faster than tape drives, but after 2003 tape drives outpaced networks, causing problems.

Variable-speed tape drives are available in the marketplace but all still have a minimum speed. Despite vendor claims, these drives have not eliminated shoe-shining. The good news is that a number of vendors have worked very hard on disk-based solutions that solve all these problems. Disk speeds are infinitely variable. Disks can take hundreds of slow or fast backups all at once without multiplexing. You can then dump them to tape at its maximum speed or replicate them to another location.

Disk can be used in various ways for backup both as a disk (e.g., SAN or NAS) and as a tape (e.g., a virtual tape drive or a virtual tape library). Now the main question is whether to opt for a filesystem or a virtual tape. The answer to this question depends on multiple factors, such as which backup software you are using (since not all are fully filesystem aware), the speed of backup, and whether the disk itself supports fragmentation (disk as tape doesn't support it). There is a provisioning/sharing issue when disk is used as a disk. VTL can be used as a standalone as well as integrated. A standalone VTL system sits next to your physical tape library, whereas an integrated VTL system sits in front of your physical tape library. A standalone system pretends to be another tape library, whereas an integrated tape library pretends to be your physical tape library.

VTL can be used as a single node or in clustered mode. "Clustered" VTLs allow you to expand capacity or throughput by adding additional data movers, but they manage as a single VTL.

All major VTL vendors are releasing de-dupe products right now. File de-duplication (sometimes called data reduction factoring) is space-saving technology intended to eliminate redundant (duplicate) files on a storage system. By saving only one instance of a file, disk space can be significantly reduced. De-duplication reduces effective cost of disk by 10:1 or more. To identify the redundant data, a hash comparison is used; calculating hash can be done with different hash calculation algorithms, such as SHA-1, MD5, or custom algorithms. Sometimes bit-level comparison is also used to double-check. Most products available in the market can use two methods. After the redundancy is identified, forward or reverse referencing needs to be used to identify the new data.

De-duplication can be in-band or out-of-band. In-band de-dupes in RAM and never writes redundant data to disk. Out-of-band de-dupes write original data to disk, read it later, and de-dupe it; hence it requires more I/O then in-band de-dupes.

There are various open source backup tools available in the market (e.g., BackupPC, Rdiff-backup, Rsnapshot). Preston also strongly recommends reading his book *Backup & Recovery*, which has 750 pages dedicated to free and open source backup.

POTPOURRI

Summarized by Gautam Singaraju (singara@uncc.edu)

The NMI Build & Test Laboratory: Continuous Integration Framework for Distributed Computing Software

Andrew Pavlo, Peter Couvares, Rebekah Gietzel, Anatoly Karp, Ian D. Alderman, and Miron Livny, University of Wisconsin, Madison; Charles Bacon, Argonne National Laboratory

Andrew Pavlo presented a framework for building and testing software in a heterogeneous, multi-

user, distributed computing environment. The automated tool, developed as a part of NSF Middleware Initiative, provides automated builds and tests access across administrative boundaries.

The users explicitly define the execution workflow of build-and-test procedures, which is stored at a central repository. The framework will dynamically deploy tests to appropriate computing resources. Any artifact that is created during the test process is transferred to the central repository. The NMI tool has been implemented on top of Condor, a high-throughput distributed batch-computing support tool. The NMI tool is both tool and platform independent, is lightweight,and provides well-controlled environments, centralized results, fault tolerance, and test separation.

Responding to questions, Pavlo pointed out that the software allows the users to control their test environment with a centralized repository that allows the users to replay their tests. Pavlo invited users to download the tool from http://nmi.cs.wisc.edu/.

- ***Unifying Unified Voice Messaging***

Jon Finke, Rensselaer Polytechnic Institute

Jon Finke presented his experiences merging a voice messaging system with the email domain. Rensselaer installed the unified voice messaging system after the voicemail system failed. By unifying the voice messaging into the Exchange servers, users were able to listen to their voicemail from their inbox. The files can be downloaded as .wav files, which can be used to forward the messages to other users. The Cisco Unity voicemail system was used to interact with Exchange servers. Standalone Exchange servers were used because the Active Directory schema change was not appreciated by the Exchange installation support personnel.

A tool was developed that managed voicemail by creating the mailboxes and populating the appropriate call handlers. With a call handler changing the extension was not necessary when a student changed rooms. Once the unified voicemail was configured on standalone Exchange servers, the system was then migrated to the production email domain with the help of a tool that copied each user's content into the production server.

- ***Fighting Institutional Memory Loss: The Trackle Integrated Issue and Solution Tracking System***

Daniel S. Crosta and Matthew J. Singleton, Swarthmore College Computer Society; Benjamin A. Kuperman, Swarthmore College

In the last presentation, a tool for part-time system administrators was presented. Trackle keeps a record of past actions, as the tool is used for documenting system services. Trackle provides an integrated trouble ticket and solution tracking system. Trackle has been developed as a tool that documents the process that is performed by an experienced system administrator. These actions are documented and can be used in educating untrained student system administrators.

Trackle has been developed to provide functionality for system administrators and users, an easy ticket-filing mechanism, and wiki-like referencing tools with minimal dependencies on existing software.

Referring to the additional feature requests, Crosta stated that the tool will provide ticket extensions, multiple machine support, file revision control, and further high-level abstractions. When asked about the tool, Crosta invited system administrators to download and try Trackle from http://www.sccs.swarthmore.edu/org/trackle/.

NETWORK SECURITY TRACK

- ***Zombies and Botnets: Attacks on Messaging Security by Organized Criminal Enterprises***

Dmitri Alperovitch, Research Scientist, Cipher Trust, Inc.
Summarized by Nathaniel Husted (nhusted@iupui.edu)

Dmitri Alperovitch is a former researcher for CipherTrust and now works for Secure Computing after the merger. He discussed recent trends in online attacks such as the increase in spam and phishing. He also discussed the prevalence of botnets and touched upon the organizations behind them. The talk ended with a short question-and-answer session.

Dmitri started by discussing various trends in online criminal activity from the past 25 years. Dmitri stated that criminals are getting dumber, but their populations are increasing. He also mentioned that the attacks are getting smarter and so are the security tools. Everyone is now a victim of these attacks. He also broke into three stages the types of people who have been behind these attacks for the past 25 years. The first stage was the hackers of the 1980s. The second stage was the crackers of the 1990s. The third and final stage is the organized crime enterprises we are now seeing in the 2000s. Dmitri said that these organized crime enterprises are now winning. Fully 90% of all email is spam, 400,000 bots appear daily, 5,990 vulnerabilities were reported in 2005, 3,744 phishing sites are found monthly, 200,000 viruses have been reported this year, and all these numbers have gone up from their previously recorded metrics.

Dmitri did state that things are improving, but there are also challenges. Law enforcement efforts are improving, arrests and prosecutions are going up around the world, international cooperation and trust are improving, and laws are slowly catching up to technology. Progress is still slow, however, because investigations can take months and even years to complete. There is also considerable corruption in some countries where these Internet criminals operate, thus allowing them to buy their way out of the judicial system. The enemy is also progressing in its tactics, becoming more secretive, less centralized, and operating with more secure communication channels.

Dmitri then discussed how spam and phishing attacks have evolved over the years. In the 1990s spam consisted of basic text-based messages. Then in 2003 they included random words to throw off the content-filtering spam blockers. Now, in 2006, the appearance of image-based spam has required a new style of OCR-based content filtering. The spammers are also committing large-scale but short-lived zombie attacks that make blacklisting useless. Dmitri also stated that in the future more spam will be image-based and possibly composed of random images from a zombie computer's hard drive. Phishing evolved from small-scale email-based phishes to special trojan-creating toolkits with a software support structure. Trojans were preferred over traditional phishing emails and Web sites because they are longer lived and easier to use. Dmitri ended the spam and phishing discussion by talking about various forms of online bank security and how much trojan-creating software costs.

The presentation finished with Dmitri discussing zombies and botnets. Zombies are the workhorse behind almost all online attacks. Now zombies adapt themselves to check for blacklists. Dmitri also stated that 20% of all bots are currently located in China, with the United States coming in second at 10.55%. Bots now have greater intelligence and also use peer-to-peer communication mechanisms instead of IRC. This modification in how bots talk to one another has made botnets harder to shut down. Dmitri stated that the best defense against botnets is to shut them down if possible or filter traffic from any known compromised network into a network with limited functionality. He also suggested that end users need to be made more responsible for the security of their machines. Other ways to lower the botnet population include increasing banking security and lowering the monetary benefits of spamming. Sadly, Dmitri stated that this problem will never go away, but we need to hope that it doesn't get any worse.

The questions asked in this talk revolved around what can be done to limit the propagation of zombies and why the problem is so bad. Dmitri suggested that one reason the bots are still spreading rapidly is that users still open executable email attachments. He also suggested that when IPv6 is implemented it might slow down the propagation because of the increased IP range that needs to be scanned. To make sure we are not the problem, Dmitri suggested that we make sure we know exactly where our network traffic is going.

INVITED TALK

Power-Managed Storage: Longer Data Life and Lower Energy Consumption

Aloke Guha, CTO, COPAN Systems

Summarized by Sumeet Jain (jain.sumeet@yahoo.com)

Aloke Guha began by saying that we have witnessed some of the most extraordinary growth in the recent history of technology in computing power, switching/routing capability, and data-carrying capacity. Data storage growth has outpaced all other growth rates, being explosive rather than exponential, touching 200 billion gigabytes in the year 2007, up from 12 billion in 2002. This focus has changed from system-centric in the 1970s to content-centric, with a complementary increase from ten million users to close to a billion users currently. We have witnessed islands of data move from Monolithic Direct Attached Subsystems to Dynamic Tiered Storage (DTS). DTS is capable of handling transactional as well as persistent data.

Managing persistent data is easier said than done when compared to managing transactional data. Persistent data, though matured (i.e., having a very low probability of any changes), has to be retained for a longer duration not only because of stringent regulatory compliance requirements but also because of the vital role it plays in business today. Coupled with the event-driven requirement, bandwidth constraints, and small recovery windows of few hours, the challenge of managing large volumes of data on tape drives is mission impossible. Tape backups are more beneficial for vaulting.

Storage on disks has its own challenges: power, cooling, reliability, longevity, maintaining life cycle from migration to salvage/regeneration, and, last but not least, floor space requirements. According to recent studies, power costs consume 40% of IT budgets and 33% of data centers expect to be out

of power and cooling capacity by the end of 2007, while 96% expect to reach the ceiling by 2011. More persistent data, almost 80% of which is being retained for a longer duration, still consumes power, has cooling requirements, and needs to be accessed online for regulatory compliance, while more data is being generated each minute for corporate governance and business continuity.

This is where MAID can help in managing data in a better and intelligent manner. MAID (Massive Array of Inexpensive Disks) is basically a large array of power-managed disks, with 50% of its drives powered off and only 25% of its drives spinning at any given time. It's a storage system comprising an array of disk drives that are powered down individually or in groups when not required, which helps to reduce power consumption. MAID has a three-tier architecture to scale performance with capacity and uses DISK AEROBICS software to check disk reliability and data integrity.

COPAN provides Enhanced MAID systems, which scale from 28 TB to 448 TB, with 5.2 TB per hour performance and the capability of handling 1 billion stored files (file and disk block) or 8,192 virtual tape cartridges. Since the system has been designed inside-outside with energy consumption and data integrity concerns, MAID performance outpaces FC storage systems by 2,140% and SATA-based storage systems by 424% on a TB/kW unit.

The MTBF of DISK AEROBICS software is 3.3 million hours, compared to 1.2 million for FC and 0.6 million for SATA. DISK AEROBICS proactively monitors and manages drives as well as RAID groups. Any suspect drive is backed up on spare drives and is "failed out" to avoid long RAID rebuilds. It performs policy-based turn-off of drives or RAID groups when not in use. It also assures drive health by exercising and testing idle drives at least once every 30 days to ensure data integrity.

INVITED TALK

The Future of System Administration: How to Stop Worrying and Learn to Love Self-Managing Systems

Alva L. Couch, Associate Professor of Computer Science, Tufts University

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

Alva Couch gave an informative talk about how the profession of system administration must change in the face of vendor-propagated autonomic computing initiatives. Vendors would have us believe that self-managing systems will be able to handle most current administration tasks on their own. On the assumption that they are correct, Couch proposes a clear path for evolving the job title of today's system administrators from "Plumber" to "Translator/Manager." To help us along, the talk focuses on the big picture before and after the coming "autonomic revolution." For example, the duties of managing configurations, twiddling bits on disk, troubleshooting configurations, and understanding file formats will be replaced by the duties of managing architecture, setting policy, analyzing system dynamics, and understanding performance factors, respectively. The dream of autonomics is to have present-day managers input business process and have everything work properly. The difficulty is that "manager speak" does not represent to an autonomic system what is *needed*. This is where the new sysadmin comes in. He or she will figure out what the real business process is, what aspects of it can be supported, and how to translate it for implementation by an autonomic infrastructure.

Pushing the profession into the next phase requires not just a new set of skills, but a new attitude as well. According to Couch, we have not yet started to retool our thinking in a way that will produce the new breed of sysadmin. This is dangerous because the modern world no longer promotes "survival of the fittest"; rather, we are in a world of "survival of those who fit." The old niche is full, and without a change in attitude there will be a slow death for the profession as we know it. We must create a new niche populated by "managers of human-computer communities." Old survival skills (communication, intrapersonal, time management, analysis, etc.) must be coupled with new survival attitudes: Value yourself and your professionalism, place management goals above self-interest (which requires understanding the attitudes and language of management), be able to "close the box" and delegate, and be able to leave good enough alone. The new sysadmin cannot base his or her job security upon being essential now; sysadmins must be perceived as essential to the future. The best way to become more important and indispensable is to get on management's radar by making your job the easiest (and most efficient) way to accomplish business objectives.

Couch went to review several case studies and lessons from his own experience. Good works are not always sufficient to keep your job in the face of changing business needs or structure: In a single year, everyone important at Tufts University

who owed Couch a favor for pulling them out of the fire was let go, forcing him to rebuild his reputation. As another example, he took a look at database administration. He made the convincing points that design is already outsourced, automation can tune performance to within 80% of human capability, and much programming is being replaced with reflection modeling. To maintain his value, the new DBA really needs to serve as the interface (and interpreter) between management and infrastructure. As a third angle, Couch urged us to consider autonomics as analogous to asbestos abatement: This stuff is dangerous; one slip and the business loses a lot of money; it's all driven by complex policies that untrained people (i.e., those in management) shouldn't try to understand. Finally, Couch provided tips for interfacing with management: Stop distinguishing between "us" and "them"; make our goals the same as those of management; learn to speak like the managers; learn to justify our decisions in management terms; listen intently instead of explaining; and make ourselves partners rather than servants.

During the Q&A session it was pointed out that this type of evolution is nothing new. Couch agreed but went a step further by claiming that if autonomics begins to replace the junior sysadmin, the training loop will be broken and it may spell disaster for the profession. It was asked whether we have enough time to evolve, given the reduction in LISA attendance in the past few years. The response was that the drop was due to social rather than technical factors and that the profession may actually be reaching a saturation point (all the more reason to evolve more quickly). When asked to what extent the new breed of sysadmin will need (or want) to understand the technology of autonomics itself, Couch answered by analogy to admins understanding the kernel.

WORK-IN-PROGRESS REPORTS

Summarized by Robert Marmorstein and Beth-Lynn Eicher (rmmarm@cs.wm.edu, bethlynn@lookandsee.net)

- ***NAGIOS and SEC: A Happy Re-Union for Advanced System Monitoring***

John Rouillard, University of Massachusetts, Boston

SEC is an event correlator that supports many UNIX platforms. SEC can act as an event mapping layer between NAGIOS plug-ins and the NAGIOS core to set thresholds for alarms. SEC can also monitor NAGIOS log files for errors. John is currently looking for beta testers interested in this combination of tools. Slides are available at http://www.usenix.org/events/lisa06/wips.html.

- ***PoDIM: Policy Driven Infrastructure Management***

Thomas Delaet, Katholieke Universiteit, Leuven

PoDIM is a generalized and scalable mechanism for component distribution. It interfaces with several backends, including bcfg2 and cfengine2. The software is not currently available. Thomas is working on the PoDIM project as part of his Ph.D. study at K.U. Leuven, Belgium.

- ***NIS to Kerberos in a Psynch Environment***

David Pullman, National Institute of Standards & Technology

David talked about his experiences migrating an NIS-based account system to a Kerberos-with-LDAP system. He explained the challenges involved in providing account locking and unlocking, as well as giving an outline of the architecture he used to implement the transition. The result was a Psync portal that entailed NIS, LDAP, and Kerberos.

- ***Using Redirection to Enhance Software Management***

Marc Chiarini, Tufts University

Marc discussed the problem of nondistribution standard packages overwriting files belonging to other packages. He described a solution in which packages are repackaged and wrapped in a special environment so that libraries and other dependencies match. This also protects the core distribution and allows multiple software environments to coexist peacefully.

- ***Symlinking for Fun and Profit***

Wout Mertens

Wout talked about downward compatibility problems he had with multiple versions of Solaris sharing an NFS file system. By using symbolic links and custom-built scripts he was able to design a "poor man's union-mount" that alleviated these problems. He also discussed some of the stumbling blocks to this approach (e.g., the sudoers file cannot be a symlink) and how he was able to address those issues.

- ***Portable Cluster Computers and Infiniband Clusters***

Mitch Williams, Sandia National Laboratories

Mitch gave a slideshow presentation featuring various clusters he has built. These systems vary in size from a foot high to several racks. Two notable examples are the toolbox-sized cluster of 16 ARM 200s, which was presented on the show floor of Supercomputing 2006, and the 64-TB Flustre storage cluster, which is in production at Sandia National Laboratories.

■ ***Miscellaneous Data Management II***

Jason Heiss, Yahoo!

Configuration management has been a very hot topic in the system administration world. Jason pointed out that data management is also a very challenging problem that deserves attention. Jason talked about various solutions for managing drives, data, and backups. He focused on managing small chunks of data from service configurations, such as a Kerberos database.

■ ***A Configuration Management Tool Used by the University of Wisconsin, Madison, Computer Science Department***

David Parter, University of Wisconsin, Madison, Computer Science Department

Using an existing inventory database, David created a configuration management tool that simplifies installation, configuration, and maintenance of his department's systems. The tool uses simple XML templates to describe the system policy and interfaces with both kickstart and jumpstart. The system also created templates for common types of systems (desktops and research servers) at his university.

■ ***What Is a Computer?***

Beth Lynn Eicher, Look and See Networks

Is your cell phone a computer? What about your dishwasher or your toaster? The evolution of technology has blurred the lines between what is a computer and what is not. System administrators were challenged to think of computers and users in a larger than traditional scope. Beth Lynn broke the issue down into several yes-or-no questions and emphasized the importance of privacy in computing.

After each talk was presented the WiPs chair, Esther "Moose" Filderman, called for an audience vote by round of applause. Traditionally the crowd has one or two favorites, but this year everyone enjoyed all the presentations equally. The dilemma of who gets the coveted WiPs award was quickly resolved when someone had suggested that Anthony from MSI ("the A/V guy") deserved the prize for quickly resolving projector issues.

NETWORK SECURITY TRACK

■ ***Corporate Security: A Hacker Perspective***

Mark "Simple Nomad" Loveless, Security Architect, Vernier Networks, Inc.

Summarized by Kevin L. James (kevljame@cs.iupui.edu)

Mark Loveless enlightened us about the nature of hacking today. The founder of the NMRC hacker collective, now a security researcher, he still maintains ties with hackers on both sides of the fence: black hats, who crack systems for pride, politics, and profit, and white hats, who attempt to find flaws in systems before they can be exploited.

He began with a list of hacker "goals and dreams." The first goal is to find 0days (number zero) and to be the first to exploit a flaw. Next is remote root access, a dream of many hackers as it allows unfettered access to a system without being logged into the console. The holy grail of hackerdom is the remote root 0day, wherein not only is complete control gained, but in a way that has never before been seen, and therefore is more difficult to detect and stop. According to Loveless, 0days are worth more than ever.

Mark next expanded on the concept of 0day. Originally, 0day meant the number of days a commercial piece of software had been on the market before it was hacked. Cracking a copy of a new game before it is even released was considered "wicked cool." Attempts at this were common because of the copy protection measures used in the day: All software had to be cracked to back it up. When it came to exploiting security flaws, an exploit ceased to be 0day when vendors or system administrators discovered it. Today, 0day refers to an unpatched flaw that vendors and sys admins have discovered. This applies to both nonpublic working exploits for a patched flaw and those reported to vendors or industry groups by researchers.

Interestingly enough, both white hats and black hats have a disclosure cycle when it comes to discovered flaws. Researchers report flaws to software vendors, who in turn develop a fix for the flaw and release a patch for it. Afterwards, the researcher releases an advisory to a third-party group such as Carnegie Mellon's CERT (Computer Emergency Response Team, www.cert.org). Excluded from these advisories are the technical details of the flaw. In contrast, when attackers find flaws, they share their finds with very few close friends in an effort to minimize usage of the exploit and therefore vendor discovery. Before the flaw is discovered, the attacker also attempts to find other flaws. Mark says that hackers often sell their used or discovered flaws. When these two disclosure cycles clash, both hats work vigorously to reverse-engineer patched and unpatched versions of fixes using tools such as bindiff and clues from advisories to narrow the focus of reverse engineering. Similarly, they also develop exploit code based on the discovered flaw: white hats to develop scanning signatures and black hats to de-

velop code that will be used to attack unsuspecting systems. Another commonality to their jobs is that they often look evidence of silent patches done by vendors to determine whether they fix a possible exploit. According to Loveless, all vendors patch silently at times.

He next described trends in targeted penetration and attacking techniques. Although attempts at breaking into systems still abound, statistically, the successes have not grown proportionately. Referring to the popular book series, he called much of the current crop the "Hacking Exposed" generation, as many sys administrators protect their systems using these types of book. Conversely, these books are also used by would-be hackers. To their detriment, he says, these books talk simply about the act of penetration of a system, focusing on perimeter security. They also fail to give fledgling hackers tips, such as not hacking from your own system.

There are also some new "wrinkles" in old reconnaissance techniques. Many hackers use known exploits to make sure their attacks are discovered. This is done to judge the responsiveness of system administrators, to see whether they are simply immediately fixing exploits or actually watching the attacks to see their patterns. Another interesting technique is to use "dark IP space." Many admins check to see whether their unused IP space is being probed, because if someone is attempting to use unallocated IP space, they are probably an attacker. Conversely, clever attackers sometime attack this space to see whether an admin is really watching. There are even ideas to determine whether an automated system is in use and the type. A final technique is to use a 0day attack masked by many well-known exploits. The hope is that administrators will be too occupied dealing with the known problems to determine how the attacker actually got in.

Next Mark gave facts about the professional black hat world. Traditionally, they work for a single very organized group that specializes in spamming, spyware, and id theft. Many of these groups are run by organized crime organizations and are very much like regular software businesses, with tight release cycles and product testers. They are very well paid (around $200,000 per year), often for substandard work. Some of these activities are even funded by nation states, organized cybercrime, and even legitimate computer defense companies that are willing to pay $40,000 to $120,000 for a remote root 0day. There are even sites acting like EBay, complete with rating systems, where hackers buy, sell, and trade exploits and stolen identities.

Freelance black hats also work for spammers and information brokers. They are more concerned with keeping 0days hidden from vendors and administrators. Often they are very proficient at reverse engineering, making money from exploits alone. Loveless quips, "Anytime you couple questionable morals with some type of mad-coding-fu going on, you're going to make the mad-coding-fu money."

Next he talked about what's hot in hacker circles. He said that anything WiFi or Bluetooth is very popular, as they offer disconnected points of attack. There has also been a trend toward targeted malware as more money has become involved in the process. The more victims reported, the more likely people are to patch the flaw. To minimize this, IP ranges or targets that have been prelaundered by previous spamming efforts are more likely to be attacked. Another new area will be Blackberry-like devices, as they are able to connect with many diverse systems. Lastly, vast botnets (large clusters of machines that are used for attacks) are occurring in six-figure sizes and being leased to other attackers or groups to launch attacks. Current hotspots include the perennial Microsoft, but also Apple, which he describes as one of the worst reporters of security flaws.

Concluding, Mark Loveless offered a few suggestions. Continue to patch regularly, as the average time from patch to exploit is shrinking. Limit access to what is needed, because "locking down ACLs can save your bacon with regard to 0day." Finally, consider new types of technologies to reinforce your security, such as newer, more intelligent intrusion detection and protection systems.

INVITED TALK

System Administration: Drowning in Management Complexity

Chad Verbowski, Software Architect, Microsoft Research

Summarized by Raj Kumar Gurung (RK-Gurung2@wiu.edu)

This invited talk dealt with the growing complexities in systems and provided various approaches for systems management to aid system administrators in increasing the number of systems a single administrator can effectively manage. Complexity is constantly growing with the growing number of devices, applications, and users. Key pints were that we simply cannot rely on software advances

to address this complexity and that advances in the management space post-software development time are required. Verbowski proposed a data-driven management approach to reduce the complexity by using automated monitoring and analysis tools. The main advantage of this approach is that it traces all the interactions between the applications and configurations for analysis, thus providing simple troubleshooting space, reducing the problem space for other techniques, and leveraging existing machine-learning work. However, designers should always keep in mind scalability and cross-machine equivalence.

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 an overview of the revised moderation software and general housekeeping announcements. (Well, we really started by picking on Adam, on Andrew Hume, who earned 2005's Most Talkative Participant award, and on Trey Harris, who was 2004's Most Talkative by a factor of 2.)

We followed that with introductions around the room. For a variety of reasons, several of the Usual Suspects weren't at this year's workshop; in representation, businesses (including consultants) outnumbered universities by about 3 to 1; over the course of the day, the room included three LISA program chairs (one each past, present, and future; down from five last year) and five past or present members of the USENIX, SAGE, or LOPSA Boards (down from seven last year).

We went around the room to say how we believed system administration has changed in the past year. The general consensus seemed to be autonomic systems; challenges of new jobs; education of teachers to bring them up to what students know; improvements in automation; life-event changes, with marriages, deaths, and births; lower budgets; metrics; more fallout from legislation (such as SOX); more reliance on external infrastructure, such as external mail, calendar/scheduling systems, and wikis; organizational restructuring and staff turnover; targeted offshore security attacks; telecommunications integration; and rising virtualization.

Our first subject of discussion was storage. Several organizations have larger and larger storage needs; one example is a site that's growing at 10 TB a month or 2.5 PB a year, and smaller (such as 16-GB) drives no longer scale. Other places are more than doubling their data storage every year. We discussed some options, such as the promise of iSCSI/ZFS (with which current users are pleased, for the most part), and the forthcoming open source GFS-like. The comment about determining your real needs and taking metrics is important: Some 1/10-GB switches can't really switch among many users, and if you're not measuring end to end you won't know where your bottlenecks are.

In addition to primary storage needs (how much disk is needed? how is it attached? what bandwidth do you need?), there are ancillary issues, such as backups (do you store snapshots on disk? do you back up to tape or to another spinning disk? how much is really enough, given that nobody ever deletes data?). One point was to use software compression before the data hits the tape drive; for example, hardware compression can require 90 MB/s for 3:1 compression.

Another point was that if we do the math on ECC corrections, we find we're now having enough disks that at one site in particular we are seeing bit-rot rates in untouched files on spinning disks of about 1 error per several terabyte-years (1 TB spinning for 1 year, or 2 TB for 6 months). Yes, the rate is very very low, but it's definitely nonzero, so if you don't always checksum everything you have the risk of bit-rot and thus lost or corrupted data on disk (which leads to the issue of where you store your checksums and what happens if they themselves get bit-rot).

We digressed into a brief discussion of backups: Do you back up files just at the OS level or at the application level as well? Do you back up laptops? Rates of backup can differ among data types: The OS tends to change less frequently than home directories, for example. Finally, consider not backing up what you don't need (for legal, compliance, regulatory, and similar reasons). It's recommended that if you don't have a policy, you should write one yourself, then get approval or rewrites from your legal or compliance folks afterwards.

Our next large topic area for discussion was monitoring. We went around the room: 29% are using some kind of home-grown monitoring software, 83% are using open source tools, and only 8% are using commercial software. (You'll notice that these numbers won't add up to 100%, as several places use combinations.) The software packages explicitly mentioned include Big Brother, Cacti, cron that sends email on unexpected errors, home-grown syslog watcher, logcheck, MRTG, Nagios, NetCool, Net Vigil, OpenNMS, RRD,

smokeping, and Spyglass. Most people tolerate their monitoring, but very few are "very happy" with it. Nagios had the largest representation. The consensus seemed to be, "It's the best of the bad choices"; although most of us use it, nobody was an evangelist for it. In general, the suggestions are:

- Monitor what does happen that shouldn't.
- Monitor what didn't happen that should've.
- Monitor what you care about; don't monitor what you don't care about.
- Gather history: We may not care about one ping down, but we do care about multiple successive failures, and then we won't care until it comes back and stays up (the success table).

One problem is that we need more detail than just "up/down." Nagios as written doesn't differentiate among several states: Is it there (ping)? Does it have a heartbeat? Did it give a response? Did it give a valid response, and did it give a valid and timely response? The phrase "service is up" isn't necessarily meaningful. We discussed what we'd want in an ideal monitoring system, including cooperative signaling, so "If I take 10 minutes it's okay, if it's longer there's a problem" is a valid case.

Another issue we have with Nagios is that it often doesn't monitor the right things, or it performs the wrong tests. Who writes your tests? Is it the person responsible for the application, or a monitoring group, or someone else? The actions taken also need to be aware of criticality: How urgent is the problem? How often should it be tested for? and so on.

This led to a discussion about machine learning (monitoring tools that build or configure themselves) and self-aware applications that can determine on their own whether they have a problem and can send alerts themselves. Better application design can lead to better monitoring.

After our lunch break, we went through and mentioned tools new to us as individuals since last year's conference; the tools included Adobe Lightroom, Asterisk, Aware I Am, decoy MX server to block spammers, DocBook SGML, Dragon Naturally Speaking, drupal, Google Spreadsheets, hardware security monitors and crypto ("key roach motels"), IP KVM, IP power, IPMI cards, isolation booths at work for private phone calls, LYX, Mind Manager for mind mapping, Mori, OpenID, Password Safe, photography management (for births and weddings), Rails for admin interfaces, relationships with intellectual property lawyers, RSS feed-reading software, SQL Gray, Solaris 10, Solaris Zones and ZFS, Sparrow, USB-attached RFID readers, VOIP, wikis (because "they work now"), and x2vnc and x2x.

Next we talked in more detail about ZFS. Someone asked if it was as wonderful as the hype said it would be, and the answer boiled down to "Yes and no." For the most part, it's very very well designed. It does what you want, and even though it sounds too good to be true it's pretty close to that. However, if you use it long enough you'll see the warts. It works well with zones, but not everyone at Sun support knows enough to troubleshoot problems; so far, there's only one commercial product to back it up (Legato); there aren't any best practices; and there's no way to say, "Evacuate this disk and give it back to me."

Next we discussed calendaring. As a group we use a lot of software and at best we tolerate it. The big ones are Exchange's calendaring on the PC side and iCal on the Mac. We came up with a feature list of a good system, which included multi-OS, specifically Mac, Windows, Linux, Solaris, HP-UX, and *BSD; integrating both home and work calendars, keeping them separate so other "home" users (such as spouse and kids) can only see the "work" entries as "busy" without details; being able to see free/busy on others' calendars and to schedule events with negotiation, which requires ACLs of some kind. There's no good solution yet.

We next discussed cheap scientific clusters. Now that there are quad-CPU dual-core processors, someone built an inexpensive yet effective four-node (soon growing to ten-node) cluster with Infiniband Infinipath for internode communication and gigabit TCP/IP for networking. The cluster uses RAID 5 on the head node, and each node has 32 GB RAM. This cluster can almost make the decade-old Cray obsolete (since just about any job can use up to 32 GB of memory and the Cray has only 40 GB). It's doing better than expected, but it's very noisy.

This led us to a discussion about power consumption and heat generation. One site recently got a supercomputer grant for hardware that needs 300 tons of coolant, but its entire data center only has 45 tons; the entire campus doesn't use as much power as this one supercomputer will (once it's fully loaded). Going to virtual machines reduces power and heat by using several smaller virtual machines on one larger machine. Some articles say that DC power helps some, since you can avoid converting between DC and AC. There's not a huge market for better power consumption yet,

mainly because few people in the purchasing processes are discussing it, but if you require low-power, low-voltage, slower-but-cooler hardware in the hardware selection process, possibly by specifying "total wattage" instead of a number of systems, the market will auto-correct and give you more options. Other suggestions for reducing power consumption and heat generation included replacing your CRTs with LCD flat panels, using thin clients in conference rooms and secretarial desks where you don't need an entire PC (which has positive side effects on security), replacing desktops with permanently docked laptops, and replacing incandescent lights with compact fluorescent lights. Any and all of these can reduce your power costs, cooling costs, and fan noise.

After the afternoon break, we talked about support changes. As has been the case in recent years, more places are trying to do more—more services, more products, more projects, more hours of support—with fewer resources—fewer or the same number of people, fewer machines, and so on. In general, folks are accomplishing this by remote access (ssh into corporate environments, remote VNC to client, or customer machines supported from the technician's desk). There is also the issue of who supports home machines: Because they're used by the home and the corporation, they don't fit neatly into most support categories. It should be noted that supportability implies backups.

We next went around the room to discuss our most important or most difficult problems. This year, the big one was resource allocation: insufficient staff in both quantity and training, and insufficient time. Finding people is hard, keeping people can be hard (they quit or are reorganized away from your team), and cross-team communications is often hard. There are often too many fires to put out, so prioritizing which fire gets fought first is necessary. The other most common problem is the learning curve; several of us are in new environments and it's challenging first to learn what was done and why, and how things got into their current state, and then to improve things to use best practices; many resist change management, even at the level of "Tell someone when you change something." The third most common problem is career management: What can we do when we're getting bored with our current role, or if there's no growth path to "senior engineer"? Finally, compliance (for legal issues, such as HIPAA and SOX) is taking up more of our time; about 25% of us are doing more with it now than last year.

Finally, we discussed what's on our horizon, or what we expect the next year will be like for us. We predict that our challenges for the next 11 months will include application and OS upgrades back to the bleeding edge; clustering; compliance; exponential scaling; leading understaffed teams and dealing with staff retirement; making the infrastructure more reliable, more robust, and more reproducible; virtualization; and working with 10GigE.

CONFIGURATION MANAGEMENT TOOLS AND PRACTICE WORKSHOP

Summarized by Chris Cooke and Sanjai Narain (cc@inf.ed.ac.uk, narain@research.telcordia.com)

This year's workshop focused on configuration validation. Sanjai Narain presented the motivation. A central infrastructure management problem is testing whether infrastructure complies with end-to-end requirements. Requirements can be on functionality, security, performance or reliability, or those derived from government regulatory policies. Often, these span multiple components and layers of abstraction. Typical approaches to compliance testing, such as invasive testing and simulation, have significant limitations. A new approach that overcomes these limitations is noninvasive analysis of component configurations. These configurations represent the "source code" of infrastructure, in that deep prediction of infrastructure behavior can be made from their analysis. A new class of algorithms, analogous to that for static analysis of software, needs to be developed. This workshop brought together many researchers investigating this idea.

Dinesh Verma presented his work expressing configuration constraints as policies, and then ensuring conformance with those policies. This work has been applied to configuration validation of storage area networks.

Rajesh Talpade discussed a software system called VCAS for vulnerability and compliance assessment for IP networks. Over the past two years VCAS has been successfully undergoing trials at the infrastructure of six major enterprises. It is based on patent-pending algorithms for diagnosing vulnerabilities such as single points of failure and those arising out of interactions between protocols. It contains a proprietary, vendor-neutral knowledge-base of rules covering most IP network protocols.

Srinivas Bangarbale discussed challenges of managing change in mid-sized enterprises. Configuration management is a much needed discipline but many factors stand in the way of a successful configuration management practice: organizational culture, the need for flexibility, and operating constraints. Whereas large organizations can afford the overheads of a full-fledged configuration management practice, and small ones may not need to be as rigorous as the large ones, mid-sized enterprises are frequently caught between the two extremes and find the good solution to be a tough balancing act.

Geoffrey Xie argued that in order to turn network configuration into a principled engineering process, it is critical to develop a class of high-level abstractions, each equipped with a unifying analytical framework for reasoning about the joint effect of related low-level mechanisms and protocols. He introduced such a high-level abstraction, called the reachability matrix, for modeling network security policy.

John Orthoefer discussed configuration management from the systems administrator perspective, including what works in real-life situations, from small sites with fewer than 10 people and a few hundred machines, to large sites with more than 20 people and thousands of machines over a geographically dispersed area. The concept of what is a "valid configuration" and how one arrives at that configuration differs for these two cases.

Sanjay Rao discussed two key challenges to establishing configuration management as a rigorous scientific discipline in academia. First, access to configuration data is limited. Second, evaluating solutions and demonstrating "success" is also difficult. To address these challenges, his team is setting up a "white-box" approach involving extensive collaboration with network operators, including Purdue campus operators and AT&T. His goal is to empirically study operational networks with a view to systematically understanding operational practice and scientifically quantifying trade-offs managers are making while designing networks.

Yiyi Huang presented a technique for improving fault-isolation by analyzing network-wide routing configuration information. For the Abilene network, this technique detected every reported disruption to internal nodes and links and correctly explained the scenarios that caused the disruptions.

Paul Anderson presented an overview of the LCFG configuration management tool, along with thoughts on how it could support configuration validation and synthesis.

Panel Discussion on Configuration Management: The Big Picture

Alva Couch started the discussion by saying that we need a formal way to express constraints and specifications: He pointed out that during the workshop all the participants had been doing this in English. But what sort of thing would be acceptable to, and used by, system administrators? Sanjai Narain suggested that first-order logic would be suitable. He enquired whether it would be useful to embed this into languages such as Perl that system administrators already know and understand. Mark Burgess said that we need to model behavior, not specifications. The language must encompass uncertainties: unreliability, voluntary co-operation, access restrictions. He disagreed about the suitability of first-order logic because it has no typing. A meta-model is needed to relate things to each other.

Panel Discussion on Are Tools Enough?

Alva Couch also led this discussion at the end of the day. For the discussion he invited Æleen Frisch, Tom Limoncelli, and David Parter. The invited guests made a collective plea for simplicity and ease of use in configuration management tools, reiterating and emphasizing a point that had been made by John Orthoefer earlier in the day.

Æleen said that part-time, busy system administrators will not use configuration management tools if they're too complicated for the matter at hand. Needed are proper, real, more comprehensive examples of how to do realistic, valuable, real-world tasks with configuration management tools.

Tom Limoncelli made a similar point. Configuration management tools have a huge barrier to entry. He suggested that the main tool developers spend the next year removing that big barrier. He suggested stopping all-or-nothing solutions and starting with just one little hack. He made a memorable plea: "Stop making tools for smart people!" It really would be better for vendors to adopt mediocre configuration management standards than have wonderful configuration management that nobody uses. Get the vendors on board. Get standards that marketing people can boast about. He also suggested an approach to achieving this. To help a configuration management tool grow in popularity, get the authors of the ten most popular

open source software packages to support and provide hooks for your configuration management tool.

Mark Burgess objected to the implication that there are currently no configuration management standards, pointing out that international telecommunications companies do already have standards in this area, to which they are required to work.

Kent Skaar elaborated on the examples: We need not just examples, but explanations of the thought processes behind them; we need design patterns.

Luke Kanies pointed out a bootstrapping problem: A selection of real-world, useful, usable examples to accompany a configuration management tool can only come from the tool's user community. How can a community be built around the tool in order to get the examples, without the examples already existing? He would like Puppet to have such examples, but "there is no community."

Tom Limoncelli also emphasized the quick-changing nature of business process: A company's business process tends to be driven by what magazine the CEO just read! Such things are ephemeral; a new one will be along next week. Nevertheless, system administrators seem to have to spend a great deal of effort dealing with arbitrary, ad-hoc, unsuitable, or unworkable technical diktat coming from nontechnical management. Also, a lot of system administration involves systems that have been running for years and were set up by staff long since departed; such work can be termed "system archaeology." Configuration management tools have to be able to deal with such suboptimal real-world circumstances.

Main Ideas from the Workshop

Two ideas seem to have predominated. First, there was Alva Couch's observation that we should move from managing components to managing architecture. This was reiterated in his talk and paper presented at the conference. Configurations can be more helpfully represented as an interlocking mesh of interrelated "aspects" than as a lot of individual configuration parameters.

The second idea to be presented again and again was a plea for simplicity and ease of use of configuration management tools. System administrators often find them too complex and frightening to adopt. Easy routes to adoption have to be provided before a large-scale take-up of configuration management tools can take place.

About three dozen people attended the workshop. Of these, 12 were from academia, 1 was a consultant, 15 were configuration management tool developers, 15 did configuration management–related research, and 15 were new to the workshop.

For additional information, see http://homepages.inf.ed.ac.uk/group/lssconf/config2006/index.html.

Announcement and Call for Participation

21st Large Installation System Administration Conference (LISA '07)

Sponsored by USENIX and SAGE

http://www.usenix.org/lisa07

November 11–16, 2007 **Dallas, TX, USA**

Important Dates

Extended abstract and paper submissions due: *May 14, 2007*
Invited talk proposals due: *May 21, 2007*
Notification to authors: *June 27, 2007*
Final papers due: *August 20, 2007*
Poster proposals due: *September 3, 2007*
Notification to poster presenters: *September 17, 2007*

Conference Organizers

Program Chair
Paul Anderson, *University of Edinburgh*

Program Committee
Charlie Catlett, *NSF Teragrid Project*
William Cheswick, *Consultant, cheswick.com*
Alva Couch, *Tufts University*
Æleen Frisch, *Exponential Consulting*
Peter Baer Galvin, *Corporate Technologies, Inc.*
Andrew Hume, *AT&T Labs–Research*
William LeFebvre, *Independent Consultant*
Adam Moskowitz, *Menlo Computing*
Sanjai Narain, *Telcordia Technologies*
Melanie Rieback, *Vrije Universiteit Amsterdam*
Kent Skaar, *Bladelogic*
Chad Verbowski, *Microsoft Research*

Invited Talk Coordinators
Rudi van Drunen, *Competa IT/Xlexit*
Doug Hughes, *D.E. Shaw Research, LLC*

Workshops Coordinator
Lee Damon, *University of Washington*

Guru Is In Coordinators
Philip Kizer, *Estacado Systems*
John "Rowan" Littell, *California College of the Arts*

Hit the Ground Running Track Coordinator
Adam Moskowitz, *Menlo Computing*

Work-in-Progress Reports and Posters Coordinator
Brent Hoon Kang, *University of North Carolina at Charlotte*

Overview

For twenty years, the annual LISA conference has been the foremost worldwide gathering for everyone interested in the technical and administrative issues of running a large computing facility. The distinctive blend of principles and practice makes this the meeting place of choice for a wide range of participants:

- Experienced administrators meeting their peers to discuss the latest tools and techniques, or junior administrators taking tutorials from worldwide experts.
- Practicing system administrators, managers, consultants, educators, and students.
- Researchers in computer science or human factors looking to test their ideas on real-world problems.
- People from business, government, or academia, in over 30 different countries.
- Specialists in areas such as security, configuration, networking, or autonomics, as well as general system administrators.
- People working at sites with tens of machines or tens of thousands of machines.
- Administrators working with a wide range of operating systems from Linux and the BSD releases to vendor-specific systems such as Solaris, Windows, Mac OS, HP-UX, and AIX.

All of these people have a role to play in contributing to the unique character of the LISA conference, and there are many ways for you to participate:

- Submit a draft paper or extended abstract for a refereed paper.
- Propose a tutorial topic.
- Suggest an invited talk speaker or topic.
- Share your experience by leading a Guru Is In session.
- Submit a proposal for a workshop.
- **New!** Submit a poster.
- Present a Work-in-Progress Report (WiP).
- Organize a Birds-of-a-Feather (BoF) session.
- Email an idea to the program chair.

Refereed Papers

Effective administration of a large site requires a good understanding of modern tools and techniques, together with their underlying principles—but the human factors involved in managing and applying these technologies in a production environment are equally important. Bringing together theory and practice is an important goal of the LISA conference, and practicing system administrators, as well as academic researchers, all have valuable contributions to make. A selection of possible topics for refereed papers appears in a separate section below, but submissions are welcome on any aspect of system administration, from the underlying theory of a new configuration technique to a case study on the management of a successful site merger.

Whatever the topic, it is most important that papers present results in the context of current practice and previous work: they should provide references to related work and make specific comparisons where appropriate. Papers should also contain work that is different from anything that has been published previously. Careful searching for publications on a similar theme will help to identify any possible duplication and provide pointers to related work; the USENIX site contains most previous LISA conference proceedings,

which may provide a starting point when searching for related publications: http://www.usenix.org/events/byname/lisa.html.

Proposal and Submission Details

Anyone who would like help in writing a proposal should contact the program chair at lisa07chair@usenix.org. The conference organizers are keen to make sure that good work gets published, and we are happy to help at any stage in the process.

Proposals may be submitted as draft papers or extended abstracts.

- **Draft papers:** This is the preferred format. A draft paper proposal is limited to 16 pages, including diagrams, figures, references, and appendices. It should be a complete or near-complete paper, so that the Program Committee has the best possible understanding of your ideas and presentation.
- **Extended abstracts:** An extended abstract proposal should be about 5 pages long (at least 500 words, not counting figures and references) and should include a brief outline of the final paper. The form of the full paper must be clear from your abstract. The Program Committee will be attempting to judge the quality of the final paper from your abstract. This is harder to do with extended abstracts than with the preferred form, draft papers, so your abstract must be as helpful as possible in this process to be considered for acceptance.

Paper authors are also invited to submit posters, as outlined below, to accompany their presentations; these provide an overview of the work and a focal point for delegates to meet with the author.

As with most conferences and journals, LISA requires that papers not be submitted simultaneously to more than one conference or publication and that submitted papers not be previously or subsequently published elsewhere for a certain period of time.

General submission rules:

- All submissions must be electronic, in ASCII or PDF format only. Proposals must be submitted using the Web form located on the LISA '07 Call for Papers Web site, http://www.usenix.org/lisa07/cfp.
- Submissions containing trade secrets or accompanied by nondisclosure agreement forms are not acceptable and will be returned unread. As a matter of policy, all submissions are held in the highest confidence prior to publication in the conference proceedings. They will be read by Program Committee members and a select set of designated external reviewers.
- Submissions whose main purpose is to promote a commercial product or service will not be accepted.
- Submissions may be submitted only by the author of the paper. No third-party submissions will be accepted.
- All accepted papers must be presented at the LISA conference by at least one author. One author per paper will receive a registration discount of $200. USENIX will offer a complimentary registration for the technical program upon request.
- Authors of an accepted paper must provide a final paper for publication in the conference proceedings. Final papers are limited to 16 pages, including diagrams, figures, references, and appendices. Complete instructions will be sent to the authors of accepted papers. To aid authors in creating a paper suitable for LISA's audience, authors of accepted proposals will be assigned one or more shepherds to help with the process of completing the paper. The shepherds will read one or more intermediate drafts and provide comments before the authors complete the final draft.
- Simultaneous submission of the same work to multiple venues, submission of previously published work, and plagiarism constitute dishonesty or fraud. USENIX, like other scientific and technical conferences and journals, prohibits these practices and may, on the recommendation of a program chair, take action against authors who have committed them. In some cases, to ensure the integrity of papers under consideration, program committees may share information about submitted papers with other conference chairs and journal editors. If a violation of these principles is found, sanctions may include, but are not limited to, barring the authors from submitting to or participating in USENIX conferences for a set period, contacting the authors' institutions, and publicizing the details of the case. Authors uncertain whether their submission meets USENIX's guidelines should contact the program chair, lisa07chair@usenix.org, or the USENIX office, submissionspolicy@usenix.org.

For administrative reasons, every submission must list:

1. Paper title, and names, affiliations, and email addresses of all authors. Indicate each author who is a full-time student.
2. The author who will be the contact for the Program Committee. Include his/her name, affiliation, paper mail address, daytime and evening phone numbers, email address, and fax number (as applicable).

For more information, please consult the detailed author guidelines at http://www.usenix.org/events/lisa07/cfp/guidelines.html. **Proposals are due May 14, 2007.** Authors will be notified by June 27 whether their papers have been accepted.

Training Program

LISA offers state-of-the-art tutorials from top experts in their fields. Topics cover every level from introductory to highly advanced. You can choose from over 50 full- and half-day tutorials ranging from performance tuning through Linux, Solaris, Windows, Perl, Samba, network troubleshooting, security, network services, filesystems, backups, Sendmail, spam, and legal issues, to professional development.

To provide the best possible tutorial offerings, USENIX continually solicits proposals and ideas for new tutorials. If you are interested in presenting a tutorial or have an idea for a tutorial you would like to see offered, please contact the Training Program Coordinator, Daniel V. Klein, at tutorials@usenix.org.

Invited Talks

An invited talk discusses a topic of general interest to attendees. Unlike a refereed paper, this topic need not be new or unique but should be timely and relevant or perhaps entertaining. A list of suggested topics is available in a separate section below. An ideal invited talk is approachable and possibly controversial. The material should be understandable by beginners, but the conclusions may be disagreed with by experts. Invited talks should be 60–70 minutes long, and speakers should plan to take 20–30 minutes of questions from the audience.

Invited talk proposals should be accompanied by an abstract of less than one page in length describing the content of the talk. You can also propose a panel discussion topic. It is most helpful to us if you suggest potential panelists. Proposals of a business development or marketing nature are not appropriate. Speakers must submit their own proposals; third-party submissions, even if authorized, will be rejected.

Please email your proposal to lisa07it@usenix.org. **Invited talk proposals are due May 21, 2007.**

The Guru Is In Sessions

Everyone is invited to bring perplexing technical questions to the experts at LISA's unique Guru Is In sessions. These informal gatherings are organized around a single technical area or topic. Email suggestions for Guru Is In sessions or your offer to be a Guru to lisa07guru@usenix.org.

Workshops

One-day workshops are hands-on, participatory, interactive sessions where small groups of system administrators have an opportunity to discuss a topic of common interest. Workshops are not intended as tutorials, and participants normally have significant experience in the appropriate area, enabling discussions at a peer level. However, attendees with less experience often find workshops useful and are encouraged to discuss attendance with the workshop organizer.

A workshop proposal should include the following information:

- Title
- Objective
- Organizer name(s) and contact information
- Potential attendee profile
- Outline of potential topics

Please email your proposal to lisa07workshops@usenix.org.

New in 2007: Posters

This year's conference will include a poster session. This is an opportunity to display a poster describing recent work. The posters will be on display during the conference, and fixed times will be advertised when authors should be present to discuss their work with anyone who is interested. This provides a very good opportunity to make contact with other people who may be interested in the same area. Student posters, as well as practitioners sharing their experiences, are particularly welcome. For further information or to submit details of a poster, send email to lisa07posters@usenix.org before September 3, 2007. Accepted poster authors will be notified by September 17, and completed posters will be required by the start of the conference. Poster presenters who would also like to give a short presentation may also register for a WiP (see below).

Work-in-Progress Reports (WiPs)

A Work-in-Progress Report (WiP) is a very short presentation about current work. It is a great way to poll the LISA audience for feedback and interest. We are particularly interested in presentations of student work. To schedule a short presentation, send email to lisa07wips@usenix.org or sign up on the first day of the technical sessions.

Birds-of-a-Feather Sessions (BoFs)

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

Possible Topics for Authors and Speakers

Technical Challenges

- Authentication and authorization: "Single-signon" technologies, identity management
- Autonomic computing: Self-repairing systems, zero administration systems, fail-safe design
- Configuration management: Specification languages, configuration deployment
- Data center design: Modern methods, upgrading old centers
- Email: Mail infrastructures, spam prevention
- Grid computing: Management of grid fabrics and infrastructure
- Mobile computing: Supporting and managing laptops and remote communications
- Multiple platforms: Integrating and supporting multiple platforms (e.g., Linux, Windows, Macintosh)
- Networking: New technologies, network management
- Security: Malware and virus prevention, security technologies and procedures
- Standards: Enabling interoperability of local and remote services and applications
- Storage: New storage technologies, remote filesystems, backups, scaling
- Web 2.0 technologies: Using, supporting and managing wikis, blogs, and other Web 2.0 applications
- Virtualization: Managing and configuring virtualized resources

Professional Challenges

- Communication: Tools and procedures for improving communication among administrators and users
- Consolidation: Merging and standardizing infrastructures and procedures
- Devolution: Managing dependence on devolved services (calendars, mail, Web 2.0, etc.) and users
- Flexibility: Responding effectively to changes in technology and business demands
- In-house development: The (dis)advantages and pitfalls of in-house technology development
- Legislation: Security, privacy
- Management: The interface and transition between "technical" and "managerial"
- Metrics: Measuring and analyzing the effectiveness of technologies and procedures
- Outsourcing/offshoring system administration: Is it possible?
- Proactive administration: Transitioning from a reactive culture
- Standardizing methodologies: Sharing best practice
- Training and staff development: Developing and retaining good system administrators
- User support: Systems and procedures for supporting users

Contact the Chair

The program chair, Paul Anderson, is always open to new ideas that might improve the conference. Please email any and all ideas to lisa07ideas@usenix.org.

Final Program and Registration Information

Complete program and registration information will be available in August 2007 at the conference Web site, http://www.usenix.org/lisa07. If you would like to receive the latest USENIX conference information, please join our mailing list at http://www.usenix.org/about/mailing.html.

Sponsorship and Exhibit Opportunities

The oldest and largest conference exclusively for system administrators presents an unparalleled marketing and sales opportunity for sponsoring and exhibiting organizations. Your company will gain both mind share and market share as you present your products and services to a prequalified audience that heavily influences the purchasing decisions of your targeted prospects. For more details please contact exhibits@usenix.org.

2007 USENIX Annual Technical Conference

June 17–22, 2007, Santa Clara, California, USA

Join us in Santa Clara, CA, June 17–22, for the 2007 USENIX Annual Technical Conference. USENIX Annual Tech has always been the place to present groundbreaking research and cutting-edge practices in a wide variety of technologies and environments. USENIX '07 will be no exception.

USENIX '07 will feature:

- An extensive Training Program, covering crucial topics and led by highly respected instructors
- Technical Sessions, featuring the Refereed Papers Track, Invited Talks, and a Poster Session
- Plus BoFs and more!

http://www.usenix.org/usenix07/login

;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